

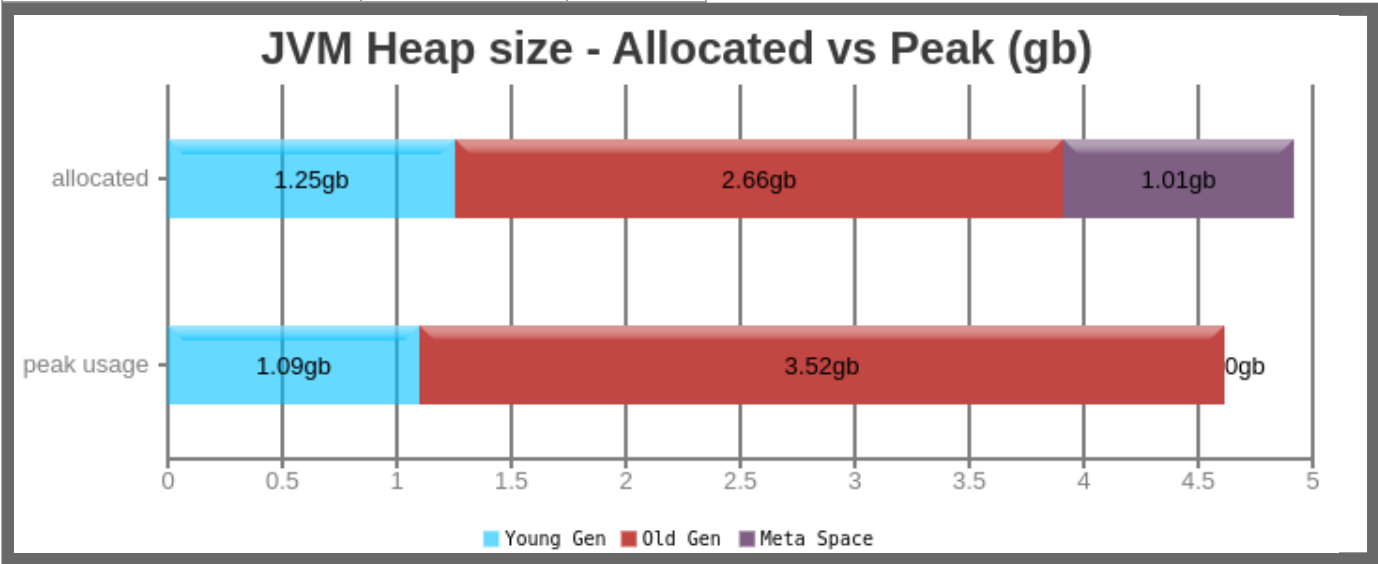


Analysis Report

JVM Heap Size

Generation	Allocated 	Peak 
Young Generation	1.25 gb	1.09 gb
Old Generation	2.66 gb	3.52 gb
Meta Space	1.01 gb	3.81 mb
Young + Old + Meta space	4.91 gb	3.91 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/))

