

# GC Intelligence Report

 Obserwuj 2984

 2b3a8d4c-3d59-474e-b2a1-6030e4b7027d.log

 Duration: 23 hrs 11 min 36 sec

 [Download](#)

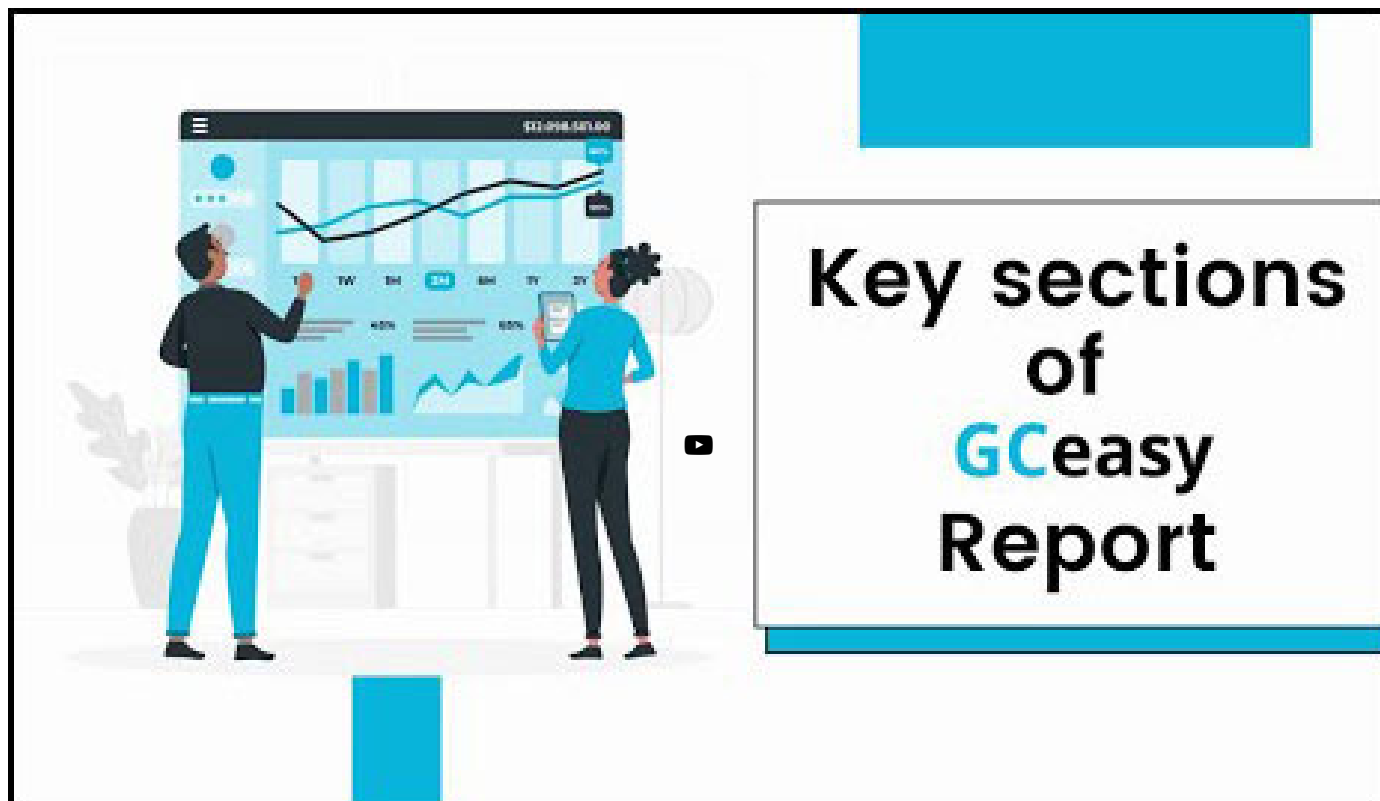
()

 [Share Report](#)

 [Expert Opinion \(Free service\)](#)

(developer-modal.jsp)

Learn key sections



(<https://www.youtube.com/watch?v=dN7S1RoKNYo>)



Congratulations! Your application's GC activity is healthy.

