
Java™
ORACLE®

Lesson 2

Dissecting Memory Problems

Poonam Parhar
JVM Sustaining Engineer
Oracle

Java Your Next (Cloud)

Agenda

1. Symptoms of Memory Problems
2. Causes of Memory Problems
3. OutOfMemoryError messages

Lesson 2-1

Symptoms of Memory Problems

Poonam Parhar
JVM Sustaining Engineer
Oracle

Java Your Next (Cloud)

Symptoms of Memory Problems

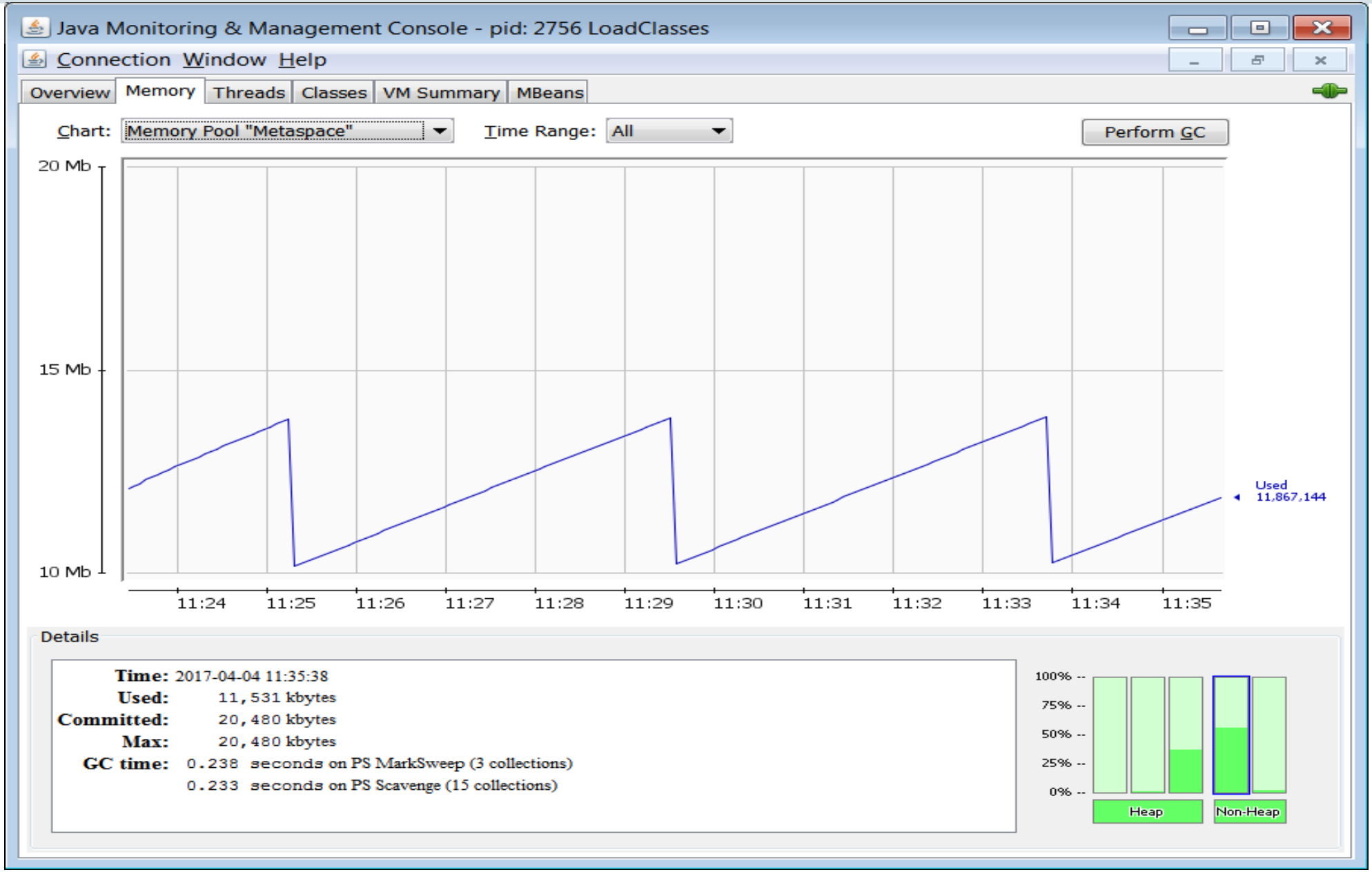
- Poor Performance of the Application
- Abnormal Memory Usage Growth
- OutOfMemoryError Messages

