What's in your JVM?

Carl Chesser
@che55er

Titanium Sponsors

Platinum Sponsors

Gold Sponsors


Cerner™
engineering.cerner.com
We're hiring!
Big Data
C#
Java
Python
Ruby
Web
& more!

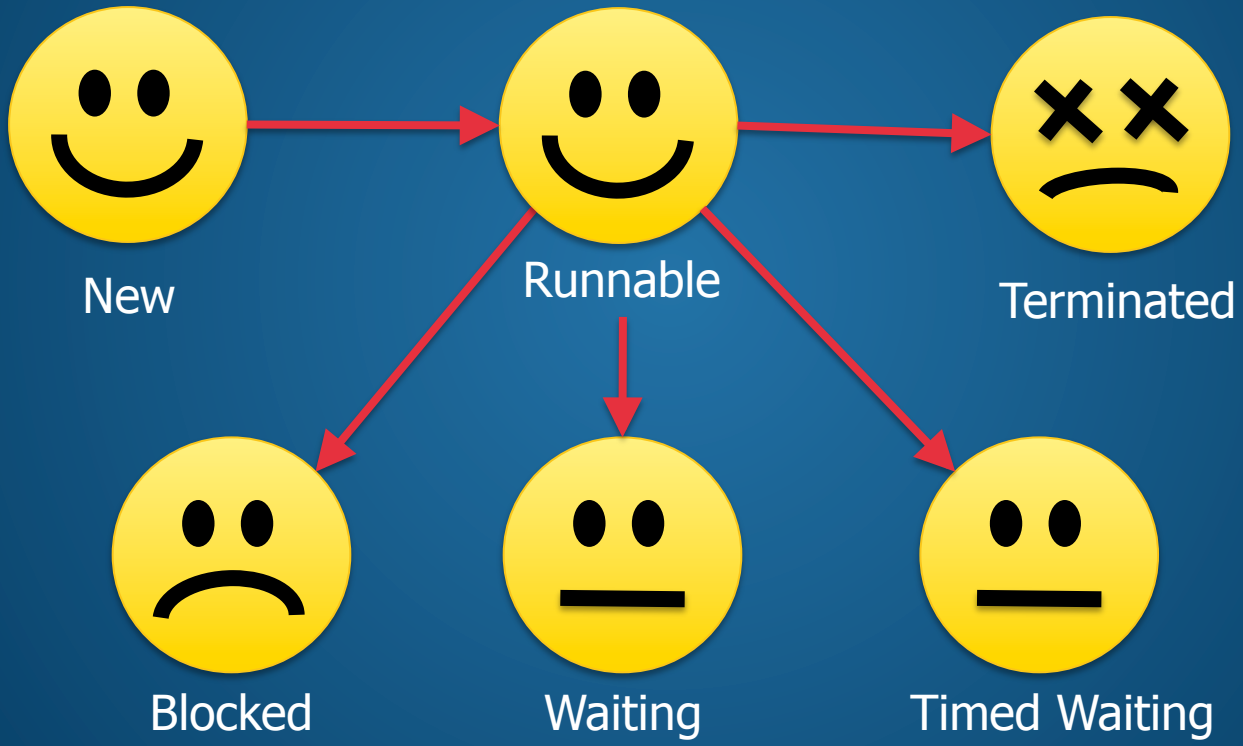
“In God we trust, all others bring data.”

