GC Intelligence Report

gc-default.log

Duration: 22 min 11 sec 662 ms

Download

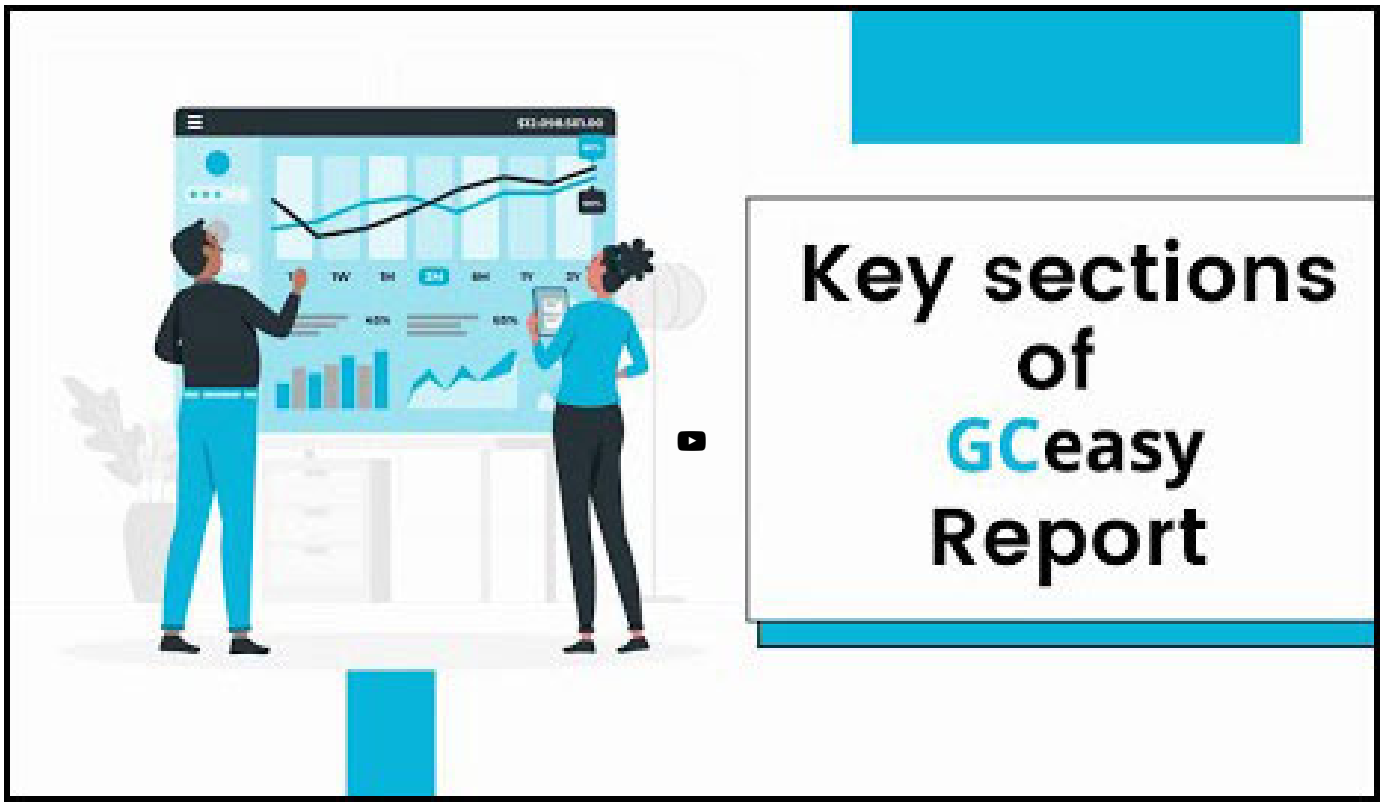
()

Share Report

Expert Opinion (Free service)

(developer-modal.jsp)

