Praudyogikee Lab – Garbage Collector Test

The garbage collection determines what memory is no longer used by an application and to recycle this memory for other uses. The memory is automatically reclaimed in the JVM. Garbage Collection frees the programmer from manually dealing with memory de-allocation.
This automatic memory management can be done manually as well, Calling System.gc() will manually activate the garbage collector
The garbage collector finds unused objects and deletes them, which can affect both the performance of an application and also the R.A.M. usage.

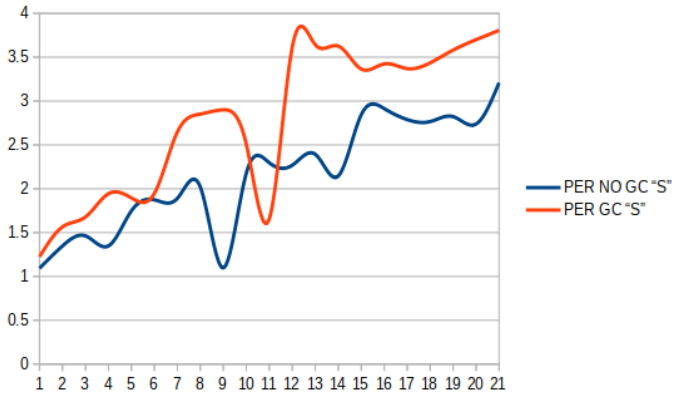
Date: 22/1/2021

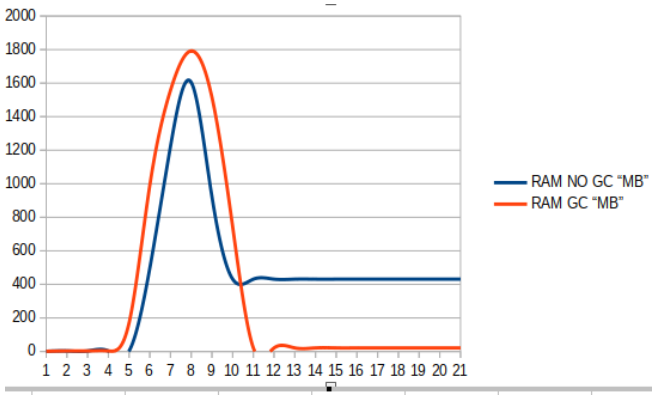
Test Description:
We made a java server which gets the GET requests “10k requests” and read the file and send it back. We made a test to see how much R.A.M. was consumed over time and how long did it take to receive a reply. The reply was a .PNG image at size of 6.296326 MB.

Data:

RAM NO GC “MB”	RAM GC “MB”	PER NO GC “S”	PER GC “S”
1	1	1.092789	1.225134
3	3	1.348015	1.565461
3	3	1.462514	1.678205
3	3	1.347305	1.943449
3	170	1.74842	1.894597
496	990	1.873948	1.946996
1223	1557	1.88996	2.647963
1605	1792	2.018249	2.851612
923	1516	1.097887	2.89985
431	744	2.178304	2.505391
431	21	2.296042	1.657154
431	21	2.267841	3.616145
431	21	2.393126	3.642046
431	21	2.144256	3.627104
431	21	2.842443	3.364645
431	21	2.913786	3.423619
431	21	2.788494	3.369239
431	21	2.765495	3.435095
431	21	2.822338	3.580642
431	21	2.737459	3.699454
431	21	3.208775	3.805704

Charts:





Conclusion:
Without the garbage collector, the application uses more R.A.M. , but it retrieved the file content "a.PNG" directly from the R.A.M. without making an IO operation, this highly affects the R.A.M as it took an average of 449.14 MB overall. While with the garbage collector it took an average of 333.8 MB overall. A 115.331 MB difference.
Looking at the performance, the performance of an application with the garbage collector is lower than the application without garbage collector.
It took an average of 2154 ms to retrieve 6.3 MB image without the garbage collector.
It took an average of 2779 ms to retrieve the same image with the garbage collector.
Therefore, a 115MB R.A.M. freed up costs 0.6 seconds of latency to be added.
Why did this happen?
Generally, Accessing the hard drive is slower than the R.A.M. the file content was stored in a byte array which has not been deleted by the garbage collector automatically. So with the next request the application didn't have to go through an IO operation. Instead, it cached the file inside the R.A.M. and then requested the content directly from the R.A.M.
This is important if you want to prioritize performance over computer resources or vice versa.
Note: ALL 10k requests were done at the same time, that's why you see a spike in the R.A.M. usage from 6 to 9 to serve all the threads.

Summary:

The garbage collector reduces resource-consuming but slows the application.