OPERATIONS VIEW OF THE JAVA VM

Bill Schwanitz

Presentation available at
https://github.com/bilsch/columbus_devops_presentations

Rough overview

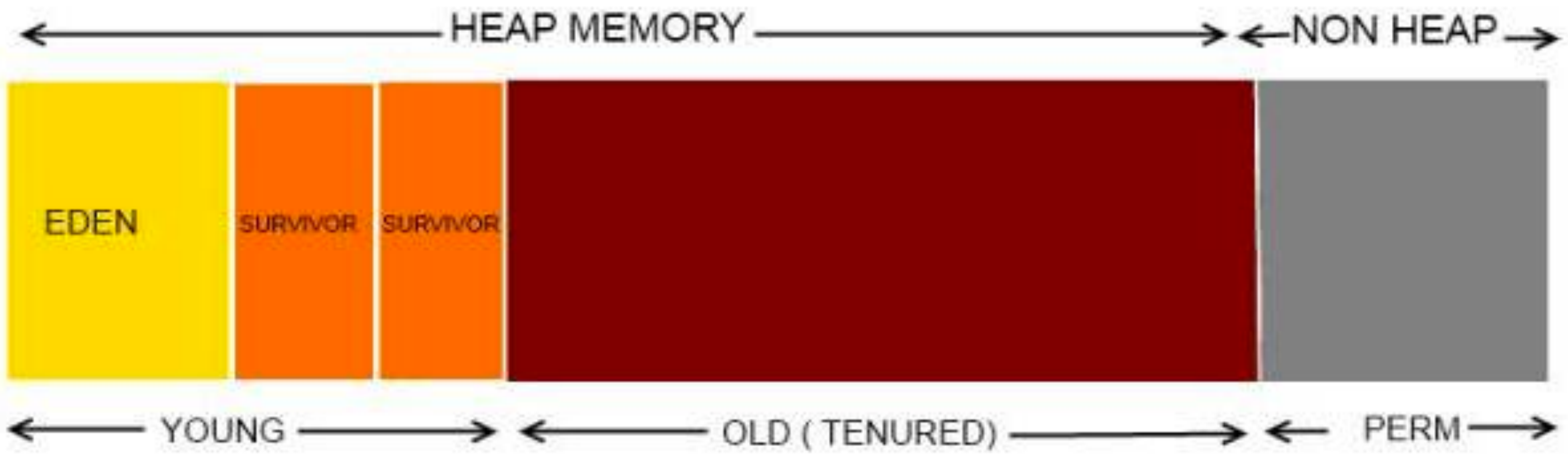
- Quick overview of java
- Crash course introduction to the Java heap
- Crash course on garbage collection (I’ll keep this very simple)
- Common simple tunable/run-time tweaks
- Helpful tools
- Common exceptions
    - OutOfMemory
    - How to read/decrypt an exception
- “Just give me more heap!”
    - Developers like to punt and just ask for more heap. I’ll give you compelling arguments to not give in too quickly

Quick overview of Java

- Object-oriented programming language developed by Sun Microsystems
  - First released in 1995
- Inspired/based on c and c++
- Original idea was to have a cross-platform just-in-time compiled language.
  - Write to the Java spec, it should run anywhere
  - The nasty stuff supposedly gets handled by the Java Virtual Machine
- Automatic memory management
  - At the expense of Garbage Collection

Crash course on Java Heap

- Broken out in to sections or generations
  - Eden, or New/Young generation
    - Further sub-divided to survivors
  - Old, or Tenured generation
  - Perm, or Permanent generation
    - Disappears in Java 8
- Generations have a default size
  - Based on runtime type, client or server and on JVM version



Crash course on Garbage collection

- non-referenced objects are marked for deletion
- Depending on java version, generation and vm type **may** be run in parallel
- Garbage collection event types
  - Minor GC – happens often and only in Eden
  - Major GC – happens very infrequently and in Eden, Old and Perm
- In Eden
  - Objects bounce between Survivor 0 and Survivor 1
  - After surviving for so many minor garbage collections, promotion to Old generation occurs
- In Old
  - Objects are usually only removed in major (aka stop the world) garbage collection events
- In Perm
  - Objects **never** removed unless a full/major gc occurs
  - Should never see memory shrink!

Common simple tunable/run-time tweaks

Java arg	Action	Value/example	When to change
-Xmx	Sets max heap size	2g	If you really need to!
-Xms	Sets min heap size	2g	Any time
-XX:MaxPermSize	Increase permgen size	256m	If you really need to
-XX: +PrintGCDetails	Turns up logging to stdout of garbage collector	n/a	Debugging mainly but can be used full time with Xloggc
-XX: +PrintGCTimeSTamps	Enhanced output of PrintGCDetails	n/a	^^
-Xloggc:<file>	Logs gc to alternate log file	/foo/bar	^^