Learn key sections



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

Recommendations

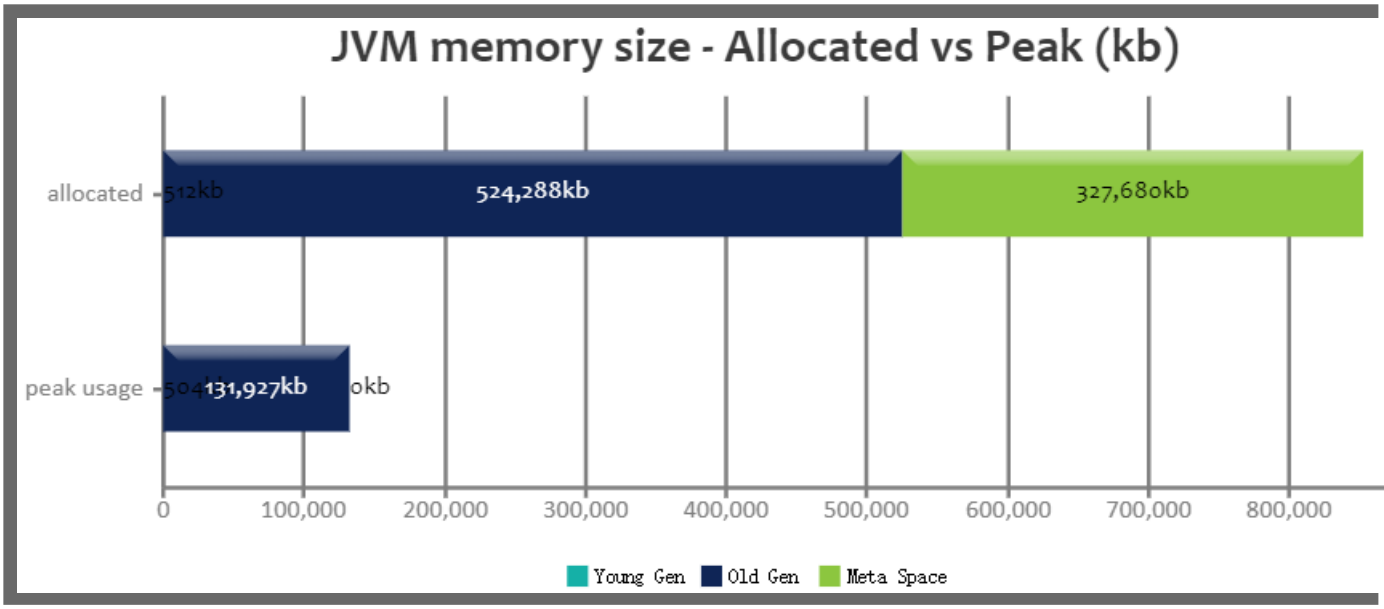
Our Machine Learning algorithms have identified memory optimization recommendations. To see those recommendations become our paid subscriber.

Select Plan (pricing.jsp)

JVM memory size

(To learn about JVM Memory, click here (https://www.youtube.com/watch?v=uJLOlCuOR4k))

Generation	Allocated	Peak
Young Generation	512 kb	504 kb
Old Generation	512 mb	128.83 mb
Meta Space	320 mb	n/a
Young + Old + Meta space	832 mb	129.17 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/))