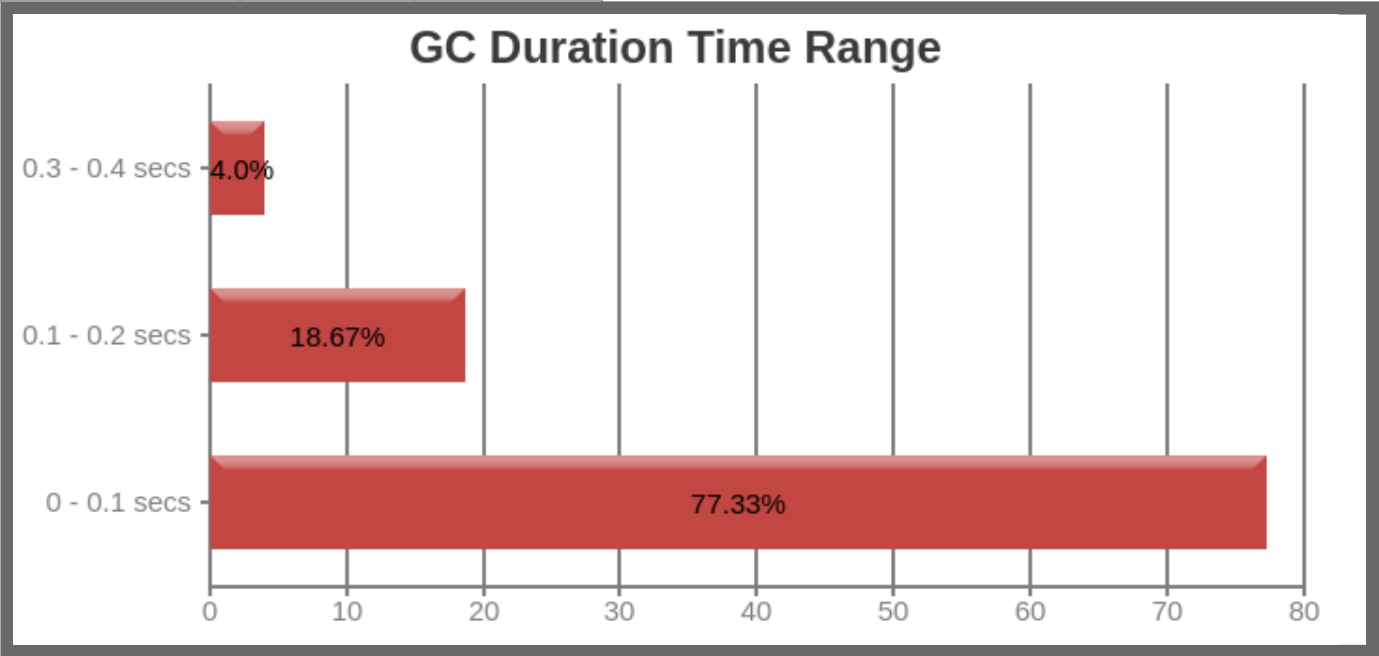
1 Throughput  : 59.184%

2 Latency:

Avg Pause GC Time 	59 ms
Max Pause GC Time 	390 ms

GC **Pause** Duration Time Range 

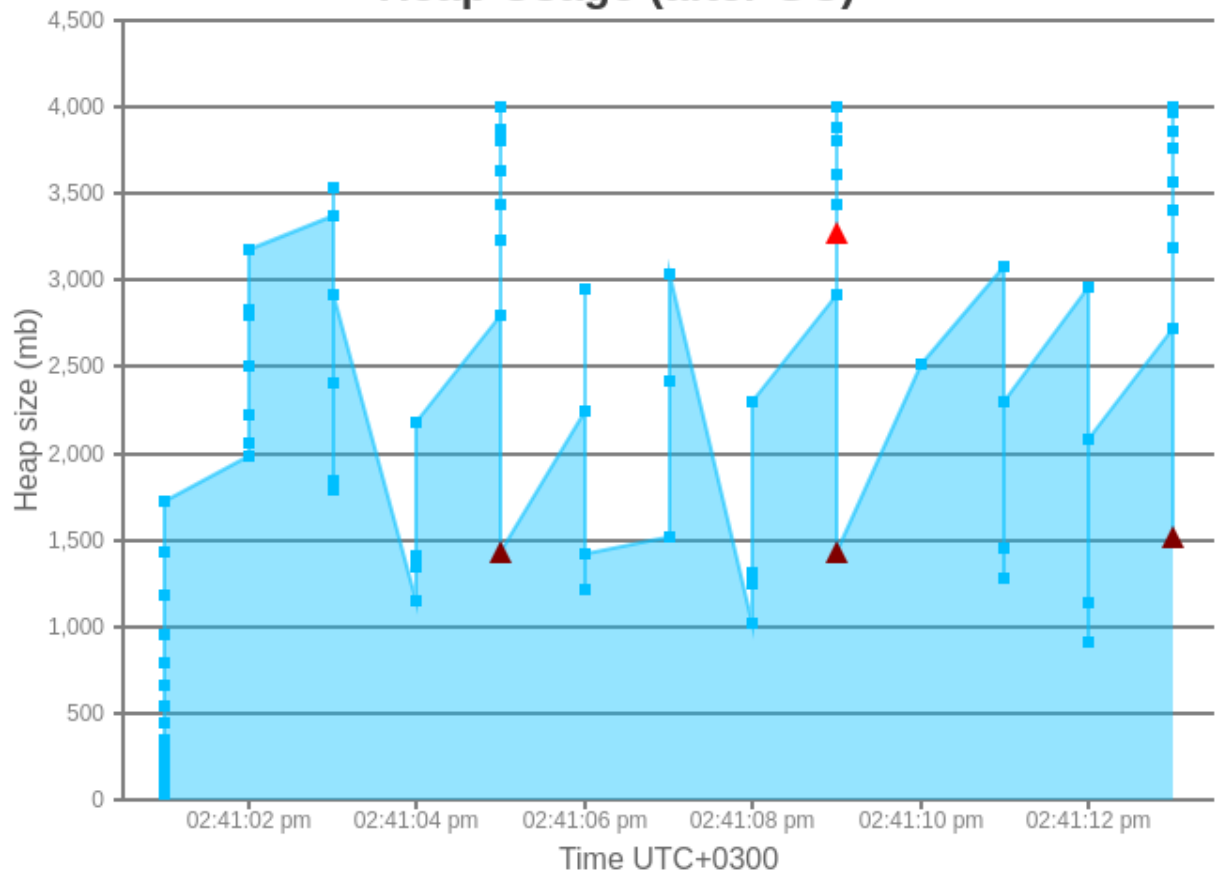
Duration (secs)	No. of GCs	Percentage
0 - 0.1	58	77.333%
0.1 - 0.2	14	96.0%
0.3 - 0.4	3	100.0%