http://www.oracle.com/technetwork/articles/java/vmoptions-jsp-140102.html

Helpful tools

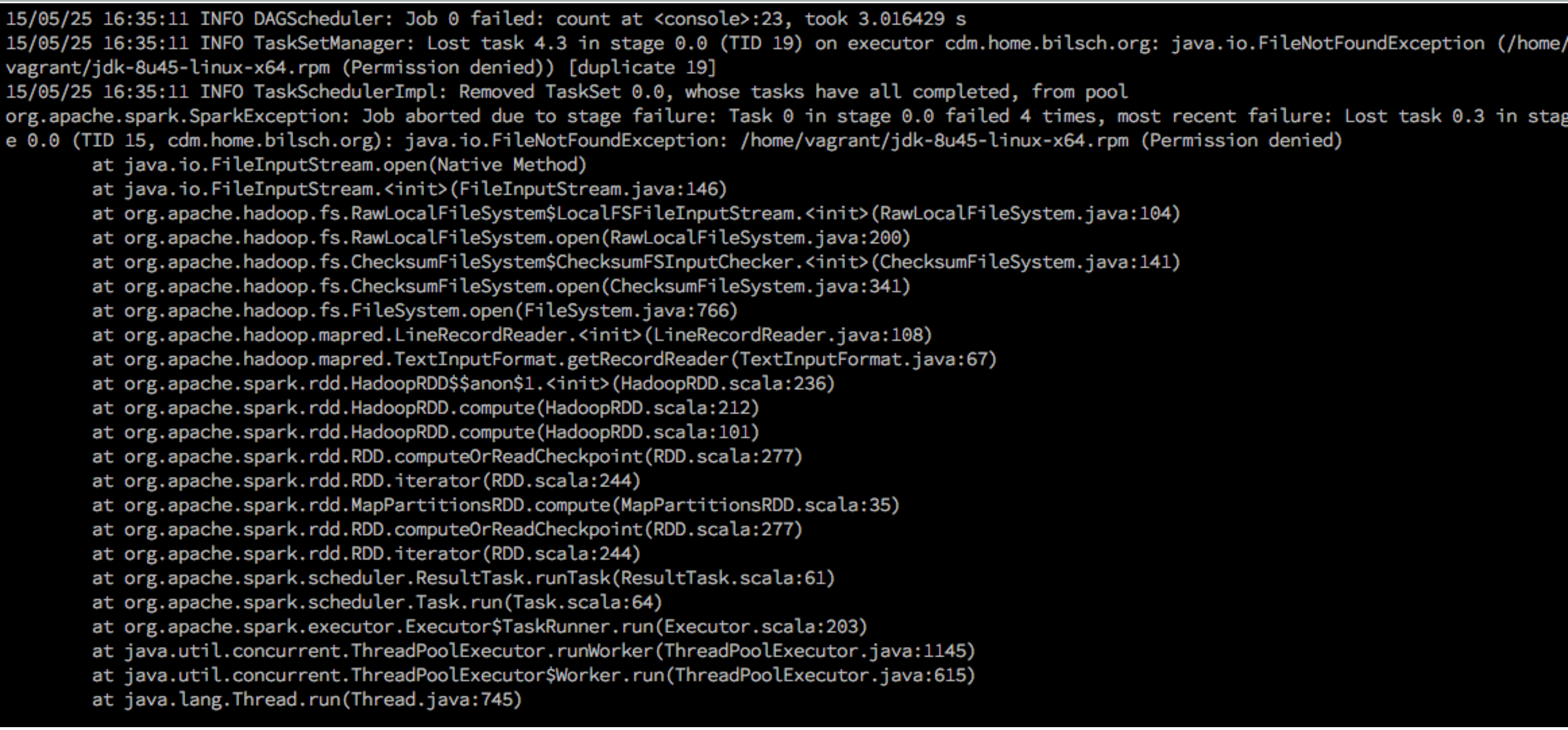
Tool Name	URL
VisualVM	https://visualvm.java.net/
Samurai	http://samuraism.jp/samurai/en/index.html
jmap	included in jdk
jstat	^^
jstack	^^
gcviz	https://github.com/Netflix/gcviz
GCViewer	https://github.com/chewiebug/GCViewer
MAT	https://eclipse.org/mat/

Common exceptions

- java.lang.OutOfMemoryError: Java heap space
- java.lang.OutOfMemoryError: PermGen space
- java.lang.OutOfMemoryError: GC overhead limit exceeded
- java.lang.NullPointerException

How to read an exception

- When a failure occurs in java, unless the developer catches it you get a nasty backtrace
- Presented from latest to oldest
  - Eg most relevant information is on top