# Recommendations



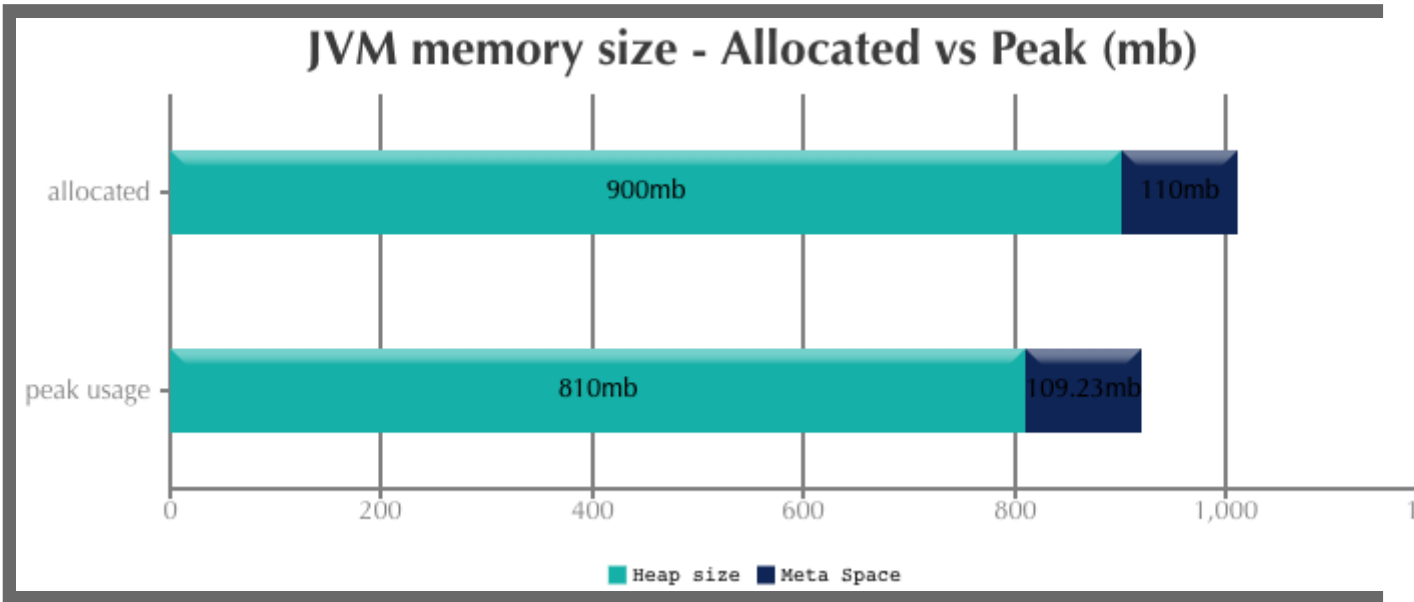
Our Machine Learning algorithms have identified memory optimization recommendations. To see those recommendations become our paid subscriber.

Select Plan ([pricing.jsp](#))

## JVM memory size

(To learn about JVM Memory, [click here](https://www.youtube.com/watch?v=uJLOICuOR4k) (<https://www.youtube.com/watch?v=uJLOICuOR4k>))

Region	Allocated ?	Peak ?
Heap	900 mb	810 mb
Metaspace	110 mb	109.23 mb
Total	1,010 mb	919.23 mb