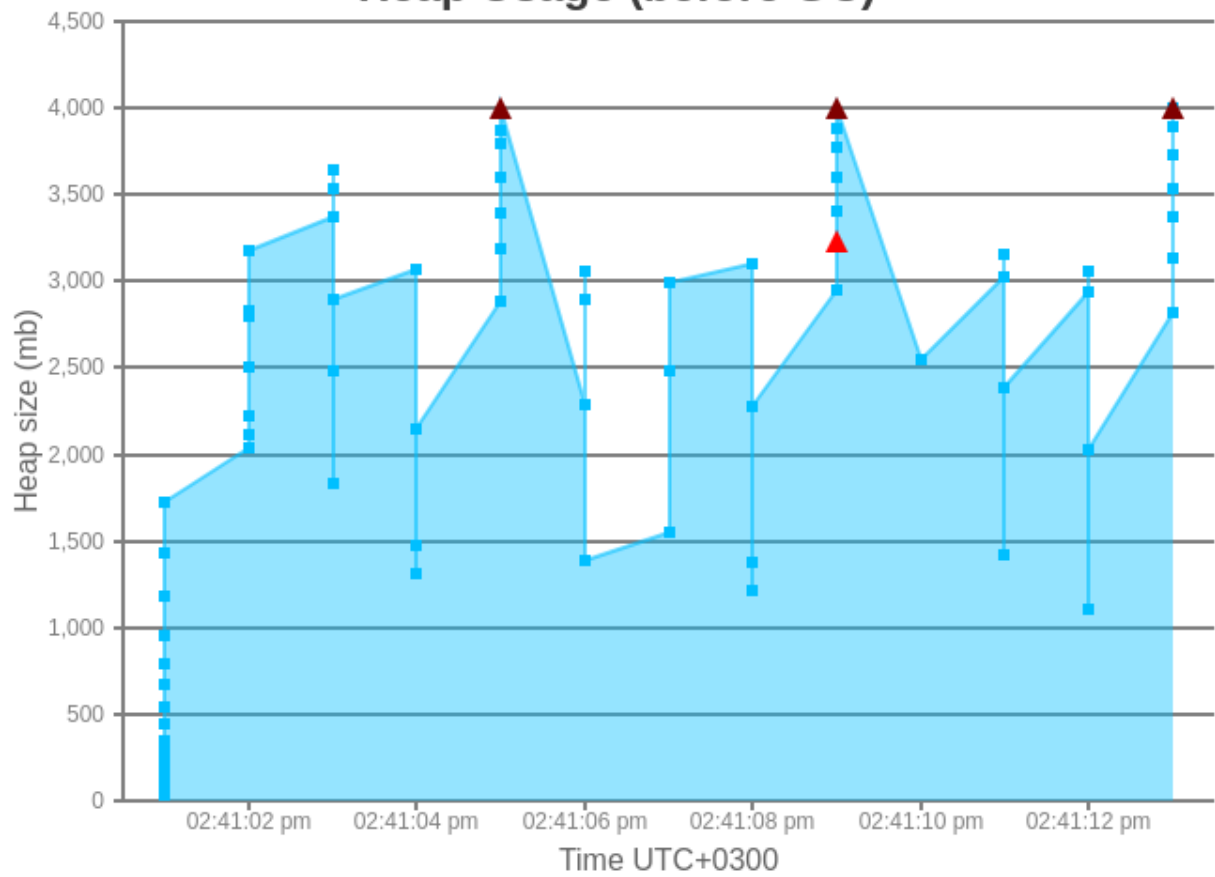
.||| Interactive Graphs

(All graphs are zoomable)

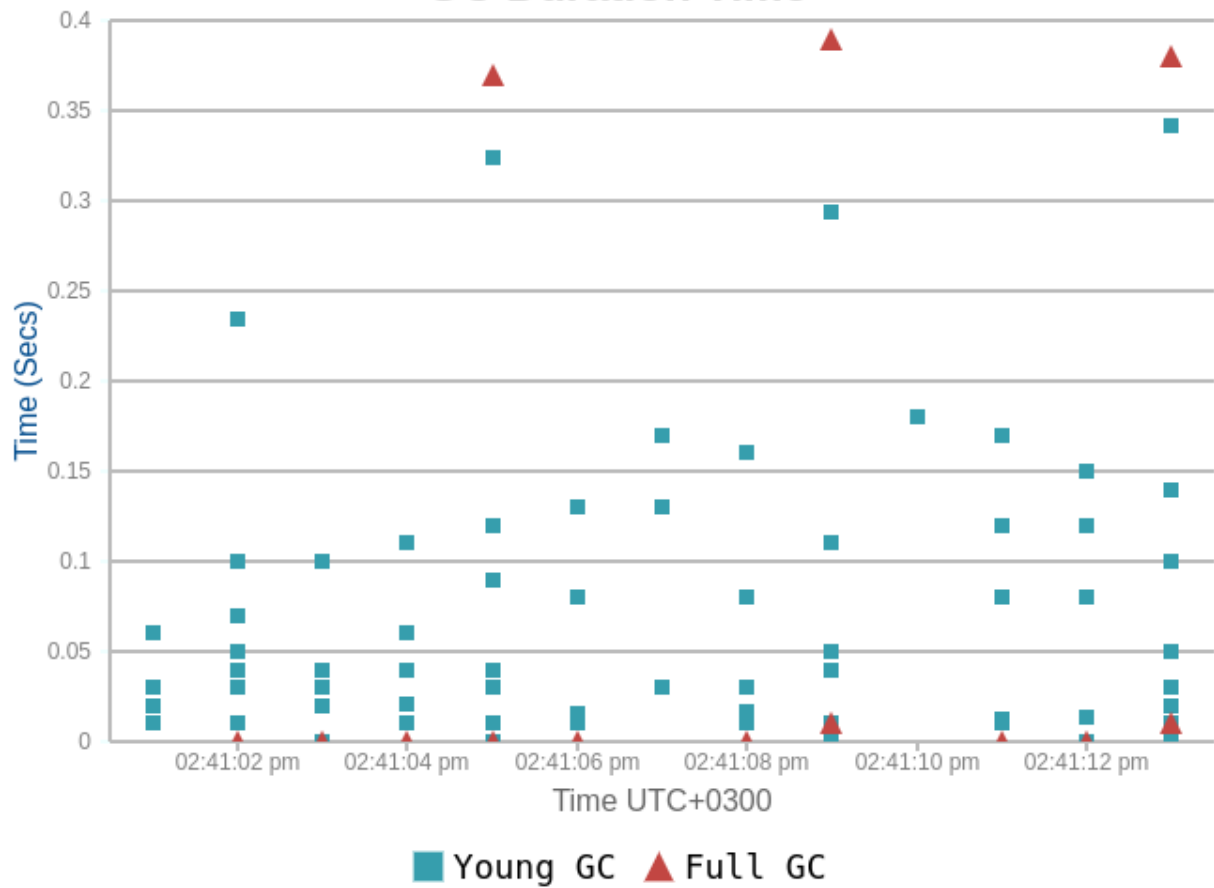
Heap Usage (after GC)