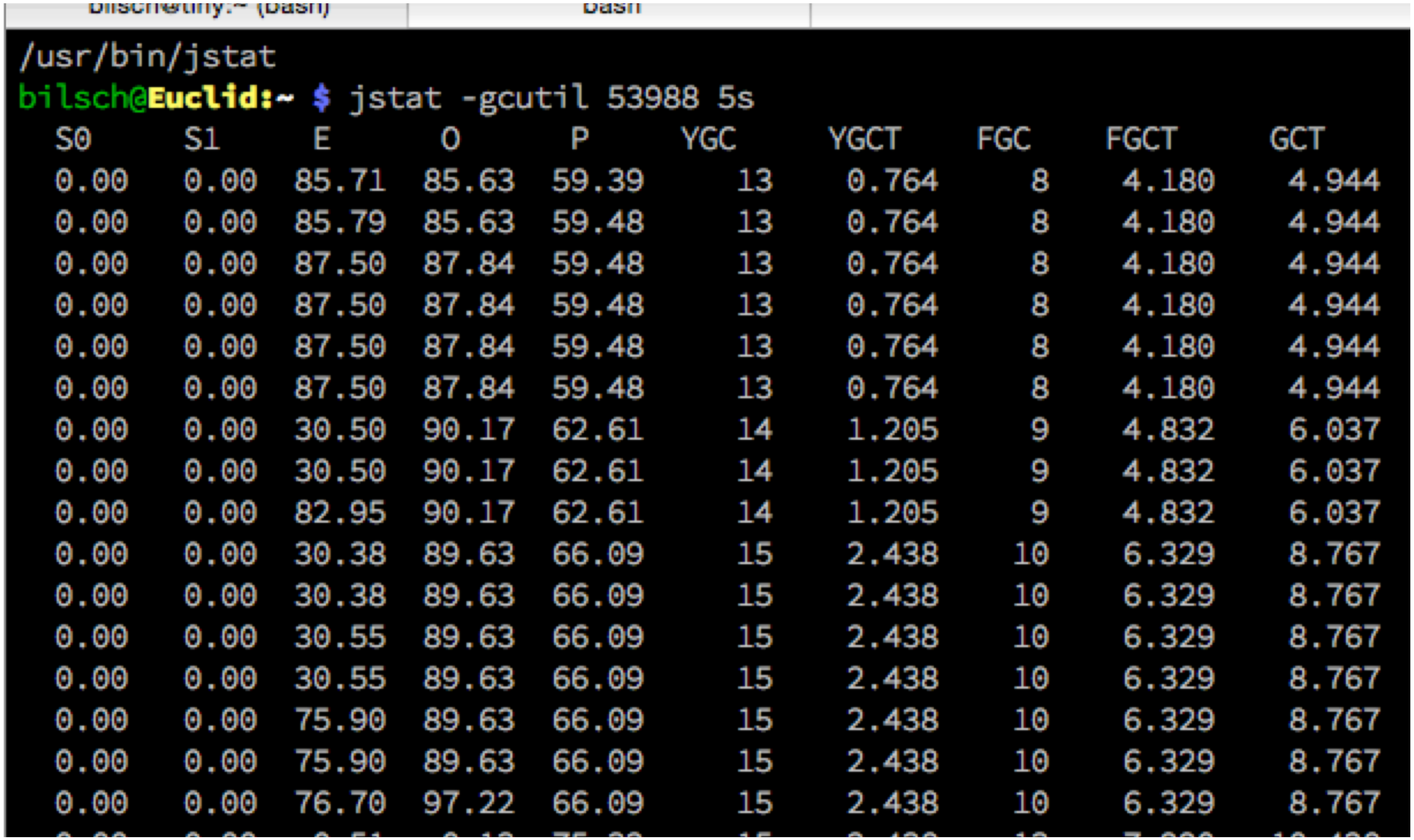
```
15/05/25 16:35:11 INFO DAGScheduler: Job 0 failed: count at <console>:23, took 3.016429 s
15/05/25 16:35:11 INFO TaskSetManager: Lost task 4.3 in stage 0.0 (TID 19) on executor cdm.home.bilsch.org: java.io.FileNotFoundException (/home/
vagrant/jdk-8u45-linux-x64.rpm (Permission denied)) [duplicate 19]
15/05/25 16:35:11 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from pool
org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage 0.0 failed 4 times, most recent failure: Lost task 0.3 in stag
e 0.0 (TID 15, cdm.home.bilsch.org): java.io.FileNotFoundException: /home/vagrant/jdk-8u45-linux-x64.rpm (Permission denied)
        at java.io.FileInputStream.open(Native Method)
        at java.io.FileInputStream.<init>(FileInputStream.java:146)
        at org.apache.hadoop.fs.RawLocalFileSystem$LocalFSFileInputStream.<init>(RawLocalFileSystem.java:104)
        at org.apache.hadoop.fs.RawLocalFileSystem.open(RawLocalFileSystem.java:200)
        at org.apache.hadoop.fs.ChecksumFileSystem$ChecksumFSInputChecker.<init>(ChecksumFileSystem.java:141)
        at org.apache.hadoop.fs.ChecksumFileSystem.open(ChecksumFileSystem.java:341)
        at org.apache.hadoop.fs.FileSystem.open(FileSystem.java:766)
        at org.apache.hadoop.mapred.LineRecordReader.<init>(LineRecordReader.java:108)
        at org.apache.hadoop.mapred.TextInputFormat.getRecordReader(TextInputFormat.java:67)
        at org.apache.spark.rdd.HadoopRDD$$anon$1.<init>(HadoopRDD.scala:236)
        at org.apache.spark.rdd.HadoopRDD.compute(HadoopRDD.scala:212)
        at org.apache.spark.rdd.HadoopRDD.compute(HadoopRDD.scala:101)
        at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:277)
        at org.apache.spark.rdd.RDD.iterator(RDD.scala:244)
        at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:35)
        at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:277)
        at org.apache.spark.rdd.RDD.iterator(RDD.scala:244)
        at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:61)
        at org.apache.spark.scheduler.Task.run(Task.scala:64)
        at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:203)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
        at java.lang.Thread.run(Thread.java:745)
```

- Simple example, permission denied trying to read a file
- Simple version of what is going on
    - Within a java thread in org.apache.spark
    - Scheduled task trying to iterate over the contents of a file
    - Bunch of hadoop classes
    - Top most is java.io.FileInputStream in the open method

Just give me more heap!