- William Edwards Deming

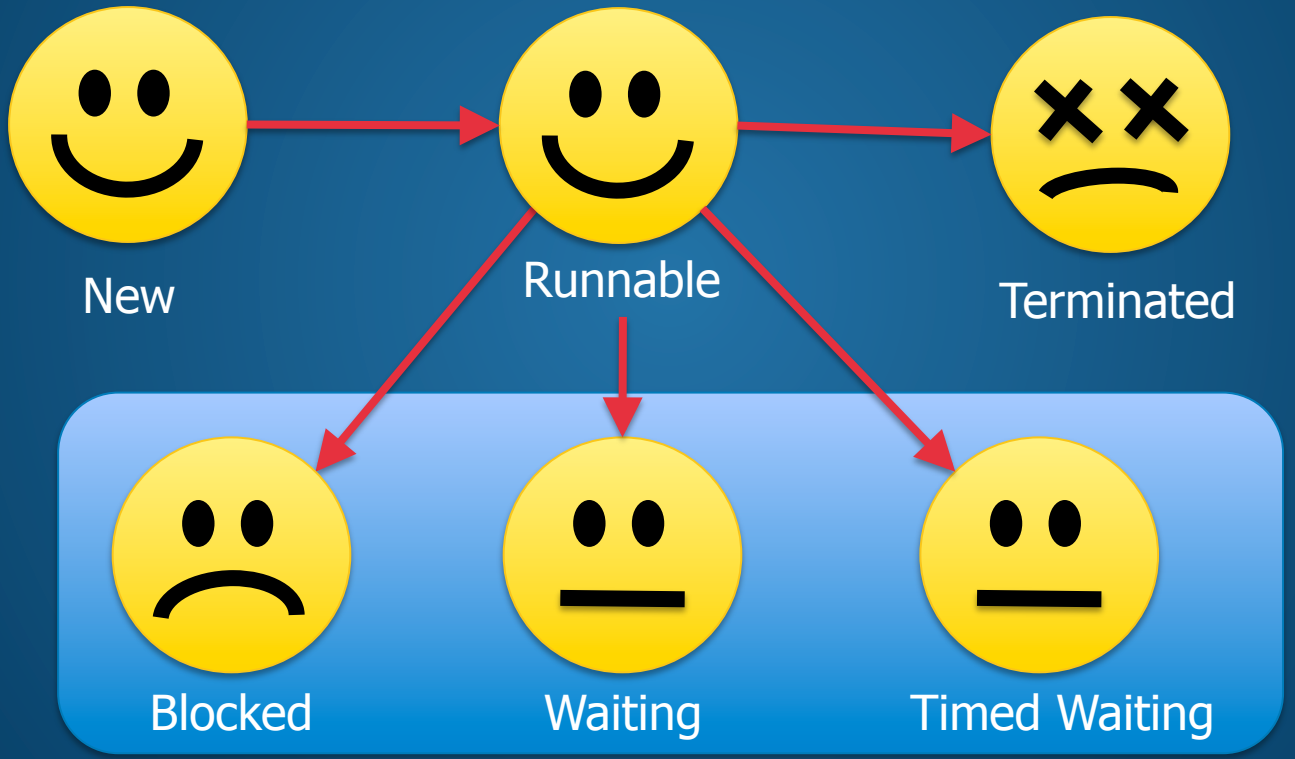
Introduction

Image: amazon.com

Thread States

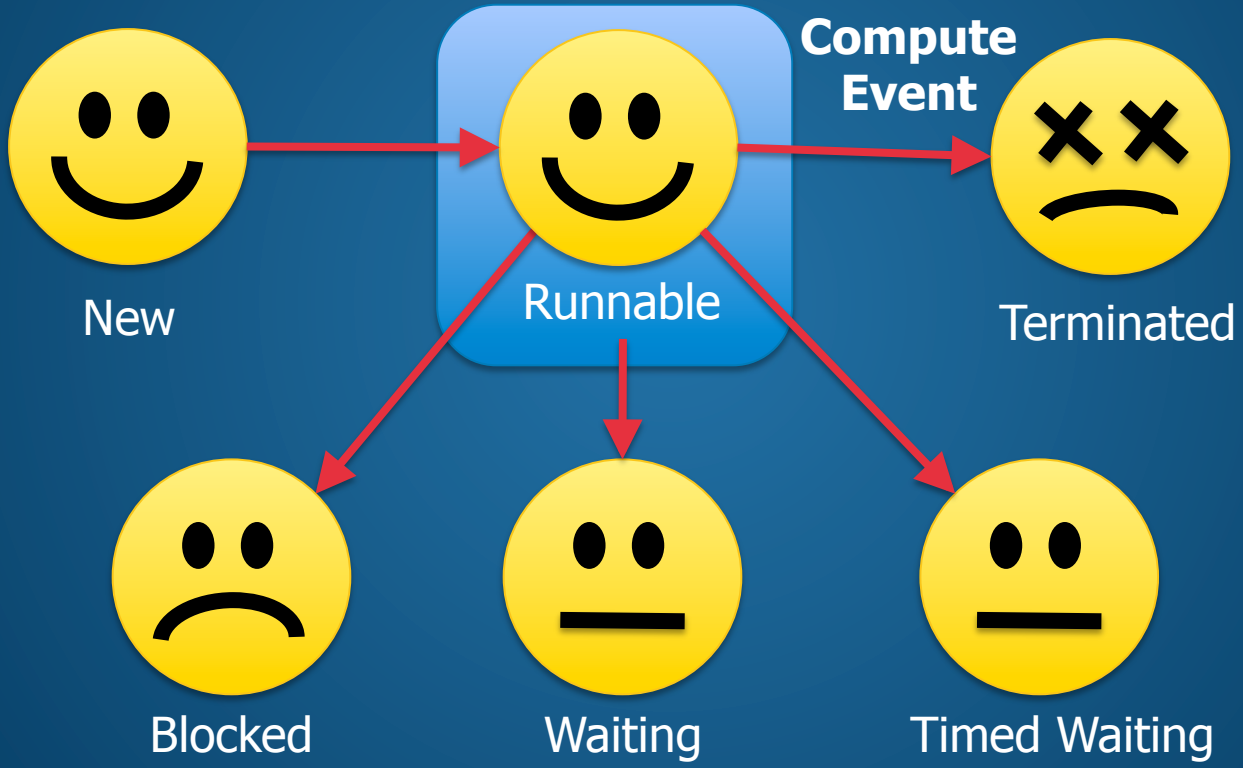


Thread States



Latency Events

Thread States



Compute | Latency Events

- Spending time on the CPU - “doing something”
- Dictated by thread scheduling

- Waiting on another thread

- Blocked on a lock

JVM Time Consumers

Garbage Collection Cycles

JIT Compilation

Garbage Collection

Types

Size of heap

When compactions occur

JIT Compilation

Compilation at runtime

Outperform interpreted code

Based on stats of “hot” code

Investigation

Capture as much evidence before altering the environment

Image: https://www.flickr.com/photos/citizenbrick

JMX

Remote JMX is your friend

Allows collecting data and performing operations on JVM

jcmd

Image: lego.wikia.com

jcmd

Why? Lightweight console tool to invoke diagnostic commands

When? Need to enable JVM options or gather JVM state (Java 7 u 4)

jcmd

```
> jcmd
7470 io.kcdc.MailServer
7471 io.kcdc.FileServer
7472 io.kcdc.CleanupAgent
23713 sun.tools.jcmd.JCmd
```

Executed with no options, displays all local JVMs by PID (like jps)

jcmd

```
> jcmd <PID> help
```

View help options on target JVM process ID

jcmd

```
com.sun.tools.attach.AttachNotSupportedException: Unable to open socket file: target process
not responding or HotSpot VM not loaded
     at sun.tools.attach.LinuxVirtualMachine.<init>(LinuxVirtualMachine.java:106)
     at sun.tools.attach.LinuxAttachProvider.attachVirtualMachine(LinuxAttachProvider.java:63)
     at com.sun.tools.attach.VirtualMachine.attach(VirtualMachine.java:213)
     at sun.tools.jcmd.JCmd.executeCommandForPid(JCmd.java:140)
     at sun.tools.jcmd.JCmd.main(JCmd.java:129)
```

When attaching to a JVM, you must be issuing the jcmd as the same OS user as target JVM

jcmd

```
> jcmd <PID> PerfCounter.print
```

Displays a vast array of performance related metrics

jcmd

```
java.threads.daemon=20
java.threads.live=26
java.threads.livePeak=38
java.threads.started=697
```

Example of output on threads from PerfCounter

jcmd

```
sun.gc.generation.0.capacity=45088768
sun.gc.generation.0.maxCapacity=179306496
sun.gc.generation.0.minCapacity=45088768
sun.gc.generation.0.name="new”
```

Example of output on GC from PerfCounter

jcmd

```
> jcmd <PID> Thread.print
```

Thread dump (just like jstack)

Can use “-l” for synchronizer information

jcmd

```
> jcmd <PID> ManagementAgent.start
    jmxremote.ssl=false
    jmxremote.authenticate=false
    jmxremote.port=<Port #>
```