# 🔑 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/>))

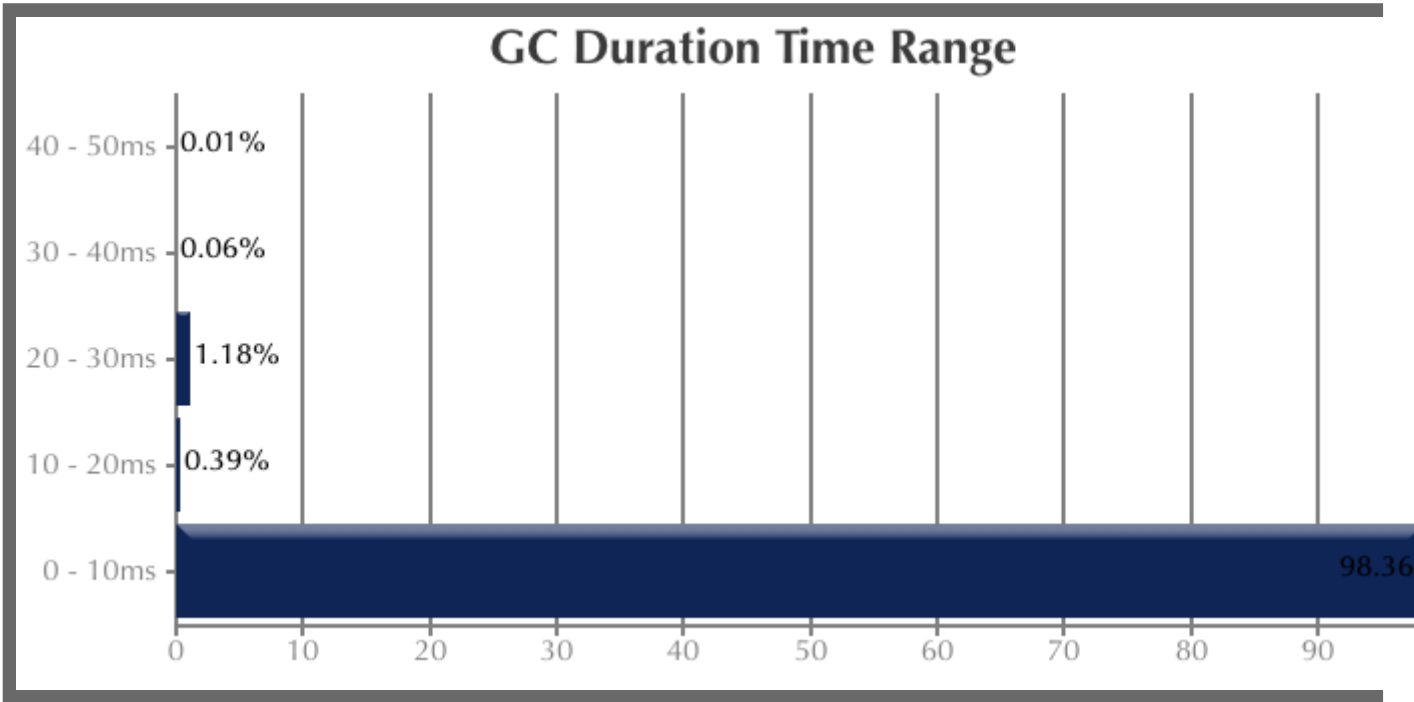
1 Throughput🔗 : 99.9%

2 Latency:

Avg Pause GC Time 🔗	10.1 ms
Max Pause GC Time 🔗	46.2 ms

GCPauseDuration Time Range 🔗:

Duration (ms)	No. of GCs	Percentage
10 ms <span>▼</span> <span>Change</span>		
0 - 10	8136	98.36%
10 - 20	32	0.39%
20 - 30	98	1.18%
30 - 40	5	0.06%
40 - 50	1	0.01%



# .||| Interactive Graphs

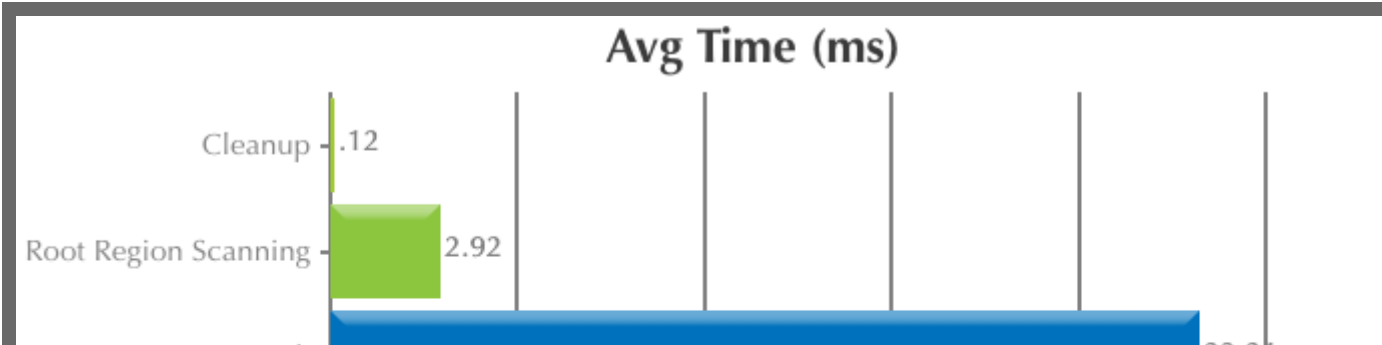
([How to zoom graphs?](#)) (<https://www.youtube.com/watch?v=JhZfJ6gJQyk>)