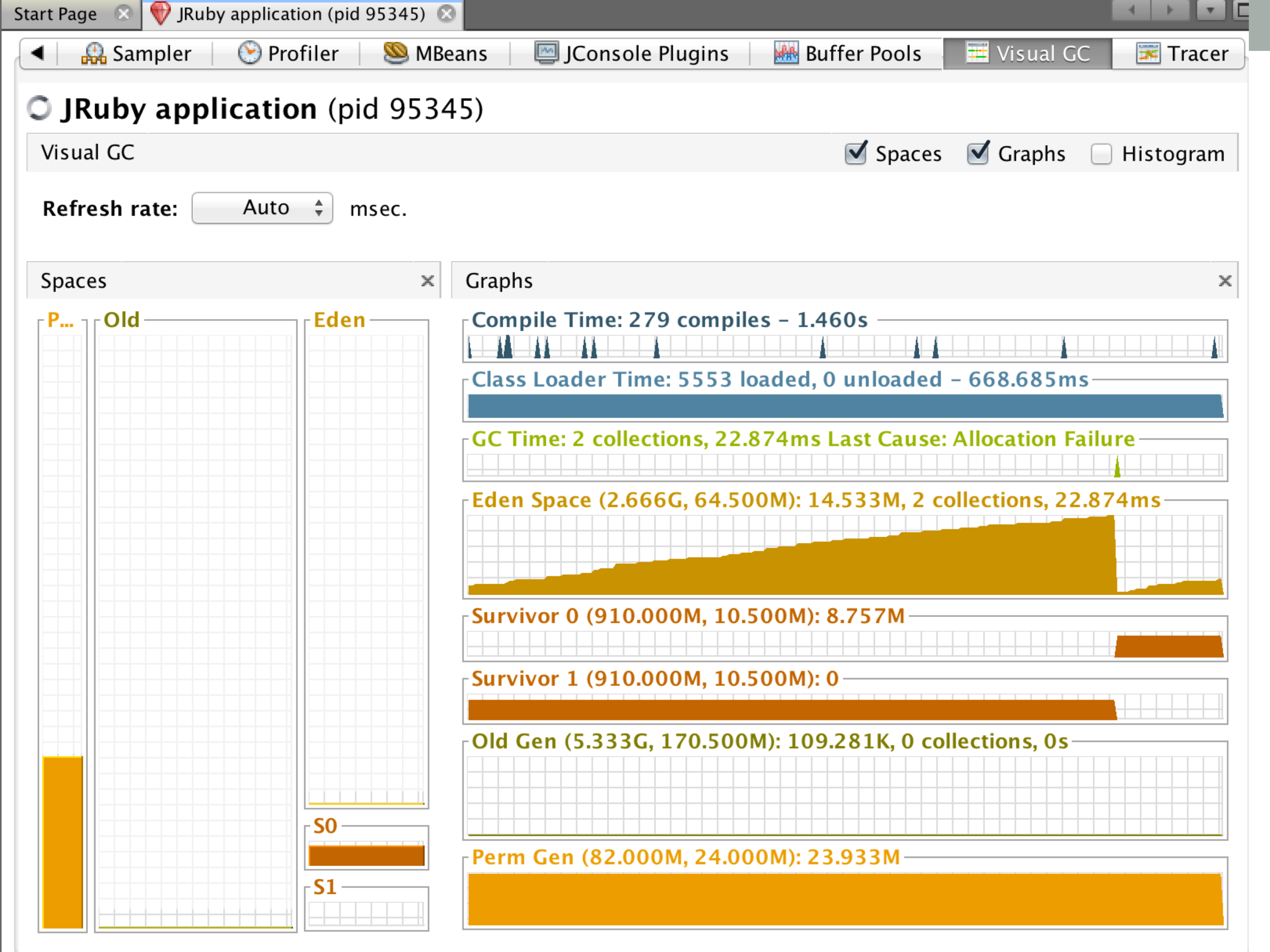
- I used to get this request a lot!
- Keep in mind garbage collection
  - Bigger heaps mean longer garbage collections
  - But could also really just be the right answer!
- Remember other applications need memory too
  - Especially the kernel caches/buffers!
- Having a bunch of JVMs on a single machine can be problematic when saturating host
  - Research garbage collector thread allocation!
  - http://architects.dzone.com/articles/how-tune-java-garbage