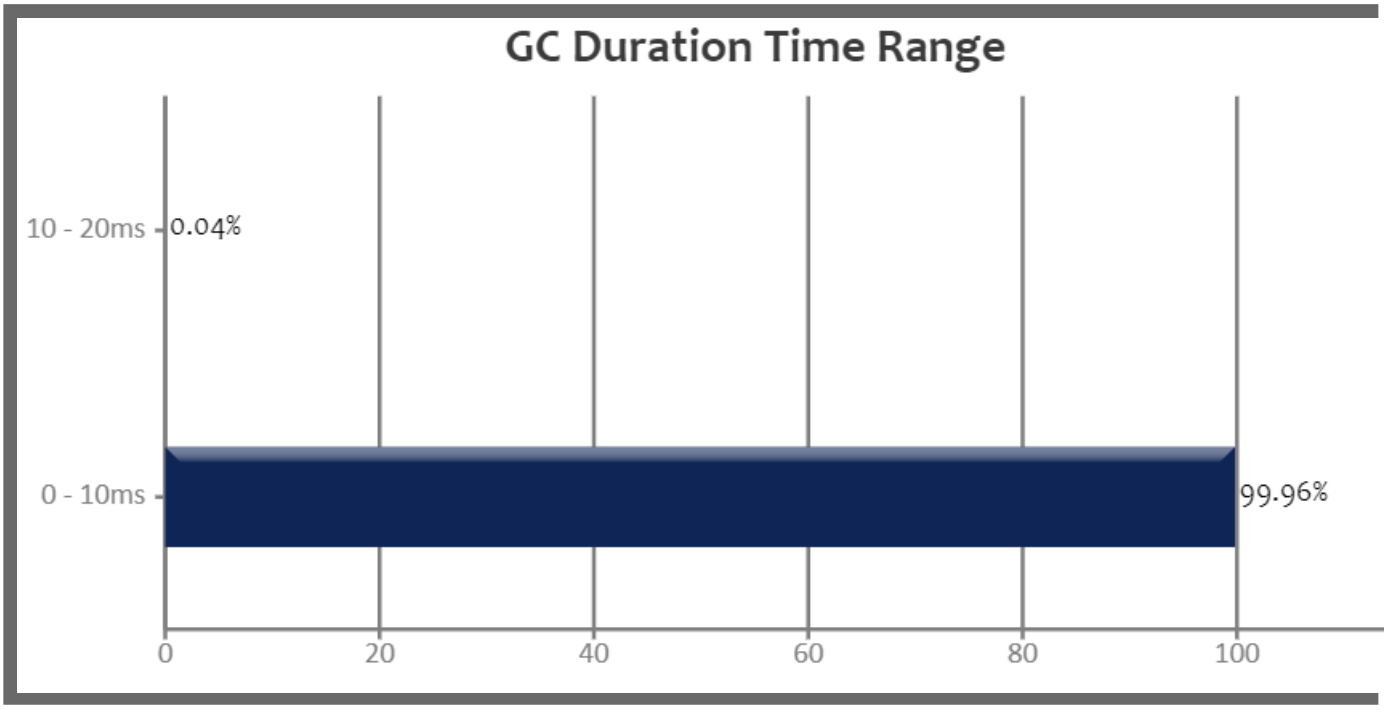
1 **Throughput : 99.386%**

2 **Latency:**

Avg Pause GC Time	**1.13 ms**
Max Pause GC Time	**20.0 ms**

GCPauseDuration Time Range :

Duration (ms) 10 ms Change	No. of GCs	Percentage
0 - 10	7225	99.96%
10 - 20	3	0.04%



Interactive Graphs

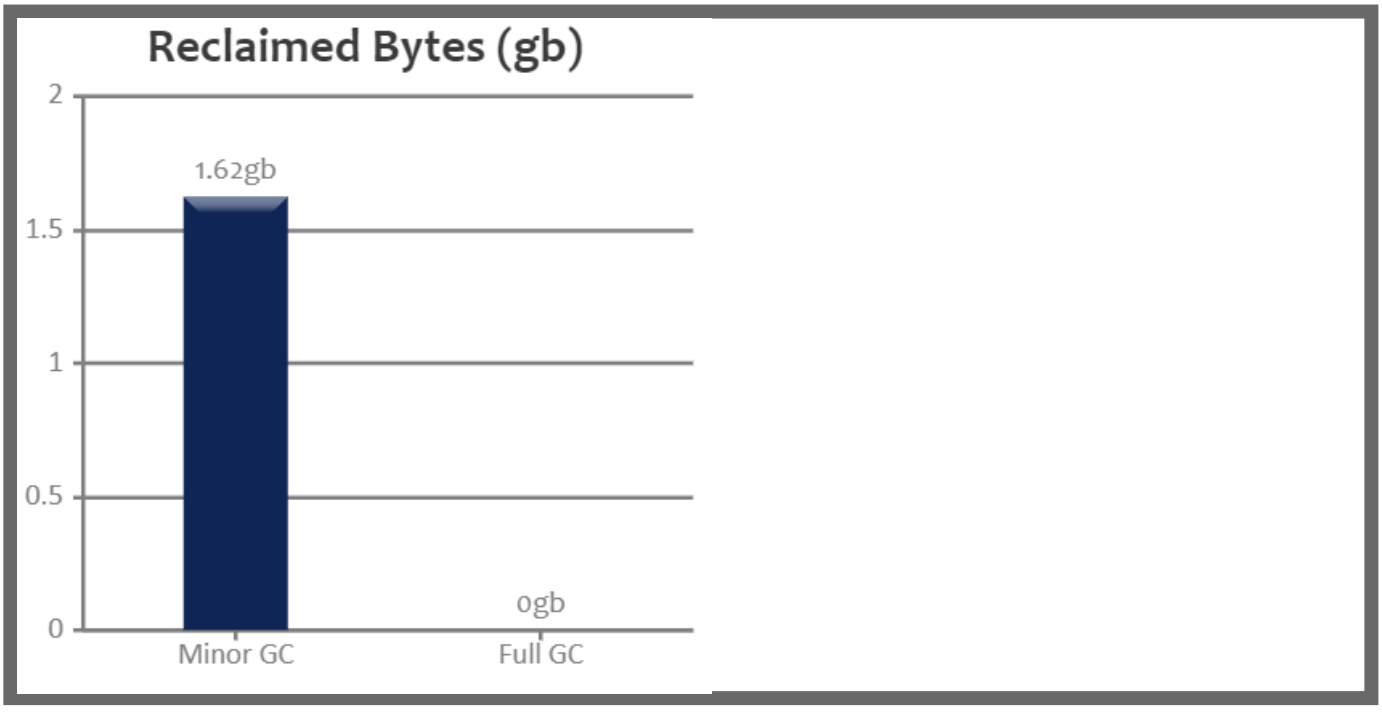
(How to zoom graphs?) (https://www.youtube.com/watch?v=JhZFj6gJQyk)