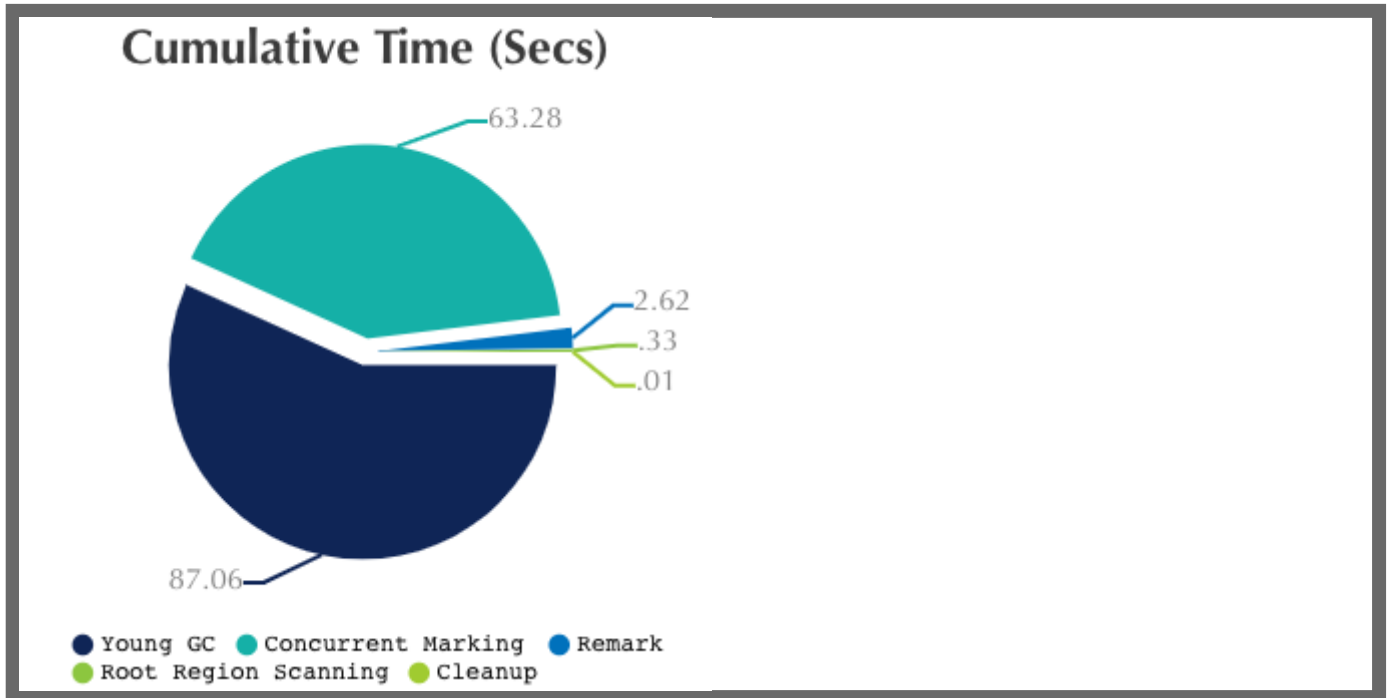
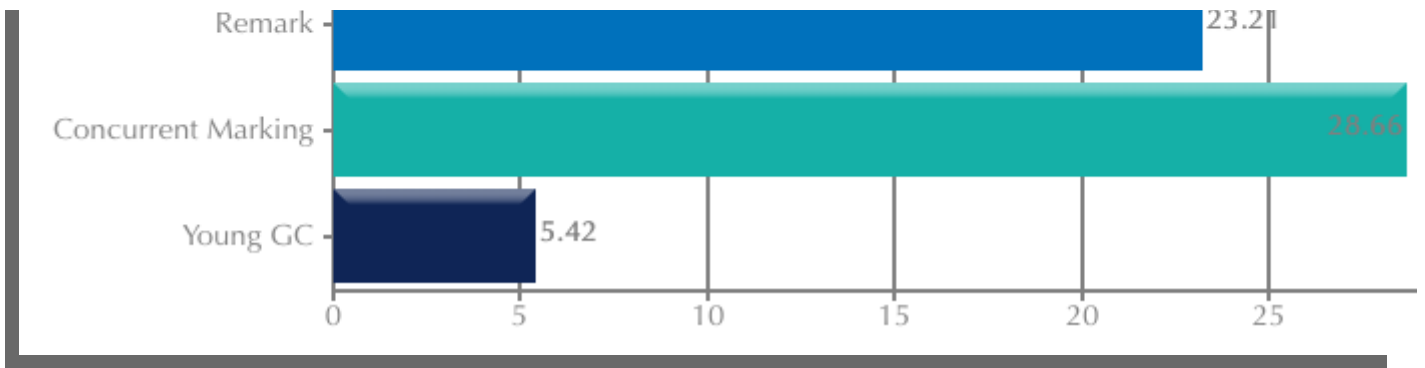
 **[Become Performance Expert! Training from GCEasy Architect!](#)**