jstat



```
bilsch@tiny:~ (bash)          bash
/usr/bin/jstat
bilsch@Euclid:~ $ jstat -gcutil 53988 5s
  S0     S1     E      O      P     YGC     YGCT    FGC    FGCT     GCT
  0.00   0.00  85.71  85.63  59.39     13    0.764     8    4.180    4.944
  0.00   0.00  85.79  85.63  59.48     13    0.764     8    4.180    4.944
  0.00   0.00  87.50  87.84  59.48     13    0.764     8    4.180    4.944
  0.00   0.00  87.50  87.84  59.48     13    0.764     8    4.180    4.944
  0.00   0.00  87.50  87.84  59.48     13    0.764     8    4.180    4.944
  0.00   0.00  87.50  87.84  59.48     13    0.764     8    4.180    4.944
  0.00   0.00  30.50  90.17  62.61     14    1.205     9    4.832    6.037
  0.00   0.00  30.50  90.17  62.61     14    1.205     9    4.832    6.037
  0.00   0.00  82.95  90.17  62.61     14    1.205     9    4.832    6.037
  0.00   0.00  30.38  89.63  66.09     15    2.438    10    6.329    8.767
  0.00   0.00  30.38  89.63  66.09     15    2.438    10    6.329    8.767
  0.00   0.00  30.55  89.63  66.09     15    2.438    10    6.329    8.767
  0.00   0.00  30.55  89.63  66.09     15    2.438    10    6.329    8.767
  0.00   0.00  75.90  89.63  66.09     15    2.438    10    6.329    8.767
  0.00   0.00  75.90  89.63  66.09     15    2.438    10    6.329    8.767
  0.00   0.00  76.70  97.22  66.09     15    2.438    10    6.329    8.767
```

http://docs.oracle.com/javase/6/docs/technotes/tools/share/jstat.html#output_options



JRuby application (pid 95345)

Visual GC ✓ Spaces ✓ Graphs ☐ Histogram

Refresh rate: Auto msec.

Compile Time: 279 compiles – 1.460s

Class Loader Time: 5553 loaded, 0 unloaded – 668.685ms

GC Time: 2 collections, 22.874ms Last Cause: Allocation Failure

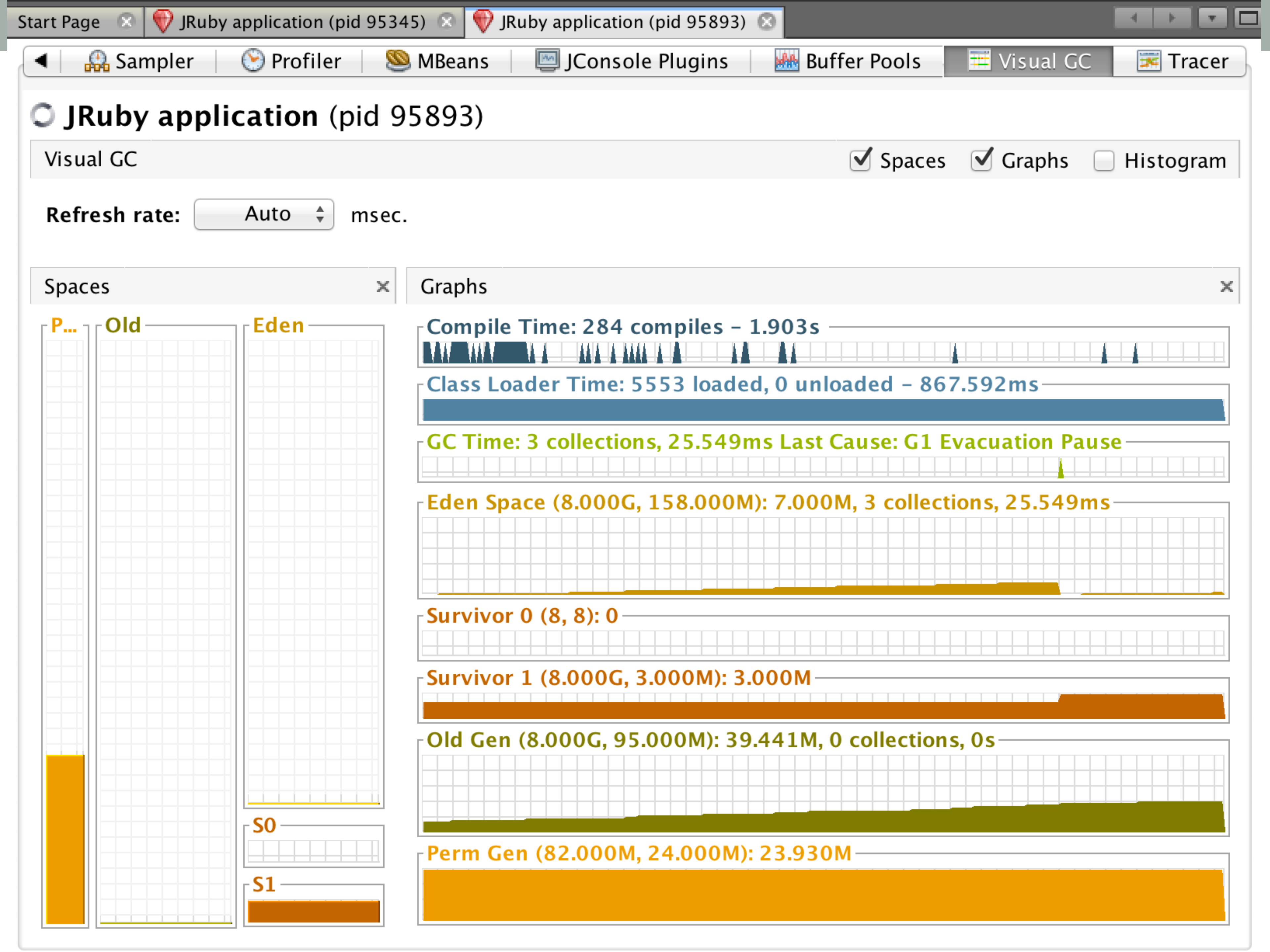
Eden Space (2.666G, 64.500M): 14.533M, 2 collections, 22.874ms