Enable JMX on already running JVM

top

Image: http://www.wikihow.com/Build-a-Big-LEGO-Tower

top

Why? Console tool to provide live results on CPU utilization

When? Unsure if JVM slowness is attributed to heavy compute events

top

```
> top
top - 14:22:01 up 208 days, 22:22,  2 users,  load average: 0.14, 0.07, 0.01
Tasks: 228 total,   1 running, 227 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.8%us,  0.6%sy,  0.0%ni, 98.2%id,  0.0%wa,  0.2%hi,  0.3%si,  0.0%st
Mem:  24608272k total, 22222568k used,  2385704k free,   639632k buffers
Swap:  4194296k total,        0k used,  4194296k free,  8872124k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
20709 minion    20   0 2939m 492m  19m S  1.0  2.1  44:04.37 java
 1328 minion    20   0 2753m 270m  15m S  0.7  1.1   9:35.16 java
 2281 minion    20   0 2947m 532m  19m S  0.7  2.2  36:58.92 java
 6236 minion    20   0 2751m 323m  16m S  0.7  1.3  87:53.46 java
16416 minion    20   0 15036 1336  948 R  0.7  0.0   0:00.03 top
19124 minion    20   0 2750m 288m  16m S  0.7  1.2  60:26.22 java
21246 minion    20   0 2743m 303m  15m S  0.7  1.3  99:47.73 java
```

See JVMs by CPU%

top

```
> top
top - 14:22:01 up  208 days, 22:29,  2 users,  load average: 0.02, 0.03, 0.00
Tasks: 227 total,   1 running, 226 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.7%us,  0.8%sy,  0.0%ni, 98.0%id,  0.0%wa,  0.3%hi,  0.3%si,  0.0%st
Mem:  24608272k total, 22258860k used,  2349412k free,   639640k buffers
Swap:  4194296k total,        0k used,  4194296k free,  8872236k cached
 Show threads On
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1452 minion    20   0 2753m 270m  15m S  4.2  1.1   2:43.92 java
28368 minion    20   0 2743m 269m  14m S  2.9  1.1  29:19.49 java
22498 minion    20   0 2746m 299m  14m S  2.6  1.2  28:01.51 java
11563 minion    20   0 2742m 253m  14m S  2.3  1.1  26:57.11 java
17424 minion    20   0 16384 2572  952 R  2.0  0.0   0:01.95 top
```

Press “Shift + H” while top is running

top

```
> top –H –p <PID>
```

View threads by CPU time on a specific JVM process

top

```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
20812 minion    20   0 2939m 492m  19m S  9.6  2.1  14:48.59 java
```

20812 is the thread ID

```
> printf "%x" 20812
514C
```

Convert decimal thread ID to hex (514C)

```
"metrics-graphite-pickle-reporter-thread-1" daemon prio=10
tid=0x00007feb88542000 nid=0x514c
```

Identify thread in thread dump (native thread ID)

Advanced

strace

Image: https://rebrick.lego.com/en-US/bookmark/daniel-sicolo-blog--lego-bob-ross-brilliant-/irp167

strace

Advanced

Why? Need lower-level details of OS and JVM interaction

When? Unsure if JVM is hitting OS limits and restricting performance

strace

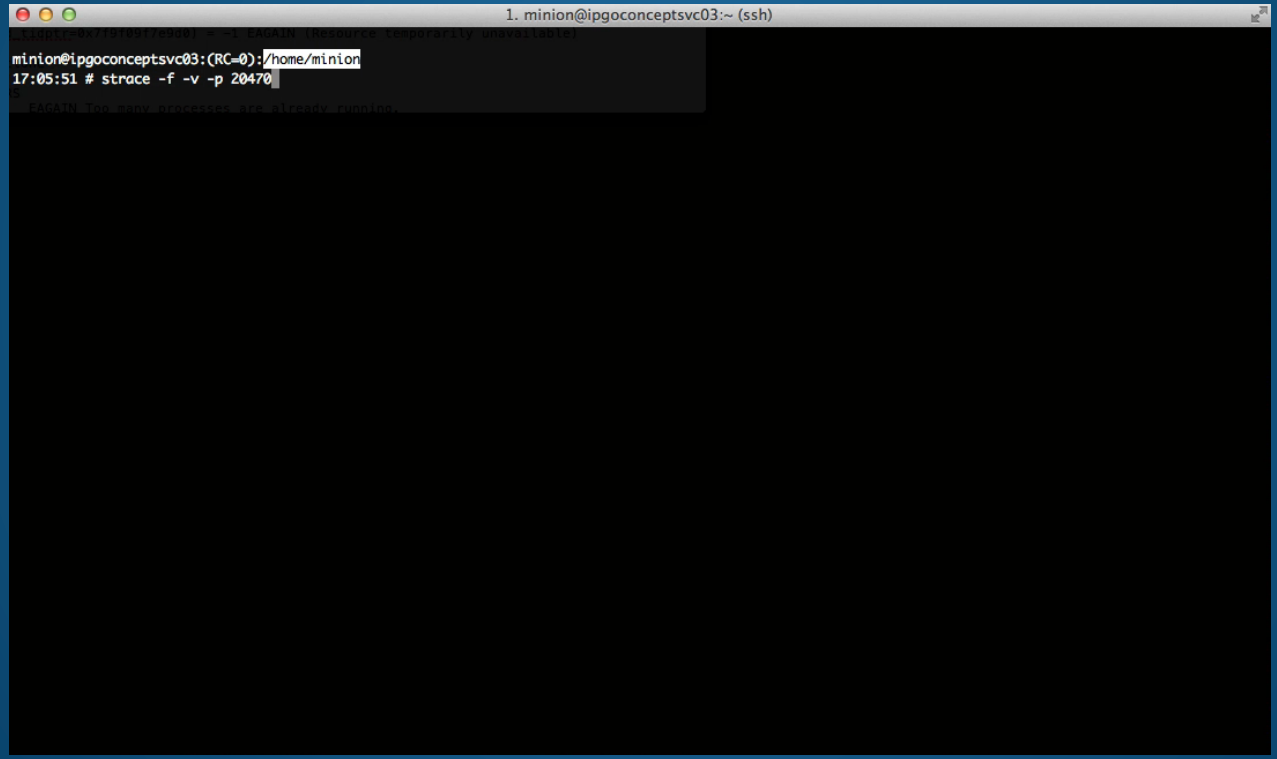
Advanced

```
> strace -f -v -p <PID>
```

Print out system calls by process ID (includes child processes)

strace

Advanced



strace

Advanced

```
java.lang.OutOfMemoryError: unable
to create new native thread
```

What causes this?