(<https://ycrash.io/java-performance-training>)



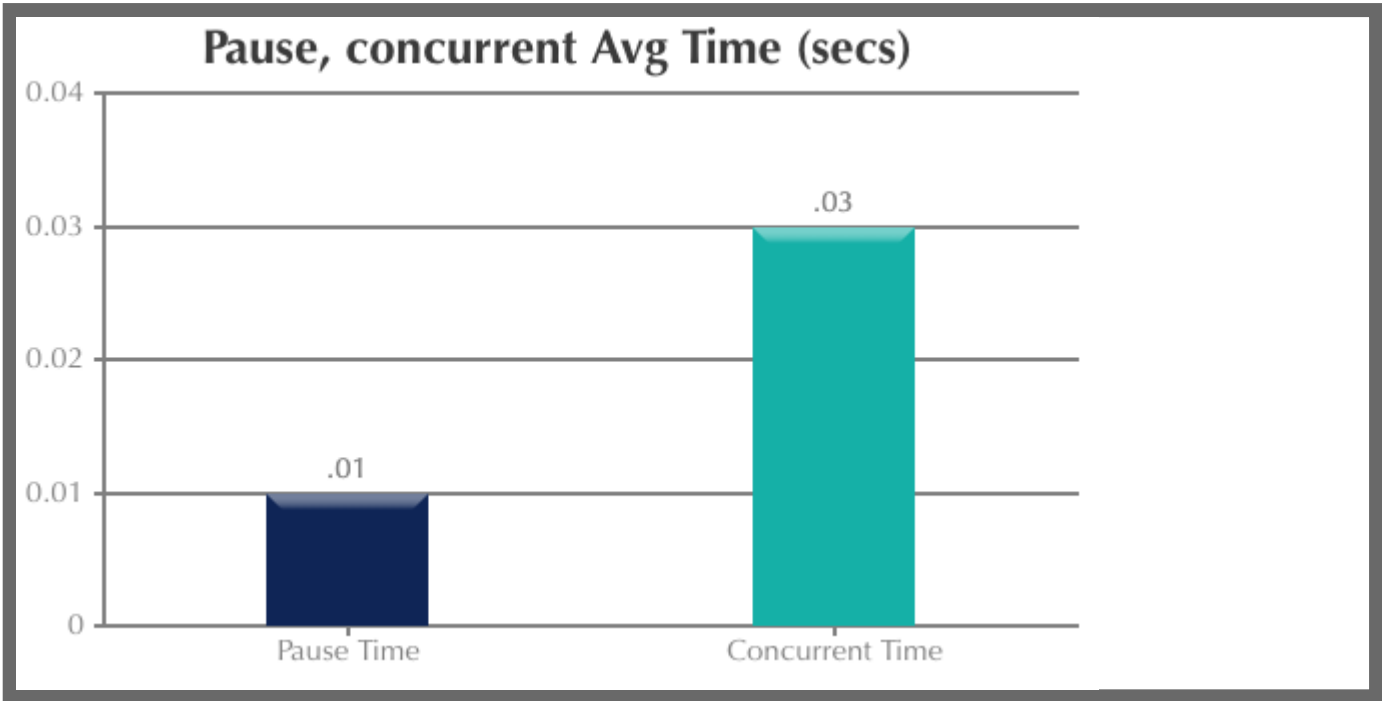