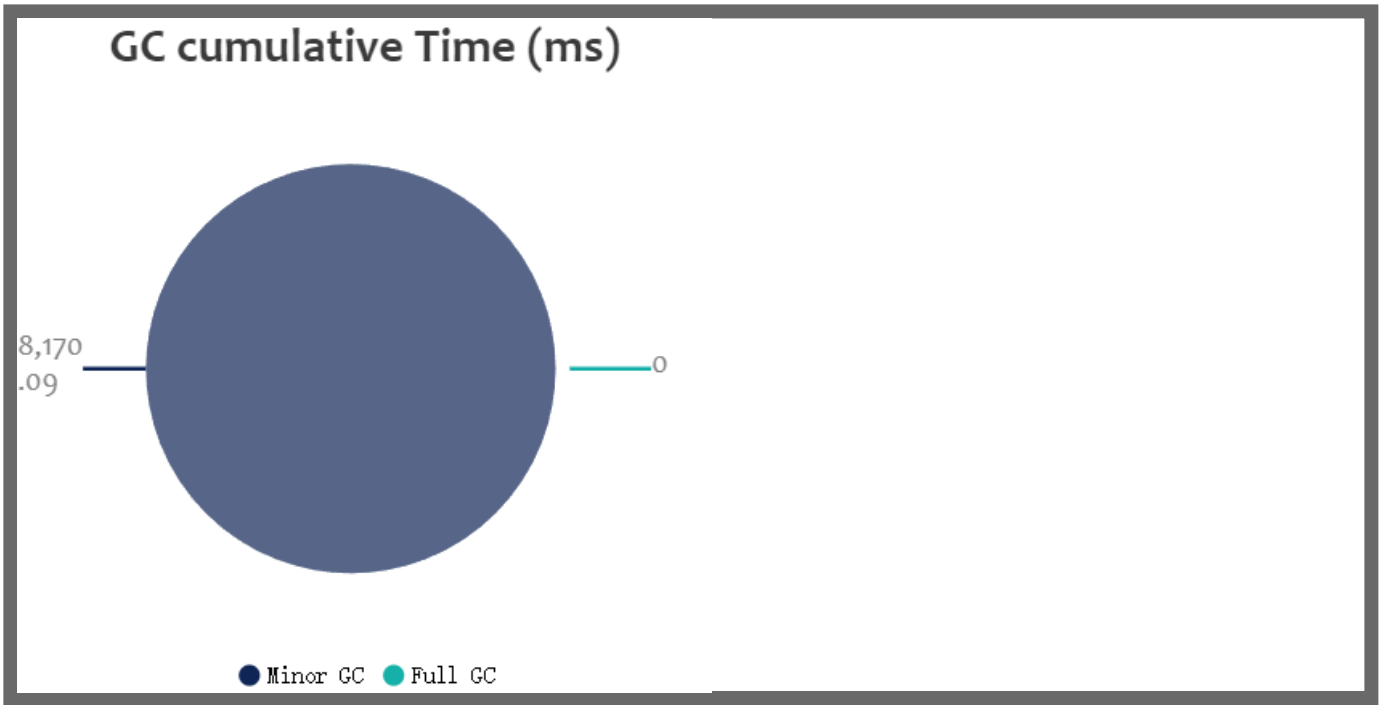
Become Performance Expert! Training from GCeasy Architect!