strace

Advanced

```
[pid 11242] 18:06:35
clone(child_stack=0x7f9f09f7dff0, flags=CLONE_VM|
CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|
CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|
CLONE_CHILD_CLEARTID,
parent_tidptr=0x7f9f09f7e9d0, tls=0x7f9f09f7e700,
child_tidptr=0x7f9f09f7e9d0) = -1 EAGAIN
(Resource temporarily unavailable)
```

strace

Advanced

```
> man clone
...
ERRORS
       EAGAIN Too many processes are already running.
```

strace

Advanced

```
> ps auxw |
  grep <SEARCH PHRASE> |
  awk '{print "-p " $2}' |
  xargs strace -f -v
```

Tracing across different PIDs

Java Visual VM

Image: http://www.brickshow.com/episode46-5-7955

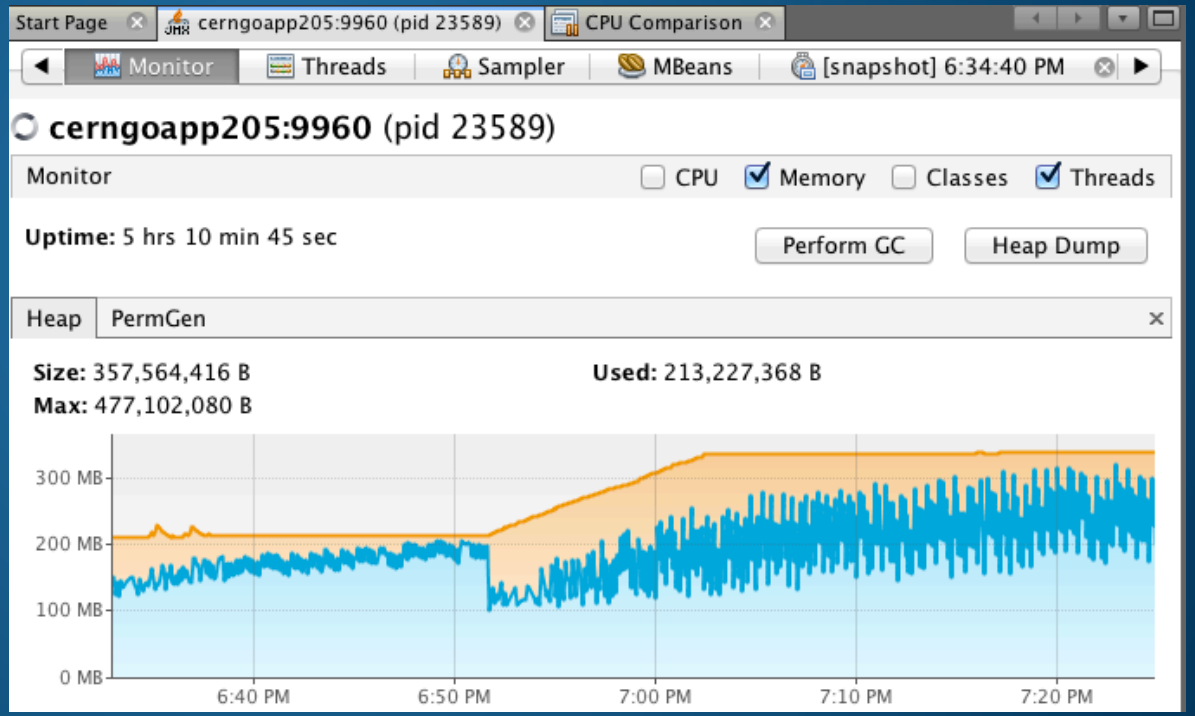
Visual VM

Why? Want simple visualizations and profiling of the JVM

When? Initial assessments of the JVM when troubleshooting performance

Visual VM

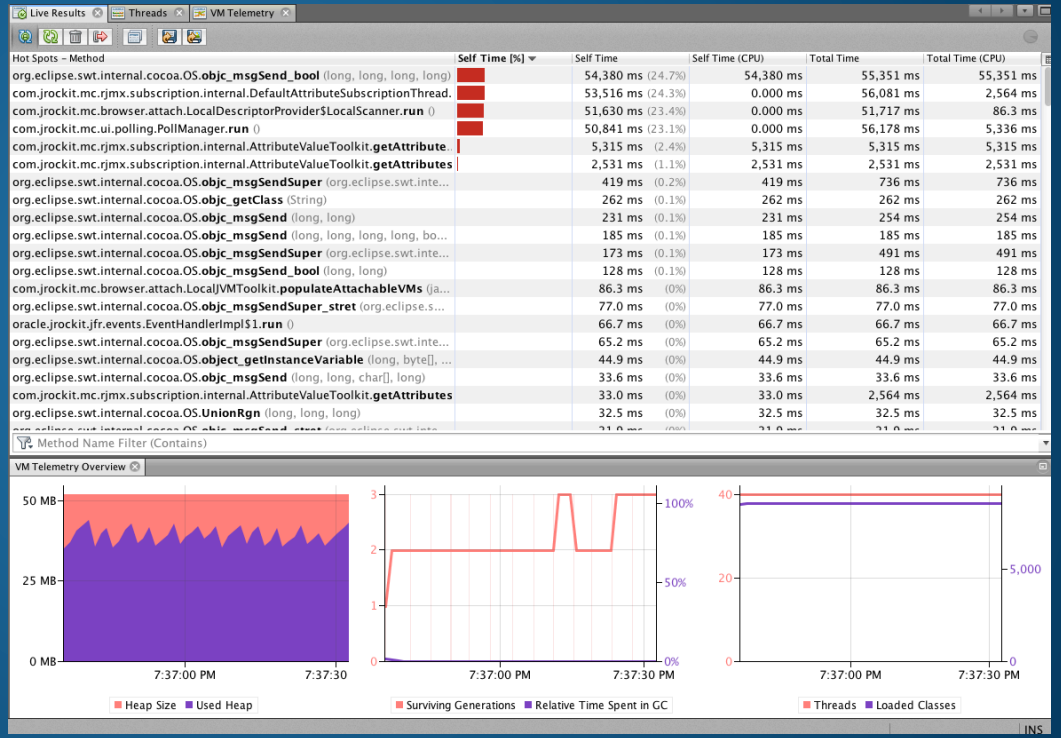
```
> jvisualvm
```

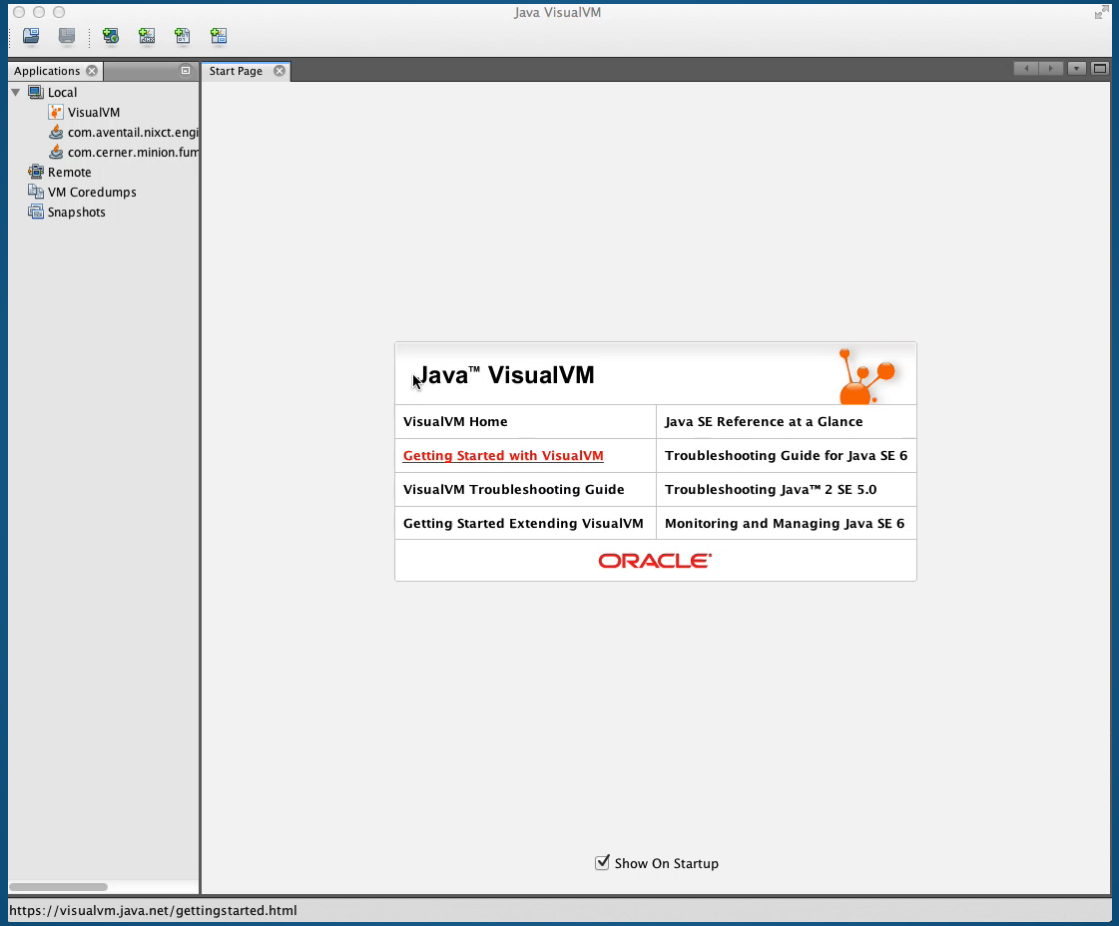


Visual VM

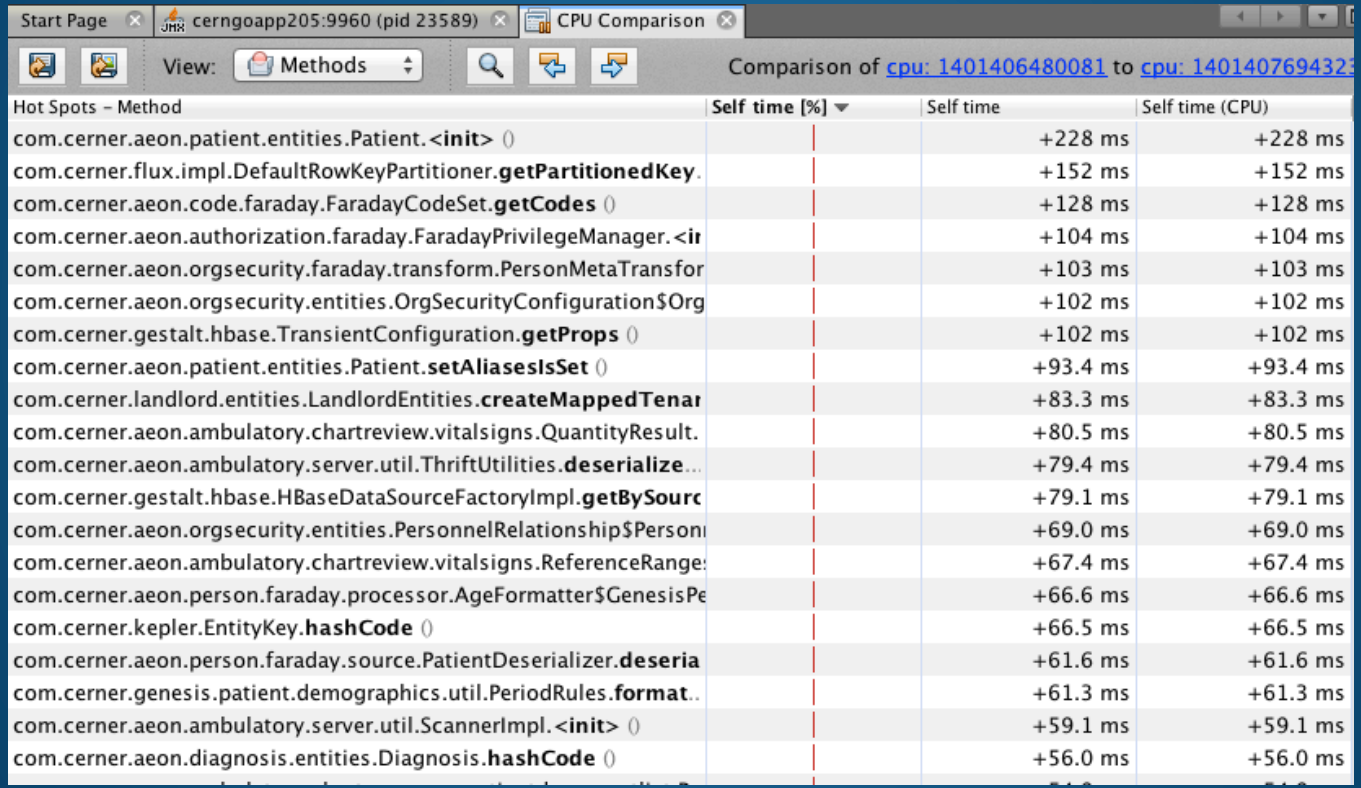
vs.