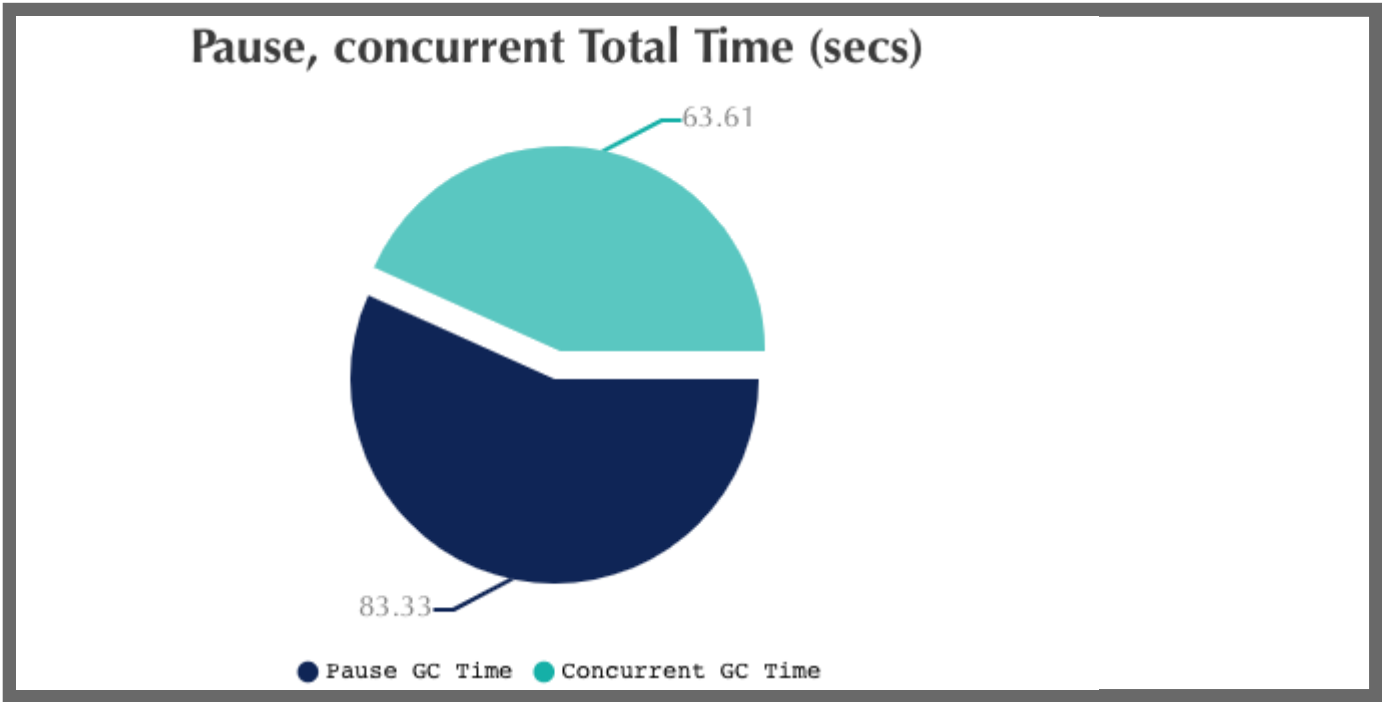
## G1 Collection Phases Statistics