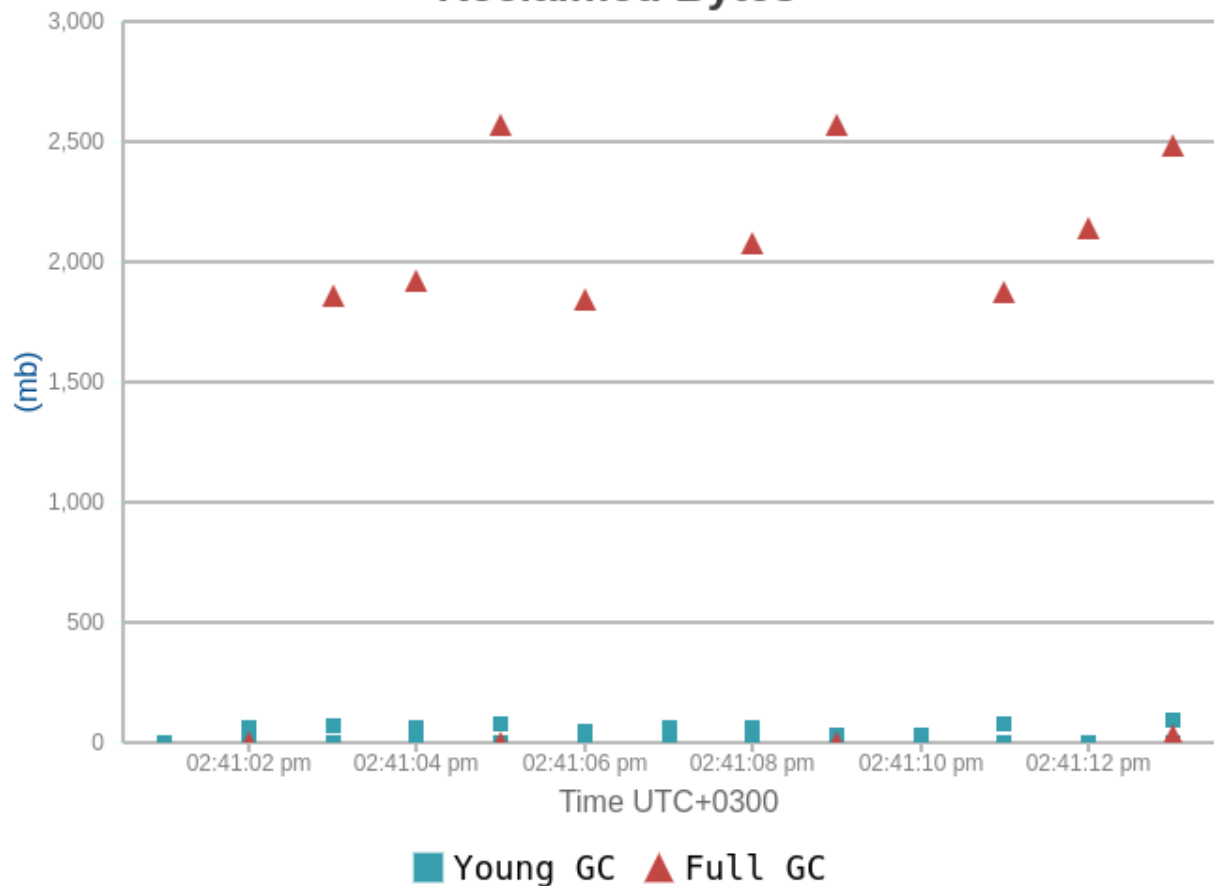
Heap Usage (before GC)



