Take your project-related code or microservice or your ‘pet’ project (it should be memory intensive application) and make experiments described below

You mentor can assign 1 or 2 or 3 tasks for you (depends on your workload) but 3 is optimal

**1 Task**

Take your application and reduce typical -Xmx in 2x(maximum Heap Space for the VM). Try different GC algorithms described in presentation, enable/disable JIT(-Xint – interpreted mode only ), use new features like String Deduplication(-XX:+**UseStringDeduplication**). Compare the results with the GC log output. Run it on Java 7 and 8 (fix minor releases too). After that tune Heap

|  |  |
| --- | --- |
| **JVM OPTION** | **MEANING** |
| -Xms | initial java heap size |
| -Xmx | maximum java heap size |
| -Xmn | the size of the heap for the young generation |

and PermGen (or play with Metaspace)

-XX:PermSize=64M -XX:MaxPermSize=128M

and try to utilize and uberize free memory. Try to reach next goals if possible

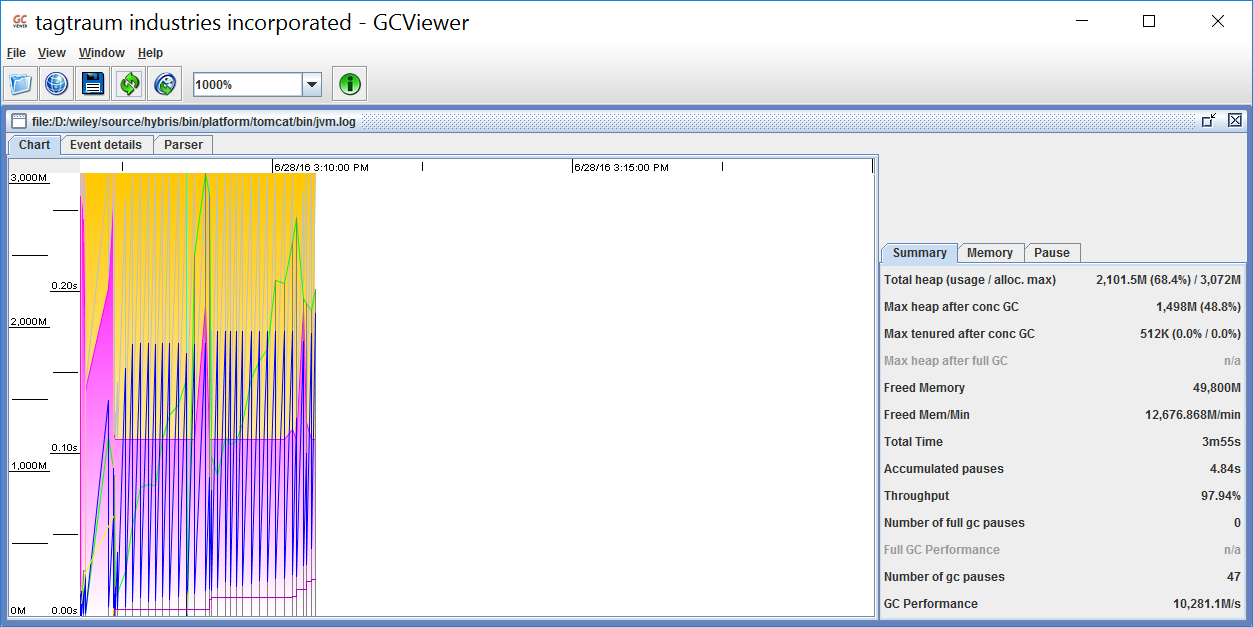
* High throughput in a single processor environment Serial Collector