	Young GC ⓘ	Concurrent Marking	Remark ⓘ	Root Region Scanning	Cleanup ⓘ
<b>Total Time ⓘ</b>	1 min 27 sec 58 ms	1 min 3 sec 281 ms	2 sec 623 ms	330 ms	13.4 ms
<b>Avg Time ⓘ</b>	5.42 ms	28.7 ms	23.2 ms	2.92 ms	0.119 ms
<b>Std Dev Time</b>	4.74 ms	111 ms	3.88 ms	0.627 ms	0.0327 ms
<b>Min Time ⓘ</b>	0	1.57 ms	19.6 ms	1.82 ms	0.0750 ms
<b>Max Time ⓘ</b>	20.0 ms	621 ms	46.2 ms	5.56 ms	0.245 ms
<b>Interval Time ⓘ</b>	5 sec 194 ms	37 sec 806 ms	11 min 44 sec 175 ms	11 min 44 sec 175 ms	11 min 44 sec 175 ms
<b>Count ⓘ</b>	16076	2208	113	113	113

# G1 GC Time



## Pause Time ?

Total Time	1 min 23 sec 326 ms
Avg Time	10.1 ms
Std Dev Time	2.05 ms

Min Time	0.0750 ms
Max Time	46.2 ms

Concurrent Time ?

Total Time	1 min 3 sec 613 ms
Avg Time	28.8 ms
Std Dev Time	111 ms
Min Time	1.57 ms
Max Time	626 ms

⚙️ Object Stats ?

Total created bytes ?	6.88 tb
Total promoted bytes ?	n/a
Avg creation rate ?	86.39 mb/sec
Avg promotion rate ?	n/a

📊 CPU Stats ? (To learn more about CPU stats, [click here](https://blog.gceasy.io/2022/08/05/garbage-collection-cpu-statistics/)  
(<https://blog.gceasy.io/2022/08/05/garbage-collection-cpu-statistics/>))

CPU Time: ?	2 min 36 sec 410 ms
User Time: ?	2 min 17 sec 130 ms
Sys Time: ?	19 sec 280 ms

# 💧 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](#) (./gc-recommendations/safe-point-solution.jsp))

	Total Time	Avg Time	% of total duration
Total time for which app threads were stopped	85.727 secs	0.006 secs	0.103 %
Time taken to stop app threads	1.07 secs	0.0 secs	0.001 %

# 🕒 Allocation stall metrics ?

(To learn more about Allocation Stall, [click here](#) (./gc-recommendations/allocation-stall-solution.jsp))

Not Reported in the log.