Survivor 0 (910.000M, 10.500M): 8.757M

Survivor 1 (910.000M, 10.500M): 0

Old Gen (5.333G, 170.500M): 109.281K, 0 collections, 0s

Perm Gen (82.000M, 24.000M): 23.933M



JRuby application (pid 95893)

Visual GC ✓ Spaces ✓ Graphs ☐ Histogram

Refresh rate: Auto msec.

Compile Time: 284 compiles - 1.903s

Class Loader Time: 5553 loaded, 0 unloaded - 867.592ms

GC Time: 3 collections, 25.549ms Last Cause: G1 Evacuation Pause

Eden Space (8.000G, 158.000M): 7.000M, 3 collections, 25.549ms

Survivor 0 (8, 8): 0

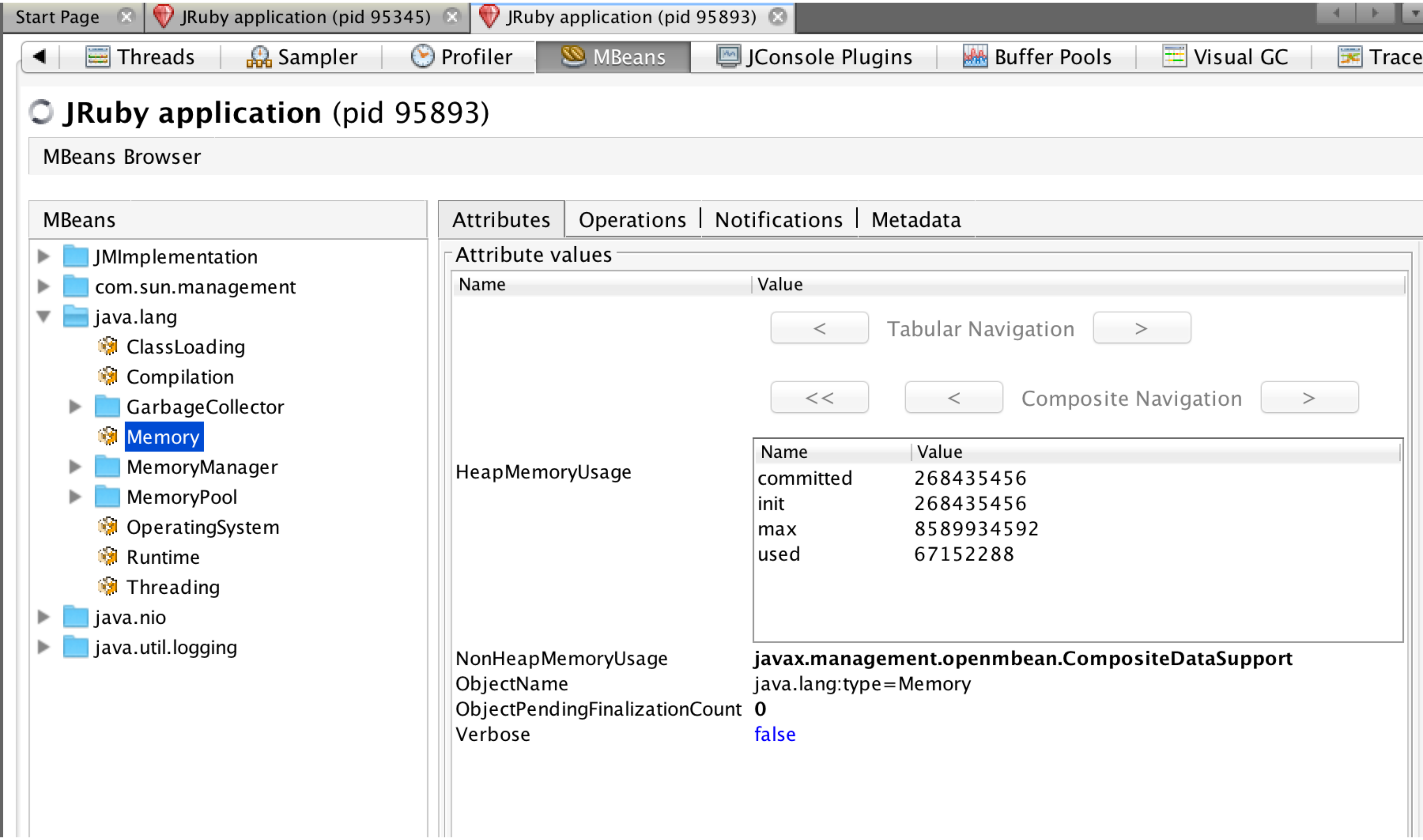
Survivor 1 (8.000G, 3.000M): 3.000M

Old Gen (8.000G, 95.000M): 39.441M, 0 collections, 0s

Perm Gen (82.000M, 24.000M): 23.930M

JMX

- Java Management Extensions
- Provides access to all kinds of neat stuff
  - Access to counters/gauges on individual MBeans
  - Ability to interrogate certain components of the MBean
    - Toggle runtime
    - Invoke methods
      - Trigger a garbage collection
      - Turn a jdbc pool on/off
      - Temporarily bump log levels
      - …