-XX:+UseSerialGC JVM

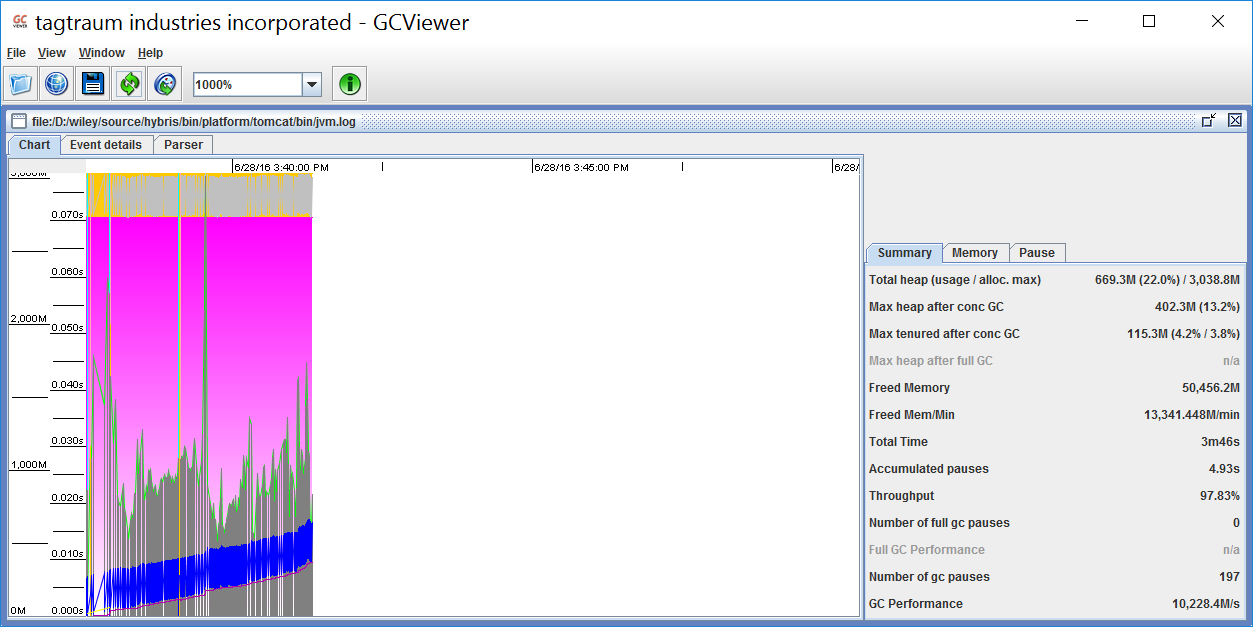
* High throughput in a multi-processor environment [Concurrent Mark Sweep (CMS)](http://www.oracle.com/technetwork/java/javase/memorymanagement-whitepaper-150215.pdf)
* High throughput and latency below one second in a multi-processor environment (Parallel GC)
* Lowest possible latency in a multi-processor environment [Concurrent Mark Sweep (CMS)](http://www.oracle.com/technetwork/java/javase/memorymanagement-whitepaper-150215.pdf) or G1 tuned

Make final performance report with charts (produce it with GC log parsers and viewers) and other column-row related stuff.

Parallel GC

**G1 Hybris Enterprise Application**  


CMS **Hybris Enterprise Application**

**2 Task**

Write your own StatelessClassLoader which will load only stateless classes (without fields) from .jar file specified as input parameter (<path-to-jar>). Don’t forget provide test code.

**3 Task**

Write simple Metaspace monitoring tool, test on your project

**Recomended links for reading**

[**https://plumbr.eu/java-garbage-collection-handbook**](https://plumbr.eu/java-garbage-collection-handbook)

[**http://openjdk.java.net/projects/code-tools/jol/**](http://openjdk.java.net/projects/code-tools/jol/)

[**http://shipilev.net/talks/devoxx-Nov2012-perfMethodology.pdf**](http://shipilev.net/talks/devoxx-Nov2012-perfMethodology.pdf)

[**http://shipilev.net/talks/devoxx-Nov2012-perfMethodology-mindmap.pdf**](http://shipilev.net/talks/devoxx-Nov2012-perfMethodology-mindmap.pdf)