JVM Performance Optimisation Training Summary

# Performance Problems

## CPU Spike

### Why

### How to Solve

## OutOfMemoryError

### Why

### How to Solve

## StackOverflowError

### Why

### How to Solve

# Useful JVM Arguments for Optimisation

## Heap

Use one of the followings:

|  |  |
| --- | --- |
| **JVM Arguments** | **Remarks** |
| -Xmx |  |
| -XMaxRAMFraction |  |
| -XMaxRAMPercentage |  |

## Metaspace

|  |  |
| --- | --- |
| **JVM Arguments** | **Remarks** |
| -Xx:MaxMetaspaceSize |  |

## Stack: -Xss

|  |  |
| --- | --- |
| **JVM Arguments** | **Remarks** |
| -Xss |  |

## GC selection

Use one of the followings:

|  |  |
| --- | --- |
| **JVM Arguments** | **Remarks** |
| -XX:+UseSerialGC |  |
| -XX:+UseParallelGC |  |
| -XX:+UseConcMarkSweepGC |  |
| -XX:+ UseG1GC |  |
| -XX:+ UseShenandoahGC |  |
| -XX:+ UseZGC |  |

## Timeouts

Use any of the followings if needed:

|  |  |
| --- | --- |
| **JVM Arguments** | **Remarks** |
| ‑Dsun.net.client.defaultConnectTimeout | Timeout to connect to host.  Example:  ‑Dsun.net.client.defaultConnectTimeout=2000 |
| -Dsun.net.client.defaultReadTimeout | Timeout when reading from input stream.  Example:  -Dsun.net.client.defaultReadTimeout=2000 |

# Useful JVM Arguments for Troubleshooting

## GC log:

|  |  |
| --- | --- |
| **JVM Arguments** | **Remarks** |
| ‑XX:+PrintGCDetails  ‑XX:+PrintGCDateStamps  -Xloggc:<file-path> | For Java 7 and below  Example: |
| -Xlog:gc\*:file=<file-path> | For Java 8 and above  Example: |

## Heap Dump

|  |  |
| --- | --- |
| **JVM Arguments** | **Remarks** |
|  |  |

## Thread Dump

|  |  |
| --- | --- |
| **JVM Arguments** | **Remarks** |
|  |  |

# Useful JVM Arguments as Reactive Actions

## When OutOfMemoryError

Use any of the followings:

|  |  |
| --- | --- |
| **JVM Arguments** | **Remarks** |
| -XX:+HeapDumpOnOutOfMemoryError  -XX:HeapDumpPath={heap-dump-file-path} |  |
| -XX:OnOutOfMemoryError=<script\_path> | Execute a script when OOME occurs.  Example:  -XX:OnOutOfMemoryError=/scripts/restart-myapp.sh |
| -XX:+CrashOnOutOfMemoryError |  |
| -XX:+ExitOnOutOfMemoryError |  |

# Recommended Practice for Optimum Performance

## Set Max Heap Size and Metaspace Size accordingly

## Use ZGC for Java 11+ Application

## Always Enable GC Logging

## Create Heap Dump on Out of Memory Error

## Increase Stack Memory Only When Needed

## Set timeout for connection

## Set Time Zone for Your Application

## Stop Wasting Memory in Your Code

1. Lazy Initialisation of your collection
2. Apply Goldilocks principle to your collection size
3. Stop String duplication

With G1GC (Java 8+), use XX:+UseStringDeduplication. With others, use ‘intern()’. However, beware of the performance impact with the latter. Thus, apply this on troublesome code that has been pointed out by your monitoring tool as a source of String duplication.

# Tools