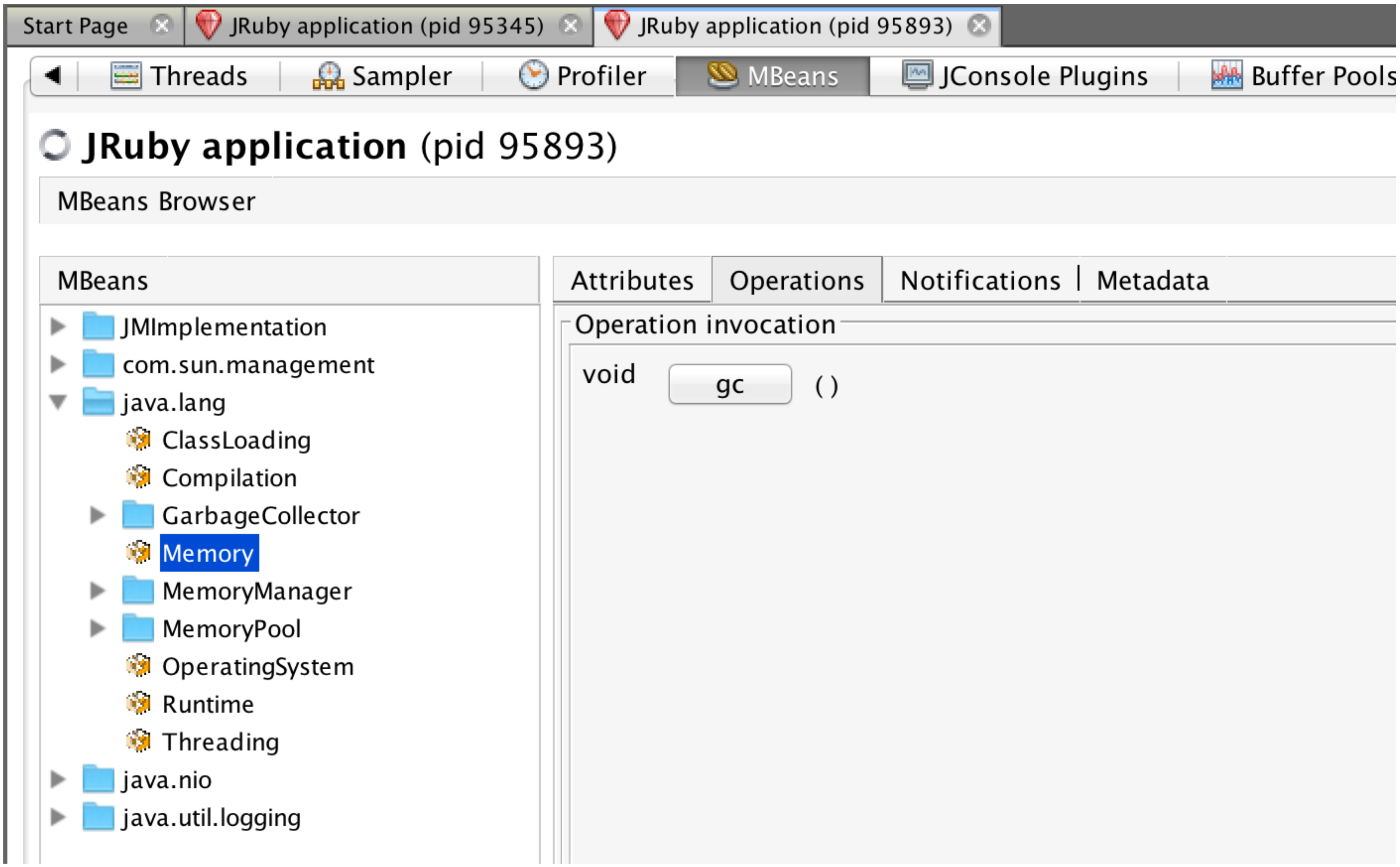


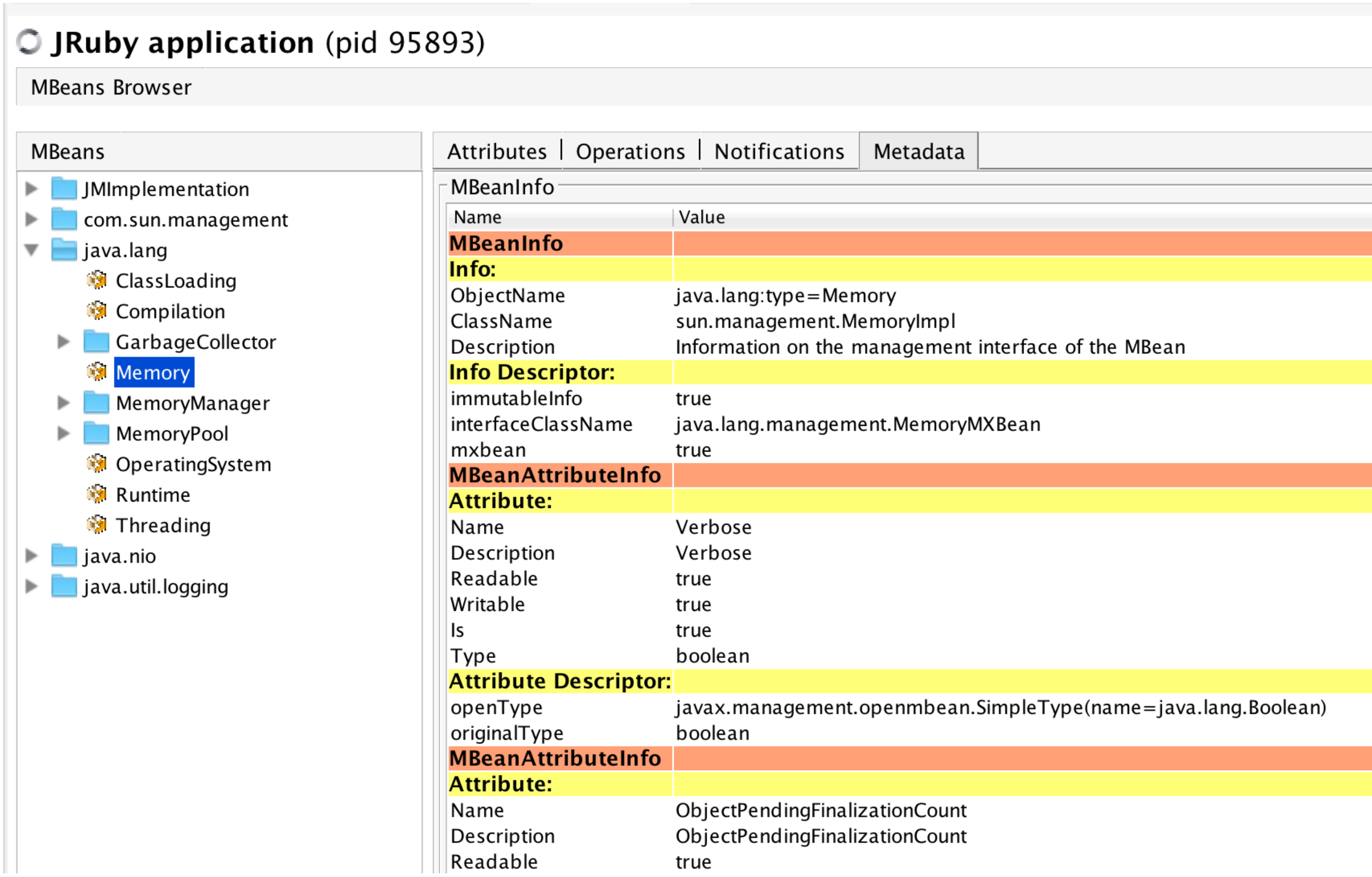
JRuby application (pid 95893)

MBeans Browser

Name	Value
HeapMemoryUsage	
NonHeapMemoryUsage	javax.management.openmbean.CompositeDataSupport
ObjectName	java.lang:type=Memory
ObjectPendingFinalizationCount	0
Verbose	false

Name	Value
committed	268435456
init	268435456
max	8589934592
used	67152288


Start Page
JRuby application (pid 95345)
JRuby application (pid 95893)
Threads
Sampler
Profiler
MBeans
JConsole Plugins
Buffer Pools
JRuby application (pid 95893)
MBeans Browser
MBeans
JMImplementation
com.sun.management
java.lang
ClassLoading
Compilation
GarbageCollector
Memory
MemoryManager
MemoryPool
OperatingSystem
Runtime
Threading
java.nio
java.util.logging
Attributes
Operations
Notifications
Metadata
Operation invocation
void
gc
( )



JRuby application (pid 95893)

MBeans Browser

Name	Value
MBeanInfo	
Info:	
ObjectName	java.lang:type=Memory
ClassName	sun.management.MemoryImpl
Description	Information on the management interface of the MBean
Info Descriptor:	
immutableInfo	true
interfaceClassName	java.lang.management.MemoryMXBean
mxbean	true
MBeanAttributeInfo	
Attribute:	
Name	Verbose
Description	Verbose
Readable	true
Writable	true
Is	true
Type	boolean
Attribute Descriptor:	
openType	javax.management.openmbean.SimpleType(name=java.lang.Boolean)
originalType	boolean
MBeanAttributeInfo	
Attribute:	
Name	ObjectPendingFinalizationCount
Description	ObjectPendingFinalizationCount
Readable	true

Final thoughts, monitoring

- Last thought on JMX. Collectd and others can pull jmx metric data
  - https://collectd.org/wiki/index.php/Plugin:GenericJMX
    - Embeds a jvm in to collectd
    - I found upstream builds do **not** compile jmx support in
  - https://exchange.nagios.org/directory/Plugins/Java-Applications-and-Servers/check_jmx/details
  - https://jolokia.org/index.html
    - Requires altering the class path and may or may not be liked by vendors
    - Much lighter weight than the check_jmx as you can get at things with curl vs. invoking a small jvm