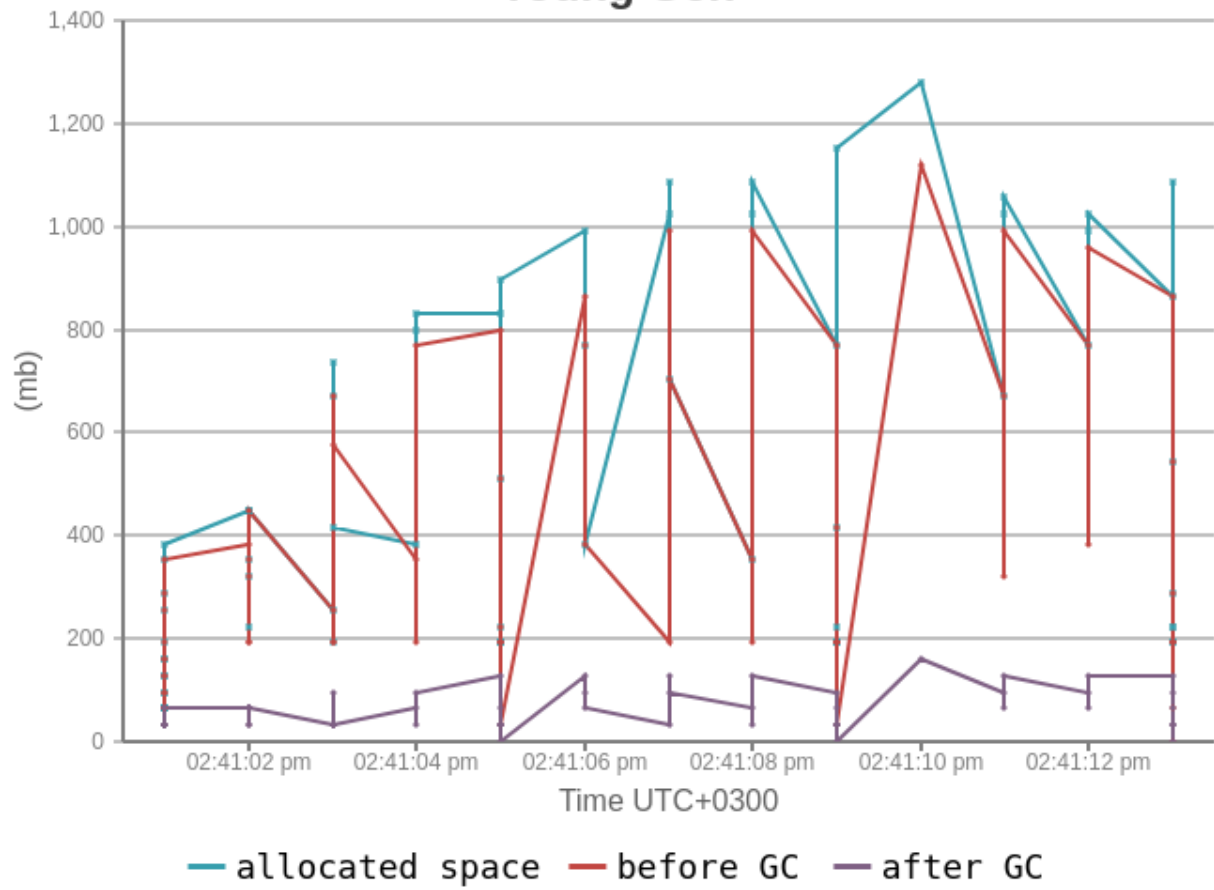
GC Duration Time



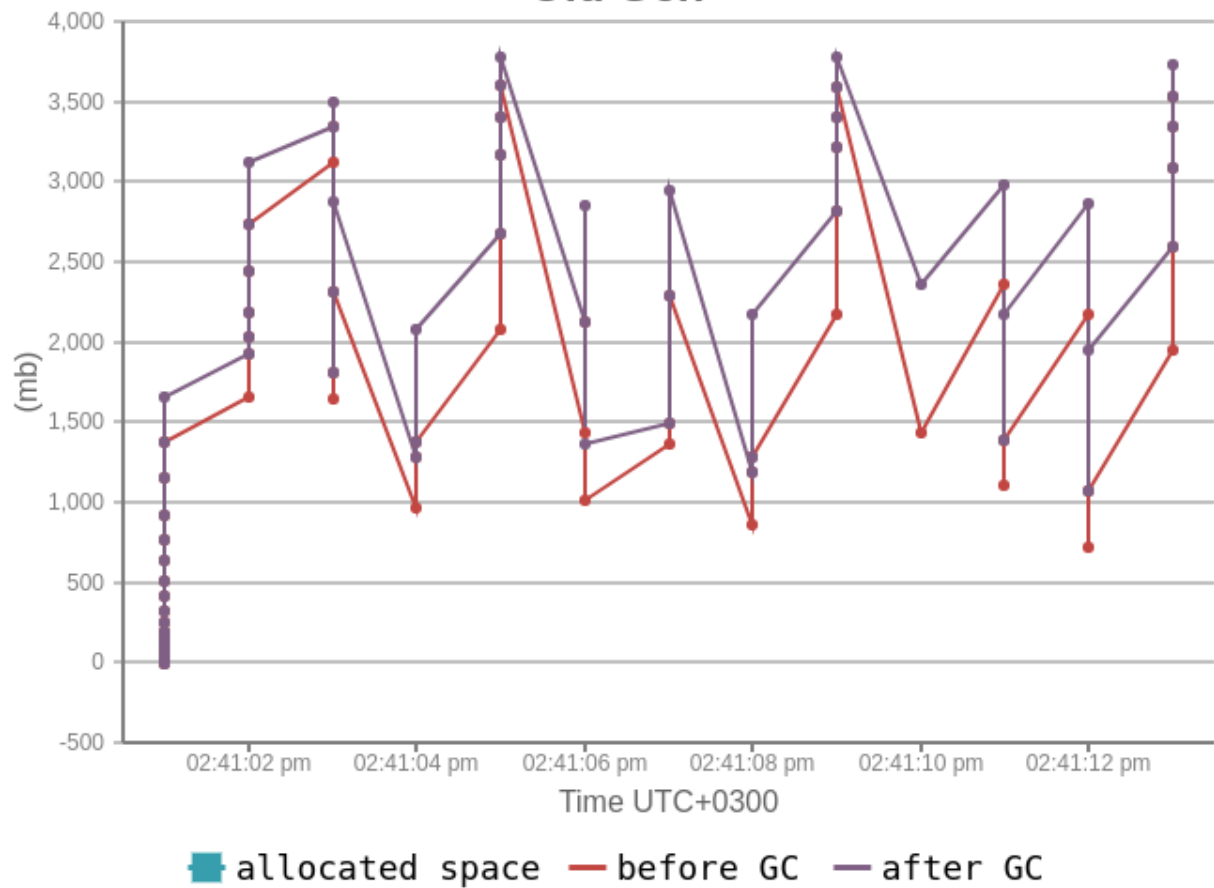
Reclaimed Bytes



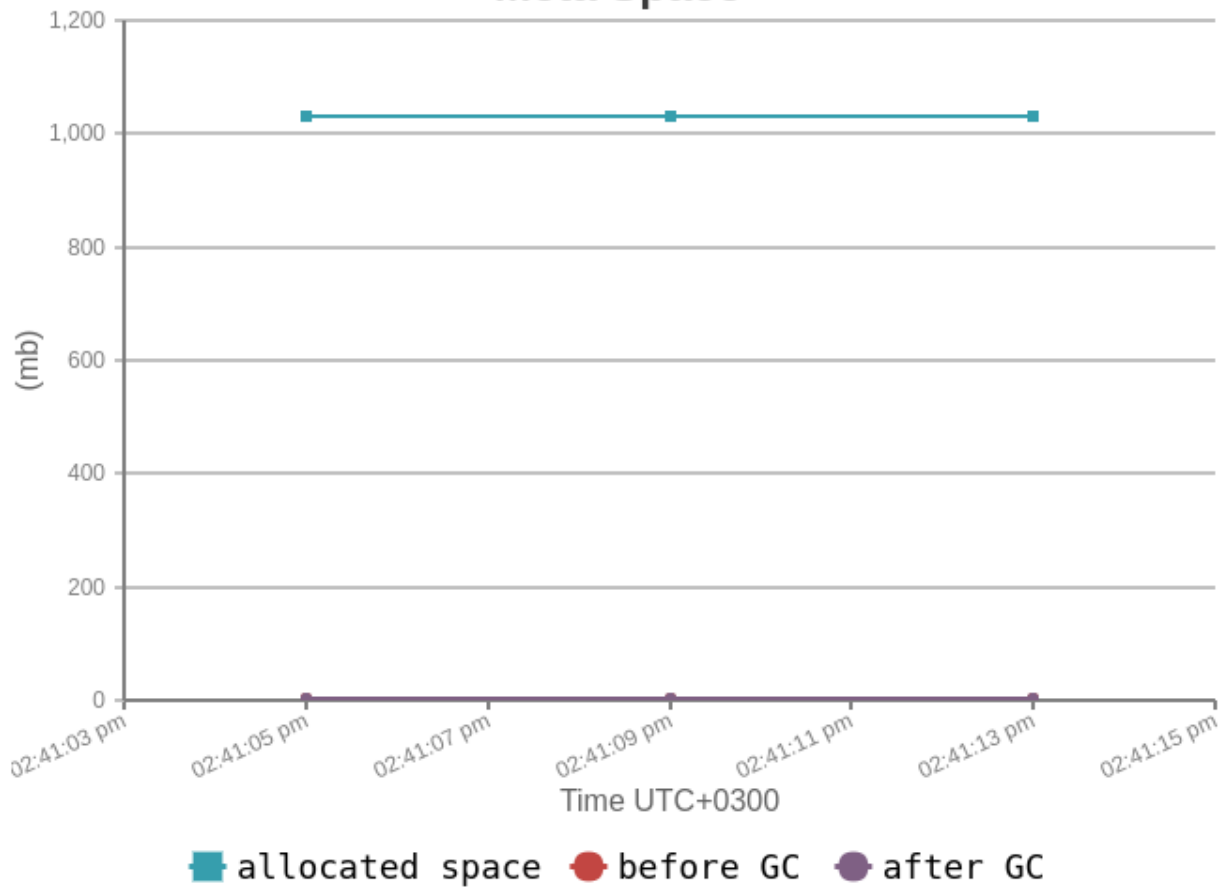
Young Gen



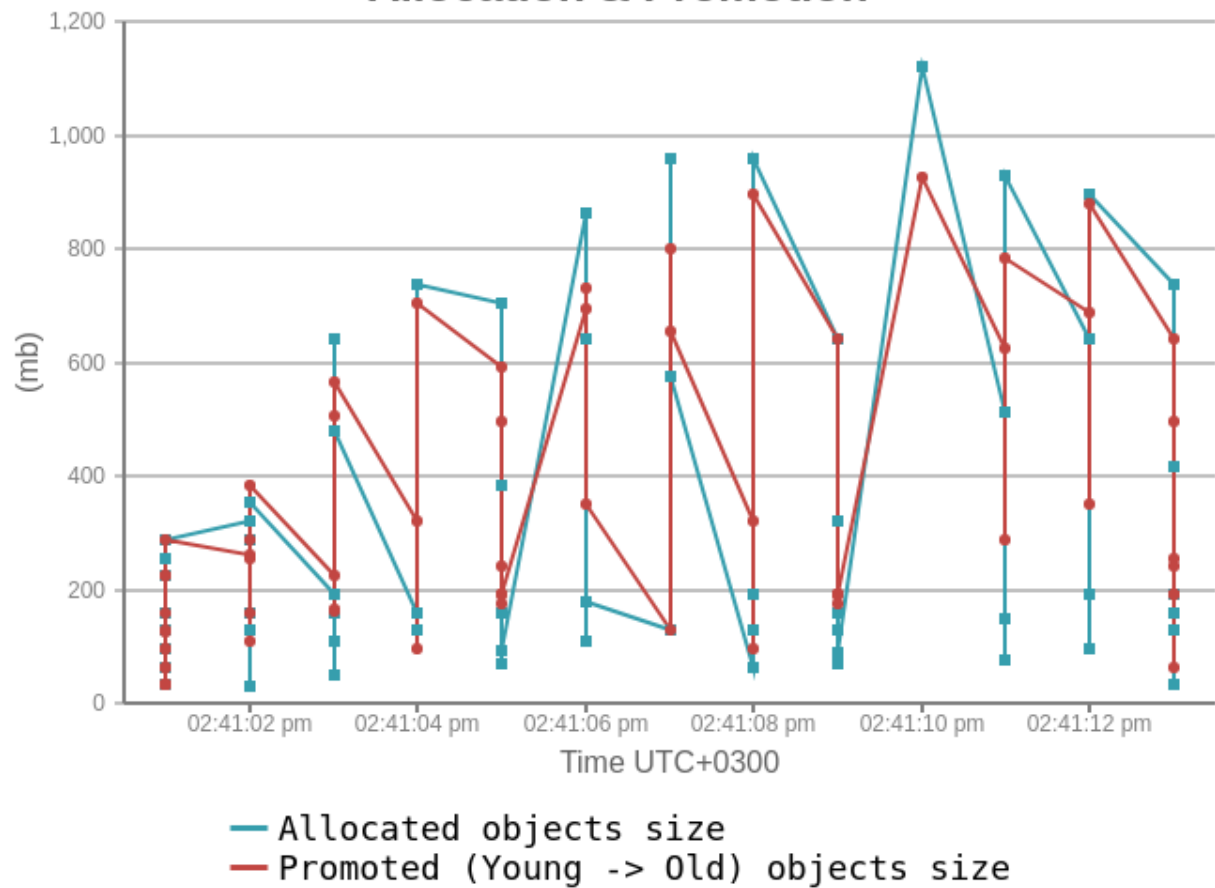
Old Gen



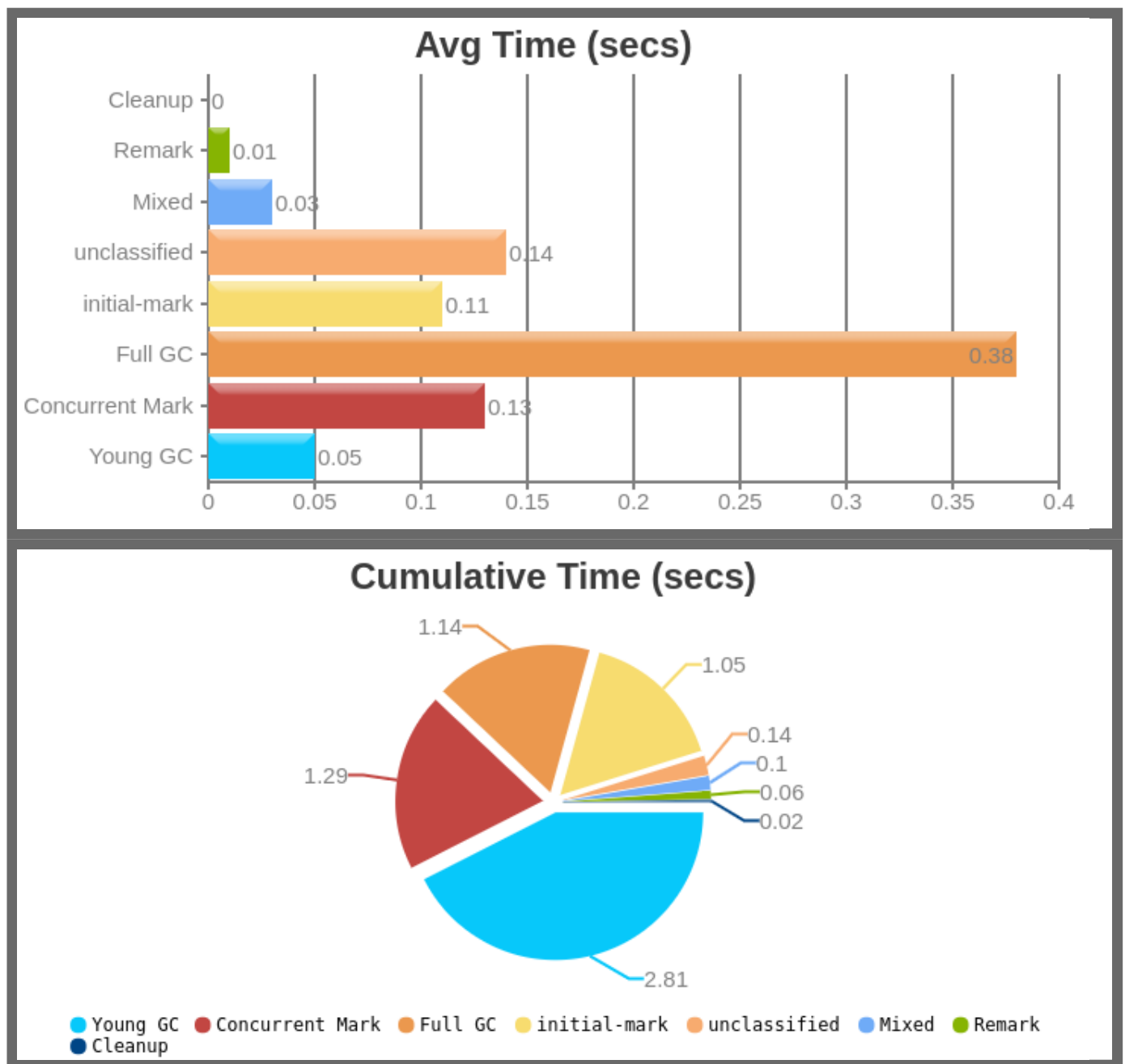
Meta Space



Allocation & Promotion



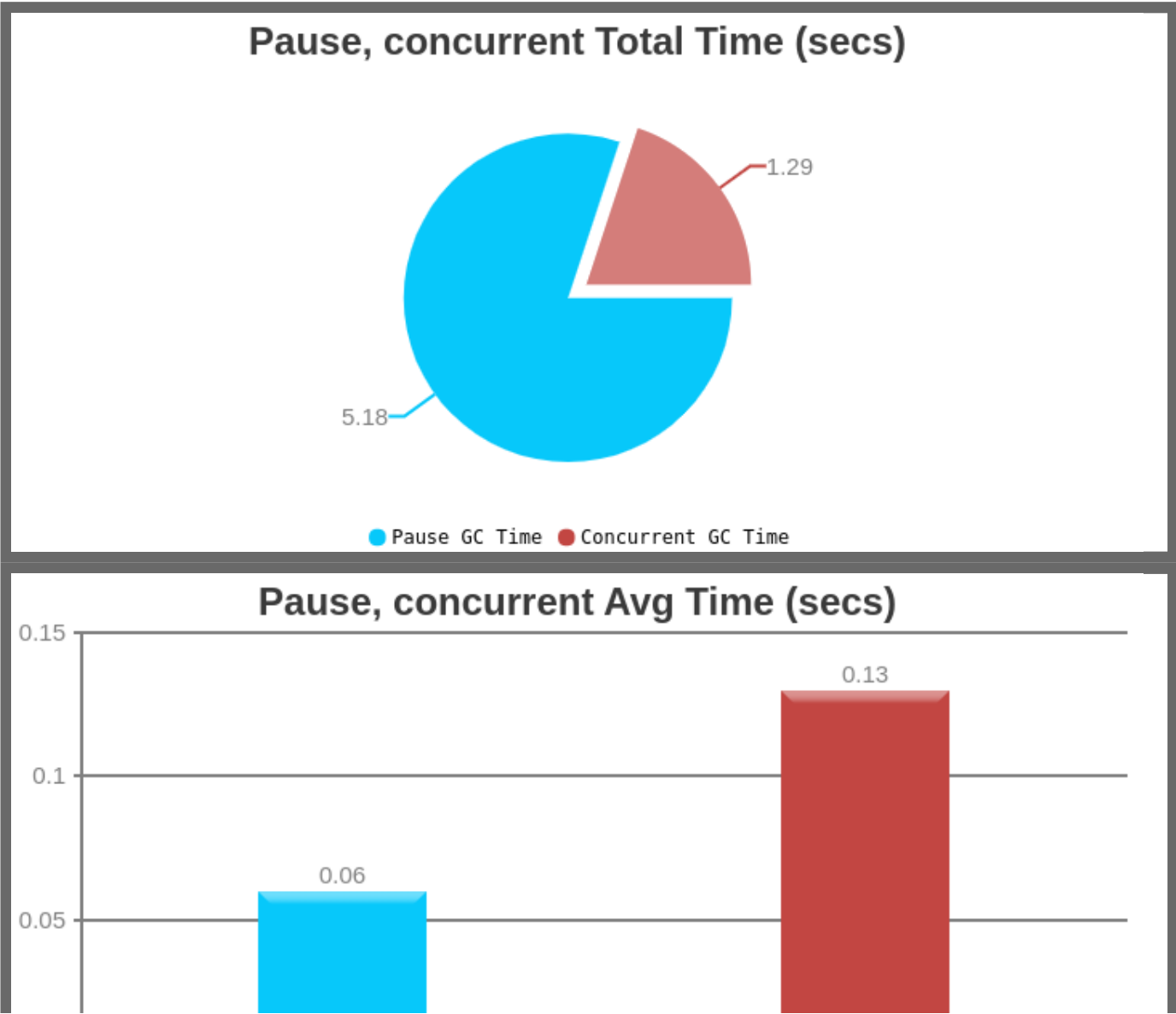
🔧 G1 Collection Phases Statistics



	Young GC 🟡	Concurrent Mark	Full GC 🟠	initial-mark 🟡	unclassified	Mixed 🟢	Remark 🟢	Cleanup 🟠	Total
Count 📊	52	10	3	10	1	3	10	10	99
Total GC Time 🕒	2 sec 810 ms	1 sec 292 ms	1 sec 140 ms	1 sec 50 ms	140 ms	100 ms	60 ms	20 ms	6 sec 612 ms