Poor Performance of the Application

- Application not performing to the expected level
- Long response times
- Client requests getting dropped
- Service unavailability
- Stuck threads

Abnormal Memory Usage Growth

- Continuous memory growth
- Memory usage spikes



OutOfMemoryError Messages

- Encountering OutOfMemoryError for various memory spaces

```
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
at java.util.Arrays.copyOfRange(Unknown Source)
at java.lang.String.<init>(Unknown Source)
at java.io.BufferedReader.readLine(Unknown Source)
at java.io.BufferedReader.readLine(Unknown Source)
at com.abc.ABCParser.dump(ABCParser.java:23)
at com.abc.ABCParser.mainABCParser.java:59)
```

Lesson 2-2

Causes of Memory Problems

Poonam Parhar
JVM Sustaining Engineer
Oracle

Java Your Next (Cloud)

Causes of Memory Problems

- Mis-configuration of Memory Spaces
- Memory Leaks
- Excessive use of Finalizers
- Explicit GC Invocations

Mis-configuration of Memory Spaces

```
20.343: [Full GC (Ergonomics) [PSYoungGen: 12799K->12799K(14848K)] [ParOldGen: 33905K-
  >33905K(34304K)] 46705K- >46705K(49152K), [Metaspace: 2921K->2921K(1056768K)],
  0.4595734 secs] [Times: user=1.17 sys=0.00, real=0.46 secs]
...... <snip> several Full GCs </snip> ......
22.640: [Full GC (Ergonomics) [PSYoungGen: 12799K->12799K(14848K)] [ParOldGen: 33911K-
  >33911K(34304K)] 46711K- >46711K(49152K), [Metaspace: 2921K->2921K(1056768K)],
  0.4648764 secs] [Times: user=1.11 sys=0.00, real=0.46 secs]
23.108: [Full GC (Ergonomics) [PSYoungGen: 12799K->12799K(14848K)] [ParOldGen: 33913K-
  >33913K(34304K)] 46713K- >46713K(49152K), [Metaspace: 2921K->2921 K(1056768K)],
  0.4380009 secs] [Times: user=1.05 sys=0.00, real=0.44 secs]
23.550: [Full GC (Ergonomics) [PSYoungGen: 12799K->12799K(14848K)] [ParOldGen: 33914K-
  >33914K(34304K)] 46714K- >46714K(49152K), [Metaspace: 2921K->2921 K(1056768K)],
  0.4767477 secs] [Times: user=1.15 sys=0.00, real=0.48 secs]
24.029: [Full GC (Ergonomics) [PSYoungGen: 12799K->12799K(14848K)] [ParOldGen: 33915K-
  >33915K(34304K)] 46715K- >46715K(49152K), [Metaspace: 2921K->2921 K(1056768K)],
  0.4191135 secs] [Times: user=1.12 sys=0.00, real=0.42 secs]
Exception in thread "main" java.lang.OutOfMemoryError: GC overhead limit exceeded at
  oom.main(oom.java:15)
```

Mis-configuration of Memory Spaces

- Memory spaces configured smaller than the live-set
- For example
  - Old generation sized smaller than the live-set of Java Objects
  - CodeCache sized smaller than the generated compiled code footprint
  - Young generation not size appropriately causing pre-mature promotion of objects
  - PermGen/Metaspace not sized appropriately causing Full GCs