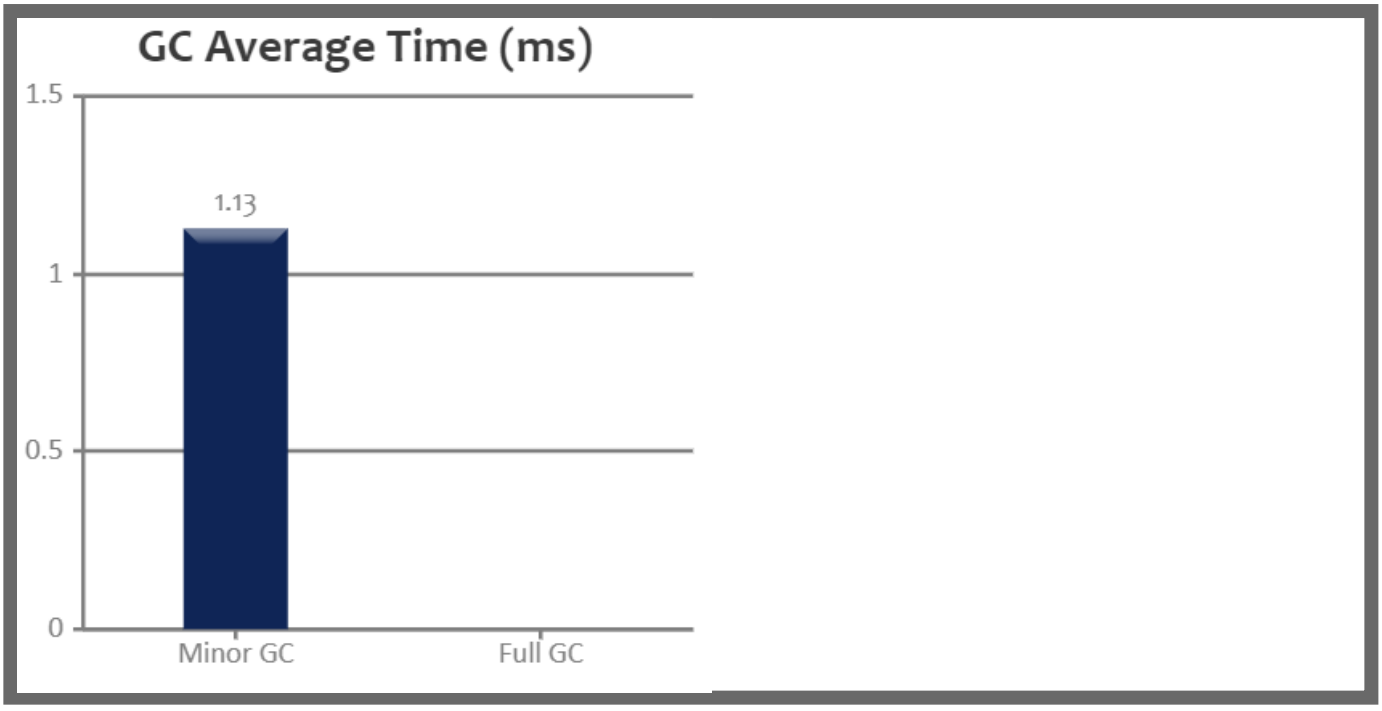
(https://ycrash.io/java-performance-training)

Choose Graph

GC Statistics







Total GC stats

Total GC count	7228
Total reclaimed bytes	n/a
Total GC time	8 sec 170 ms
Avg GC time	1.13 ms
GC avg time std dev	3.18 ms

GC min/max time	0 / 20.0 ms
GC Interval avg time	184 ms

Minor GC stats

Minor GC count	7228
Minor GC reclaimed	1.62 gb
Minor GC total time	8 sec 170 ms
Minor GC avg time	1.13 ms
Minor GC avg time std dev	3.18 ms

Minor GC min/max time	0 / 20.0 ms
Minor GC Interval avg	184 ms

Full GC stats

Full GC Count	0
Full GC reclaimed	n/a
Full GC total time	n/a
Full GC avg time	n/a
Full GC avg time std dev	n/a

Full GC min/max time	n/a
Full GC Interval avg	n/a

GC Pause Statistics

Pause Count	7228
Pause total time	8 sec 170 ms
Pause avg time	1.13 ms
Pause avg time std dev	0.0
Pause min/max time	0 / 20.0 ms

Object Stats

Total created bytes	1.75 gb
Total promoted bytes	118.16 mb
Avg creation rate	1.34 mb/sec
Avg promotion rate	90 kb/sec

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

CPU Time:	23 sec 620 ms
User Time:	23 sec 130 ms
Sys Time:	490 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))

Not Reported in the log.

Allocation stall metrics

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

Not Reported in the log.