Avg GC Time ⓘ	54 ms	129 ms	380 ms	105 ms	140 ms	33 ms	6 ms	2 ms	67 ms
Avg Time std dev	48 ms	141 ms	8 ms	34 ms	0	5 ms	5 ms	4 ms	88 ms
Min/Max Time ⓘ	0 / 180 ms	0 / 341 ms	0 / 390 ms	0 / 140 ms	0 / 140 ms	0 / 40 ms	0 / 10 ms	0 / 10 ms	0 / 390 ms
Avg Interval Time ⓘ	248 ms	1 sec 221 ms	4 sec 177 ms	1 sec 197 ms	n/a	2 sec 56 ms	1 sec 221 ms	1 sec 221 ms	757 ms

🕒 G1 GC Time





Pause Time ?

Total Time	5 sec 180 ms
Avg Time	59 ms
Std Dev Time	77 ms
Min Time	0
Max Time	390 ms

Concurrent Time ?

Total Time	1 sec 292 ms
Avg Time	129 ms
Std Dev Time	141 ms
Min Time	12 ms
Max Time	341 ms

⚙️ Object Stats

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

Total created bytes ?	20.16 gb
Total promoted bytes ?	19.96 gb
Avg creation rate ?	1.59 gb/sec
Avg promotion rate ?	1.57 gb/sec

💧 Memory Leak ⓘ

No major memory leaks.

(**Note:** there are [8 flavours of OutOfMemoryErrors](#)

(<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/>))

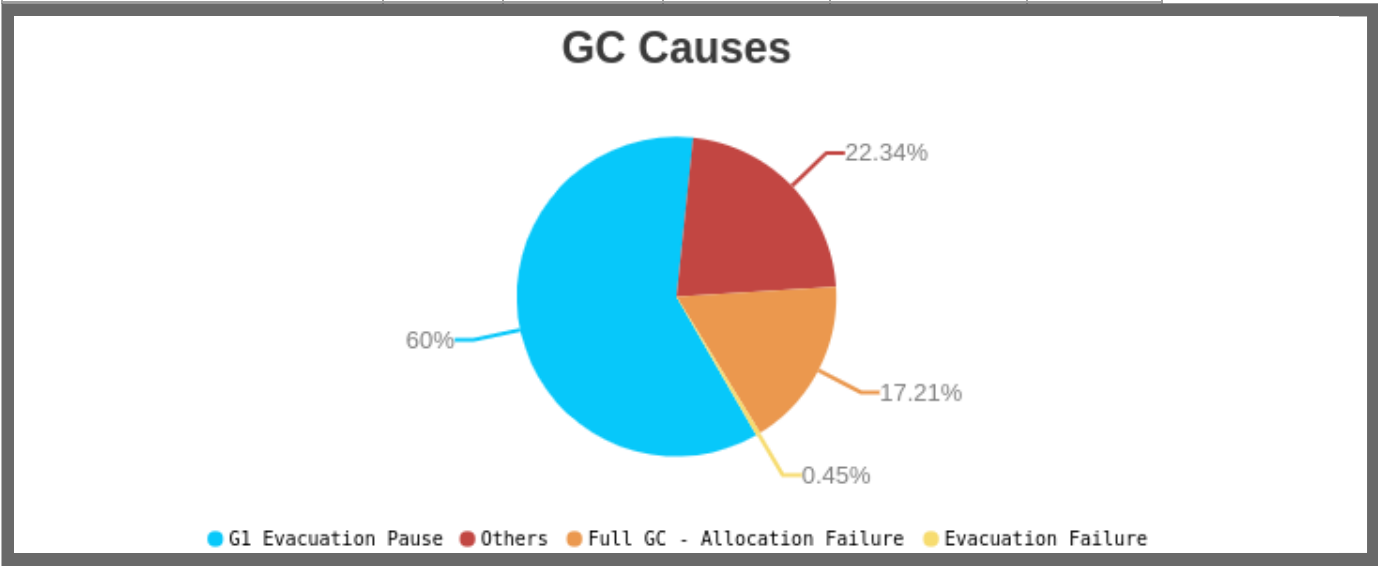
Not Reported in the log.

❓ GC Causes ⓘ

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

Cause	Count	Avg Time	Max Time	Total Time	Time %
G1 Evacuation Pause ⓘ	62	64 ms	179 ms	3 sec 967 ms	59.99%
Others	29	n/a	n/a	1 sec 477 ms	22.34%
Full GC - Allocation Failure ⓘ	3	379 ms	391 ms	1 sec 138 ms	17.21%

Evacuation Failure ⓘ	4	7 ms	15 ms	30 ms	0.45%
Total	98	n/a	n/a	6 sec 612 ms	99.99%



⌘ Tenuring Summary ⓘ

Not reported in the log.

📄 Command Line Flags ⓘ

-XX:G1HeapRegionSize=33554432 -XX:InitialHeapSize=260849920 -XX:MaxHeapSize=4173598720 -XX:+PrintGC -XX:+PrintGCDateStamps -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:+UseCompressedClassPointers -XX:+UseCompressedOops -XX:+UseG1GC