Memory Leaks

- Unintentional retention of objects in the memory spaces
  - Unintentionally holding reference to set of objects in the Java Heap
  - Not de-referencing classloader instances appropriately
  - Not releasing native resources appropriately

Excessive use of Finalizers

- Objects with a finalizer may delay their garbage collection
- Finalizer thread needs to invoke the finalize() method of the instances before those instances can be reclaimed
- If the finalizer thread does not keep up with the rate at which the objects become eligible for finalization, the JVM might fail with an OutOfMemoryError
- Objects piled up in the Finalizer thread's queue would have been otherwise collected
- Pending finalization objects are essentially accumulated garbage
- Deprecated in Java 9

Explicit GC Invocations

- System.gc()
- Diagnostic Data Collections

Lesson 2-3

OutOfMemoryError

Poonam Parhar
JVM Sustaining Engineer
Oracle

Java Your Next (Cloud)

What is OutOfMemoryError?

- When the JVM runs out of space in various memory spaces
- Thrown when JVM can not proceed further with the process execution
- Unchecked exception subclassed from java.lang.VirtualMachineError

java.lang.Throwable
java.lang.Error
java.lang.VirtualMachineError
java.lang.OutOfMemoryError

Different Types of OutOfMemoryErrors

- Java Heap
- PermGen
- Metaspace
- Native Heap

What is in Java Heap?

- Java objects get allocated in the Java Heap

  String s = new String(“Hello Java”);

- The allocated String object resides in the Java Heap
  - first in the young generation
  - and if it lives long enough, then in the old generation

OutOfMemoryError: Java Heap

- This error means that the Java Heap is filled with Java Objects and the application is requesting to allocate more Java Objects
- The JVM has already invoked Full GC (garbage collection of the whole Java Heap) but could not free up any space
- It could be that the Java Heap is sized smaller than the application footprint or the application is unintentionally holding on to some set of objects in the heap

What is in PermGen?

- Classes and classes metadata get stored in the PermGen
- Static as well as dynamically loaded classes

OutOfMemoryError: PermGen Space

- This means that the PermGen is full with the loaded classes and their metadata and the application is requesting to load more classes
- The JVM has already invoked Full GC including the cleaning of PermGen but could not free up any space in the PermGen
- It could be that the PermGen is sized smaller than the footprint of classes and its metadata, or the application is unintentionally holding on to some set of classes in the PermGen

What is in Metaspace?

- Classes and classes metadata get stored in the Metaspace
- Static as well as dynamically loaded classes
- Metaspace is not part of the Java heap and is allocated out of the native memory

OutOfMemoryError: Metaspace

- This means that the Metaspace is full with the loaded classes and their metadata and the application is requesting to load more classes
- The JVM has already invoked Full GC including the cleaning of Metaspace but could not free up any space in the Metaspace
- It could be that the Metaspace is sized smaller than the footprint of classes and its metadata, or the application is unintentionally holding on to some set of classes in the Metaspace

OutOfMemoryError: Compressed class space

```
java.lang.OutOfMemoryError: Compressed class space
```

- If UseCompressedClassesPointers is enabled, then two separate areas of native memory are used for the classes and their metadata
- UseCompressedClassesPointers is enabled by default if UseCompressedOops is turned on
- UseCompressedClassesPointers, 64-bit class pointers are represented with 32-bit values, and these compressed class pointers are stored in the compressed class space
- By default, this compressed class space is sized at 1GB and can be configured using CompressedClassSpaceSize
- MaxMetaspaceSize sets an upper limit on the total committed size of both of these regions

What is in Native Heap?