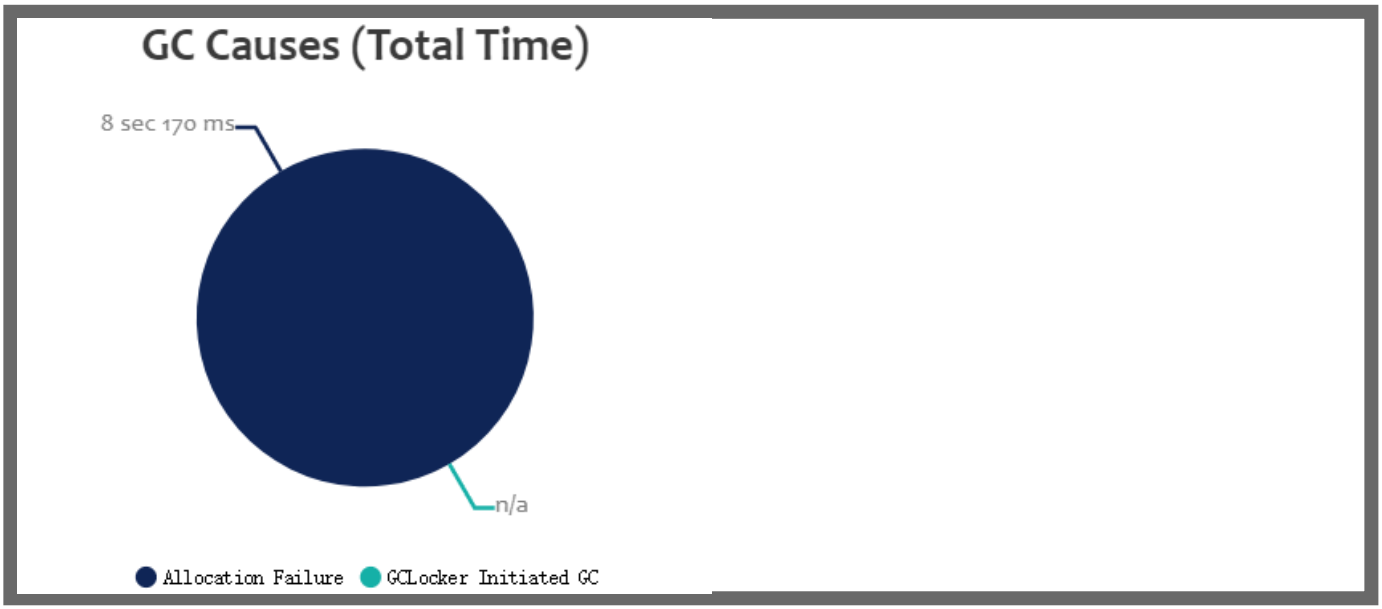
String Deduplication Metrics

Not Reported in the log.

GC Causes

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

Cause	Count	Avg Time	Max Time	Total Time
Allocation Failure	7211	1.13 ms	20.0 ms	8 sec 170 ms
GCLocker Initiated GC	17	n/a	n/a	n/a



Tenuring Summary

Not reported in the log.

JVM Arguments

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

-XX:+HeapDumpOnOutOfMemoryError

-XX:HeapDumpPath=/root/hero_web-1.0-SNAPSHOT-thread800-nio-port9001-micrometer/logs/java_heapdump.hprof

-XX:InitialHeapSize=536870912 -XX:MaxHeapSize=536870912 -XX:MaxMetaspaceSize=335544320

-XX:MaxNewSize=256 **-XX:MetaspaceSize=134217728** **-XX:NewSize=256** **-XX:-OmitStackTraceInFastThrow**

-XX:+PrintGC **-XX:+PrintGCDateStamps** **-XX:+PrintGCDetails** **-XX:+PrintGCTimeStamps**

-XX:+PrintHeapAtGC **-XX:-UseLargePages** **-XX:+UseParallelGC**

🏆 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/)
- ✔ Purchase 'Enterprise' edition (http://gceasy.io/pricing.jsp) for 10x fast, unlimited, secure usage

Do you like this report?

Quick Links

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

Stay in Touch!

Follow us on our social networks!

(https://www.facebook.com/tier1app) (https://twitter.com/tier1app) (https://www.linkedin.com/company/gceasy) (https://www.youtube.com/channel/UCM-yObJ7pBjEy1wJMq5bDdw)

Made by Tier1app (http://tier1app.com) with ♡ + soul + intelligence