Netbeans Profiler




Java VisualVM
Applications
Local
VisualVM
com.aventail.nixct.engi
com.cerner.minion.fun
Remote
VM Coredumps
Snapshots
Start Page
Java™ VisualVM
VisualVM Home
Java SE Reference at a Glance
Getting Started with VisualVM
Troubleshooting Guide for Java SE 6
VisualVM Troubleshooting Guide
Troubleshooting Java™ 2 SE 5.0
Getting Started Extending VisualVM
Monitoring and Managing Java SE 6
ORACLE
Show On Startup
https://visualvm.java.net/gettingstarted.html

VisualVM Snapshot Compare



Java Mission Control

Image: lego.com

Java Mission Control

Why? It's like VisualVM…but better.

When? Initial assessments of the JVM when troubleshooting performance (Java 7 u 40).

Advanced analysis of flight recordings.

Java Mission Control

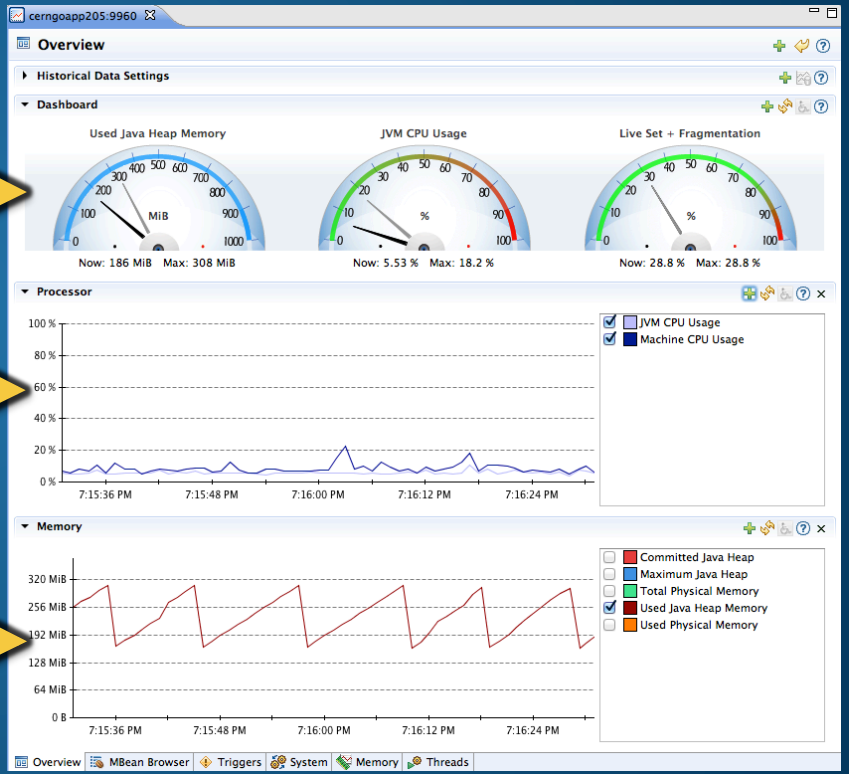
```
> jmc
```

Live gauges

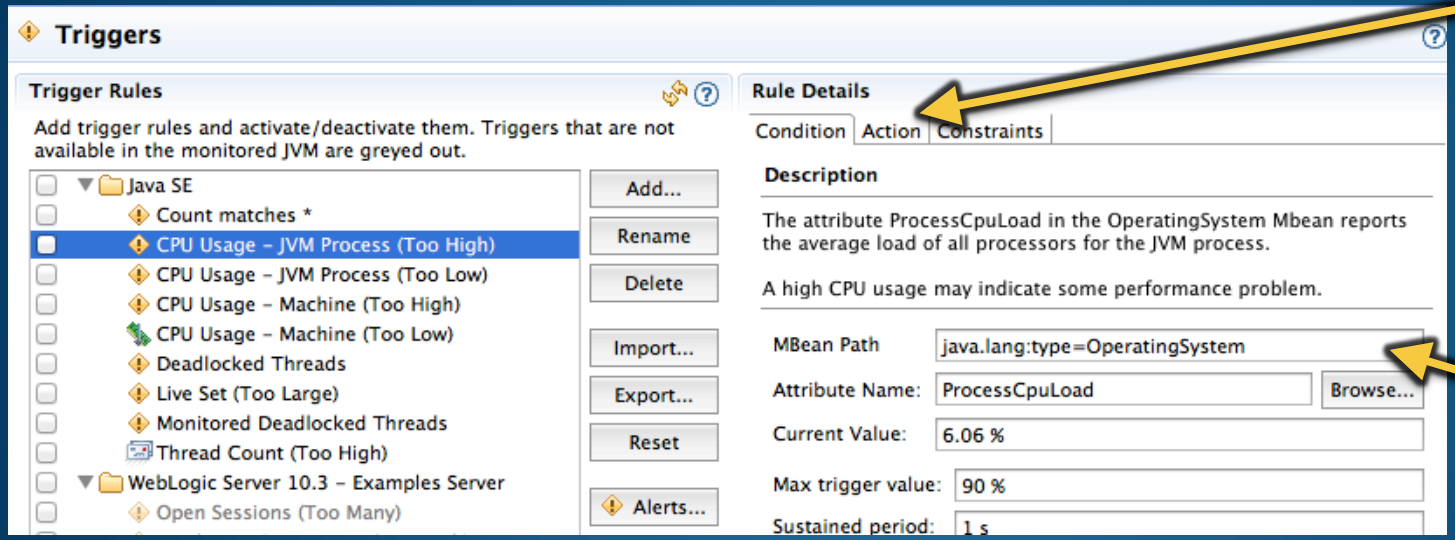
CPU

Memory

Can graph any MBean attribute



Java Mission Control

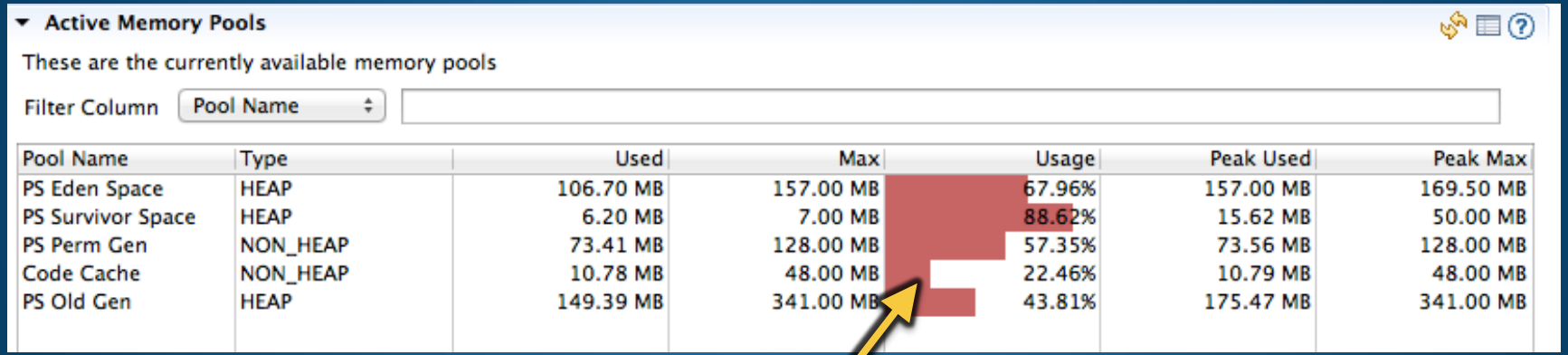


Action taken on trigger

Any MBean attribute