- Java Thread Stacks
- CodeCache used to keep generated compiled code
- Loaded jar and zip files
- Loaded native libraries
- Native resources e.g. files
- Memory allocated from the native code e.g. JNI

OutOfMemoryError for Native Heap

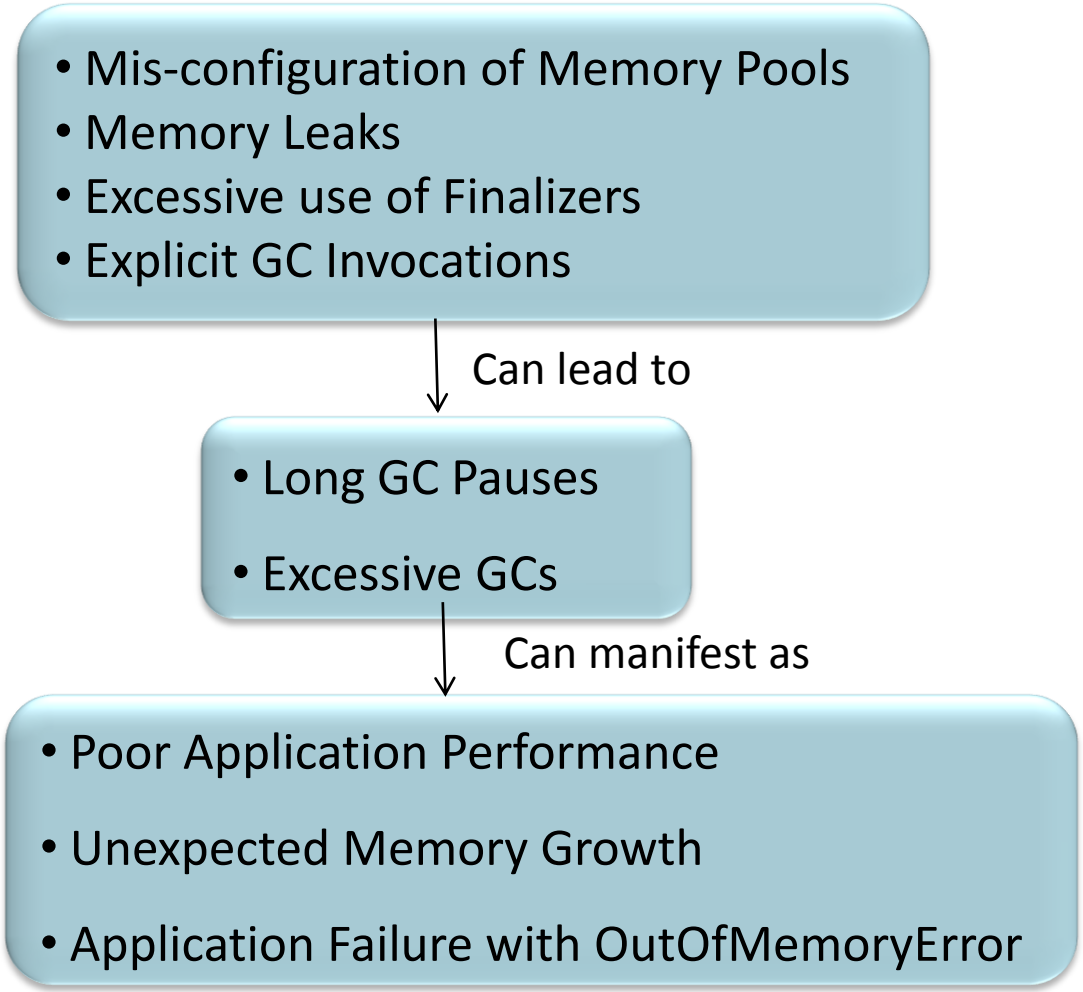
```
# A fatal error has been detected by the Java Runtime Environment:

#
# java.lang.OutOfMemoryError: requested 32756 bytes for ChunkPool::allocate. Out of swap
  space?
#
#  Internal Error (allocation.cpp:166), pid=2290, tid=27
#  Error: ChunkPool::allocate
```

```
# A fatal error has been detected by the Java Runtime Environment:
#
# java.lang.OutOfMemoryError : unable to create new native Thread
```

OutOfMemoryError for Native Heap

- JVM is not able to allocate from the native memory
- One or more processes are consuming all of the native memory
- Could mean a native memory leak

Summary




Java™
ORACLE®