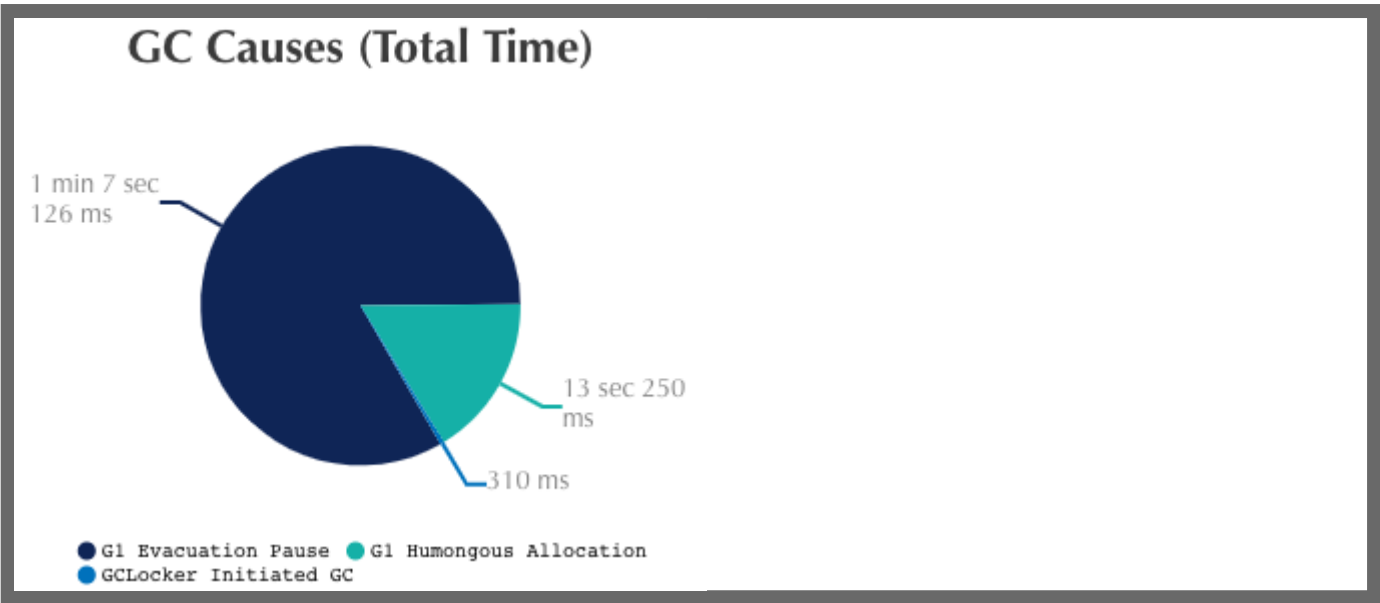
# String Deduplication Metrics

Not Reported in the log.

## GC Causes

(What events caused GCs & how much time they consumed?)

Cause	Count	Avg Time	Max Time	Total Time
G1 Evacuation Pause 	11627	5.77 ms	20.0 ms	1 min 7 sec 126 ms
G1 Humongous Allocation 	2196	6.03 ms	20.0 ms	13 sec 250 ms
GCLocker Initiated GC 	45	6.89 ms	10.0 ms	310 ms



## Tenuring Summary

Desired Survivor Size: ,

Max Threshold: 15

Age	Survival Count	Average size (kb)	Average Total 'To' size (kb)
age 1	13867	474.45	474.45

age 2	13861	106.21	580.71
age 3	13859	96.07	676.7
age 4	13859	90.95	767.68
age 5	13859	87.23	854.78
age 6	13859	84.56	939.37
age 7	13859	82.53	1021.9
age 8	13859	80.93	1102.76
age 9	13859	79.53	1182.24
age 10	13859	78.33	1260.57
age 11	13859	77.23	1337.82
age 12	13859	76.25	1414.04
age 13	13859	75.34	1489.33
age 14	13859	74.52	1563.74
age 15	13859	73.76	1637.58


## JVM Arguments

(To learn about JVM Arguments, [click here](https://blog.gceasy.io/2020/03/18/7-jvm-arguments-of-highly-effective-applications/) (https://blog.gceasy.io/2020/03/18/7-jvm-arguments-of-highly-effective-applications/))

Not reported in the log.

## Become a DevOps champion in your organization

(Best practises/tools)

- ✓ Use **fastThread.io** (<https://fastthread.io/>) tool to analyze thread dumps, core dumps and hs\_err\_pid files
- ✓ Use **HeapHero.io** (<https://heaphero.io/>) tool to analyze heap dumps
- ✓ Do proactive Garbage Collection analysis on all your JVMs (not just one or two) [using the GC log analysis API](https://blog.gceasy.io/2016/06/18/garbage-collection-log-analysis-api/)  (<https://blog.gceasy.io/2016/06/18/garbage-collection-log-analysis-api/>)
- ✓ Purchase 'Enterprise' edition (<http://gceasy.io/pricing.jsp>) for 10x fast, unlimited, secure usage

# Do you like this report?



GCeasy is the industry's first online Garbage collection log analysis tool aided by Machine Learning.

It's used by thousands of enterprises globally to tune & troubleshoot complex memory & GC problems.

## Reach Us

📍 Dublin, CA, USA

☎ +1-415-578-1205

✉ team@tier1app.com (mailto:team@tier1app.com)

## Quick Links

- › Terms & Conditions (terms.jsp)
- › Privacy policy (gc-privacy.jsp)
- › **Thread** (sister product) (<https://fastthread.io/>)
- › **Hero** (sister product) (<https://heaphero.io/>)
- › **Y** (sister product) (<https://ycrash.io/>)

## Stay in Touch!

Follow us on our social networks!

**f** (<https://www.facebook.com/tier1app>) **t** (<https://twitter.com/tier1app>) **in**

(<https://www.linkedin.com/company/gceasy>) **▶** (<https://www.youtube.com/channel/UCM-yObJ7pBjEy1wJMq5bDdw>)

Made by [Tier1app \(http://tier1app.com\)](http://tier1app.com) with  + soul + intelligence