Actions can be: application alert, email, generate a heap dump, log to a file, echo out in your console, etc

Java Mission Control



Live usage on memory pools

Help define sizing to minimize thrashing

Java Flight Recorder

Image: lego.com

Java Flight Recorder

Why? Low-profile option to collect rich information of the JVM (1 – 2% overhead)

When? Needing to capture rich amount of information during profiling

Java Flight Recorder

Can continuously collect in a buffer

Output on shutdown

Restrictions

- **FREE** to use for development
- **NOT** FREE to use in production
  - Licensing:
    - Oracle Java SE Advanced
    - Oracle Java SE Suite

Java Flight Recorder

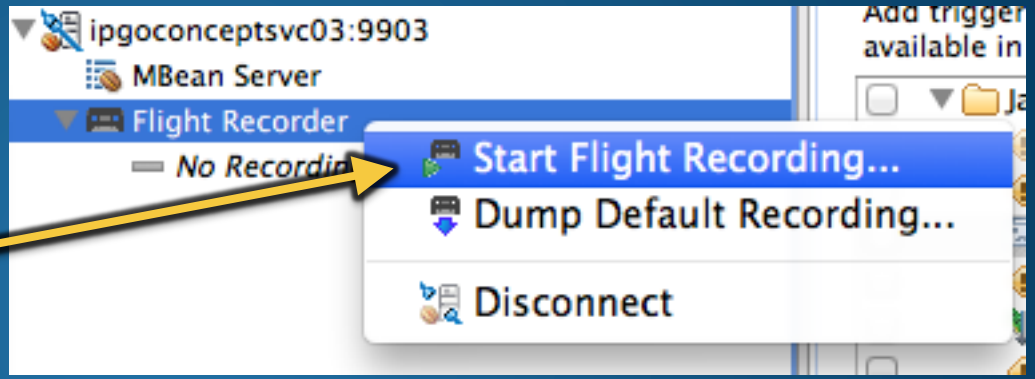
```
-XX:+UnlockCommercialFeatures
-XX:+FlightRecorder
```

Need to enable target JVM

Supported in Java 7 update 40

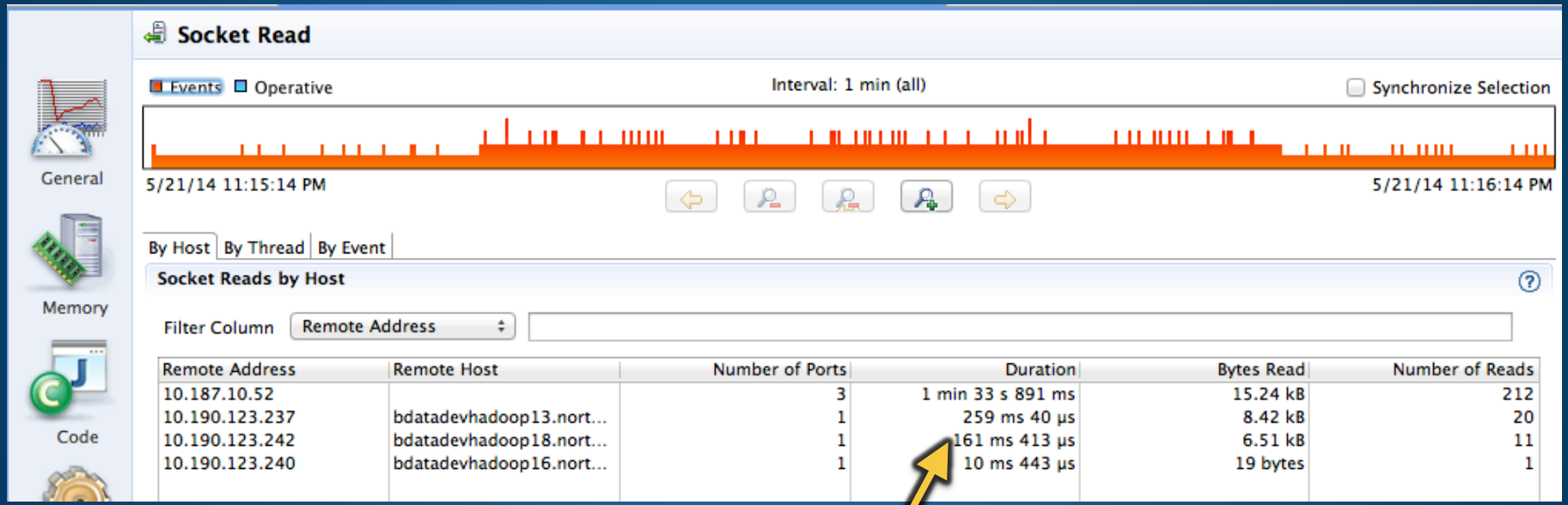
Java Flight Recorder

Enable on demand in Java Mission Control




```
> jcmd <PID> JFR.start duration=60s
    filename=myrecording.jfr
```

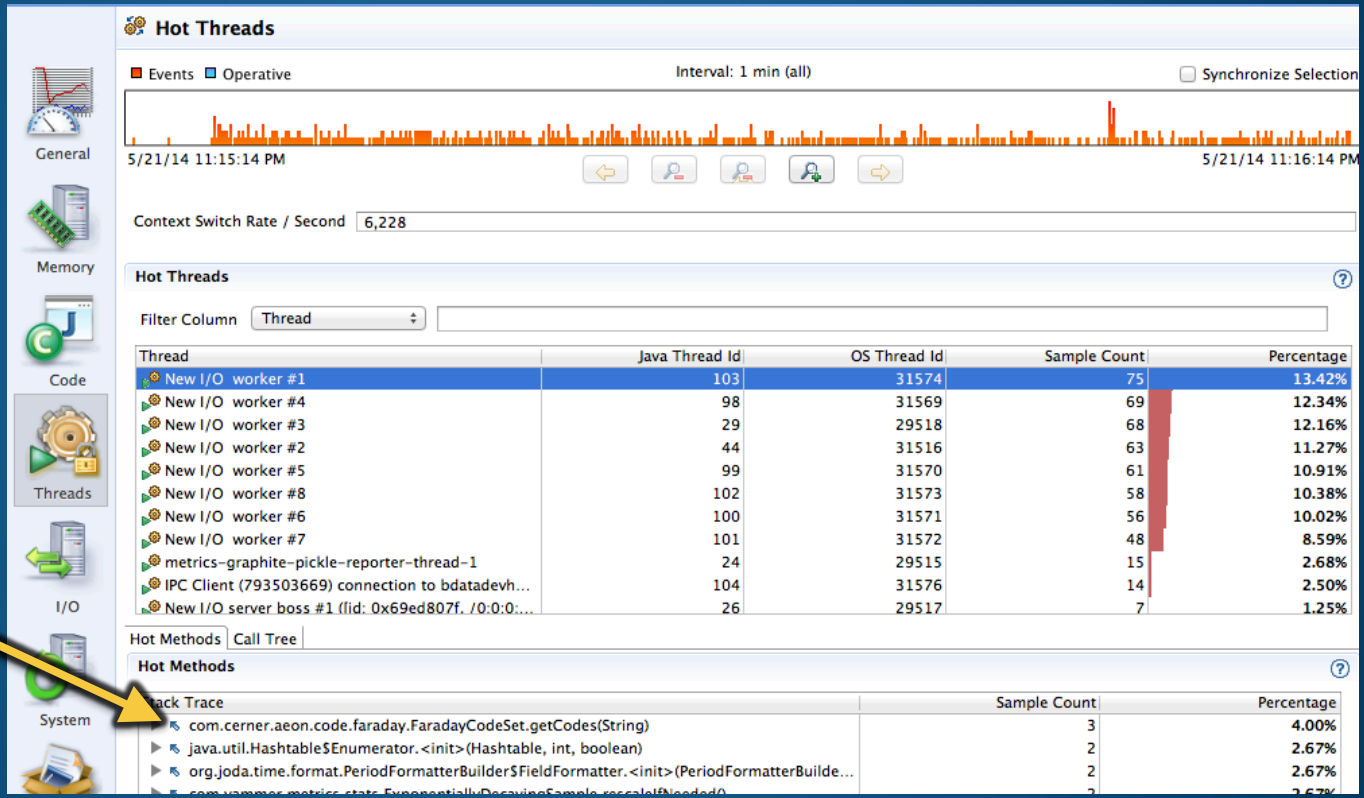
Java Flight Recorder



Time spent by host

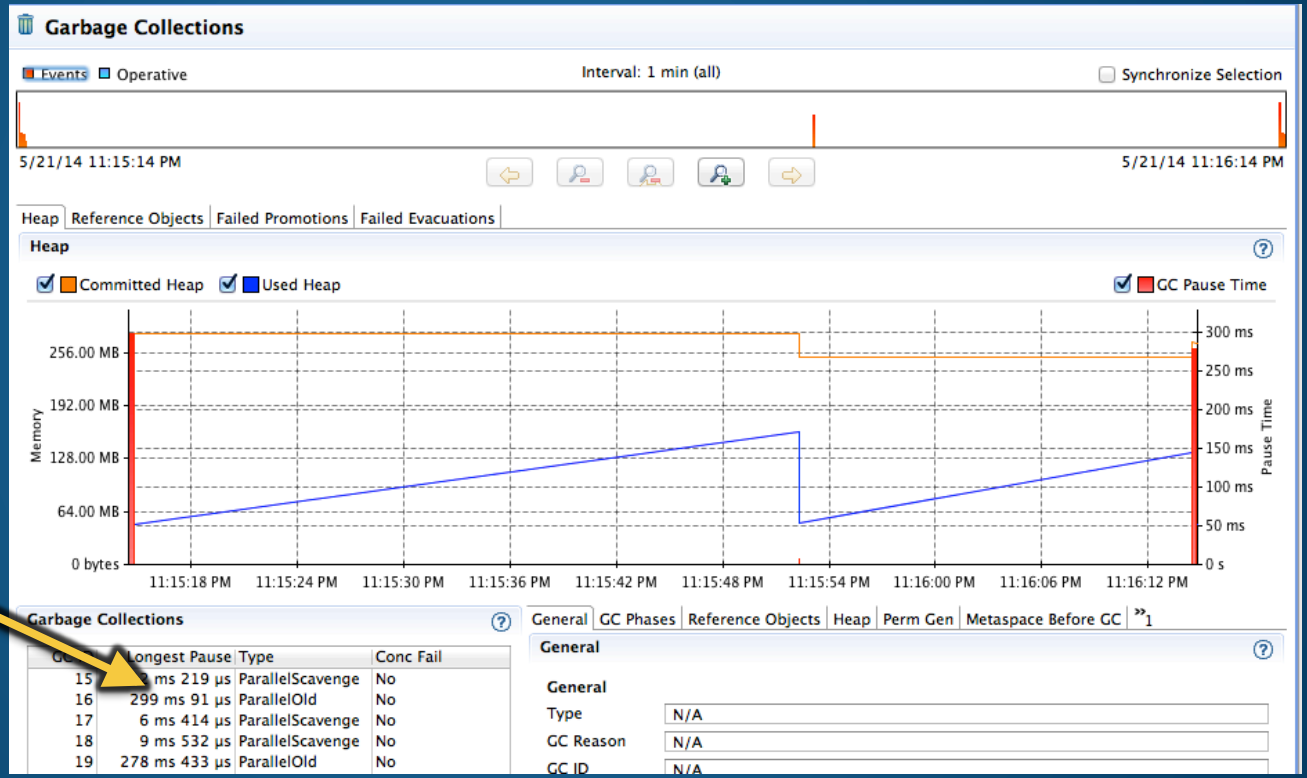
Java Flight Recorder

Hot methods

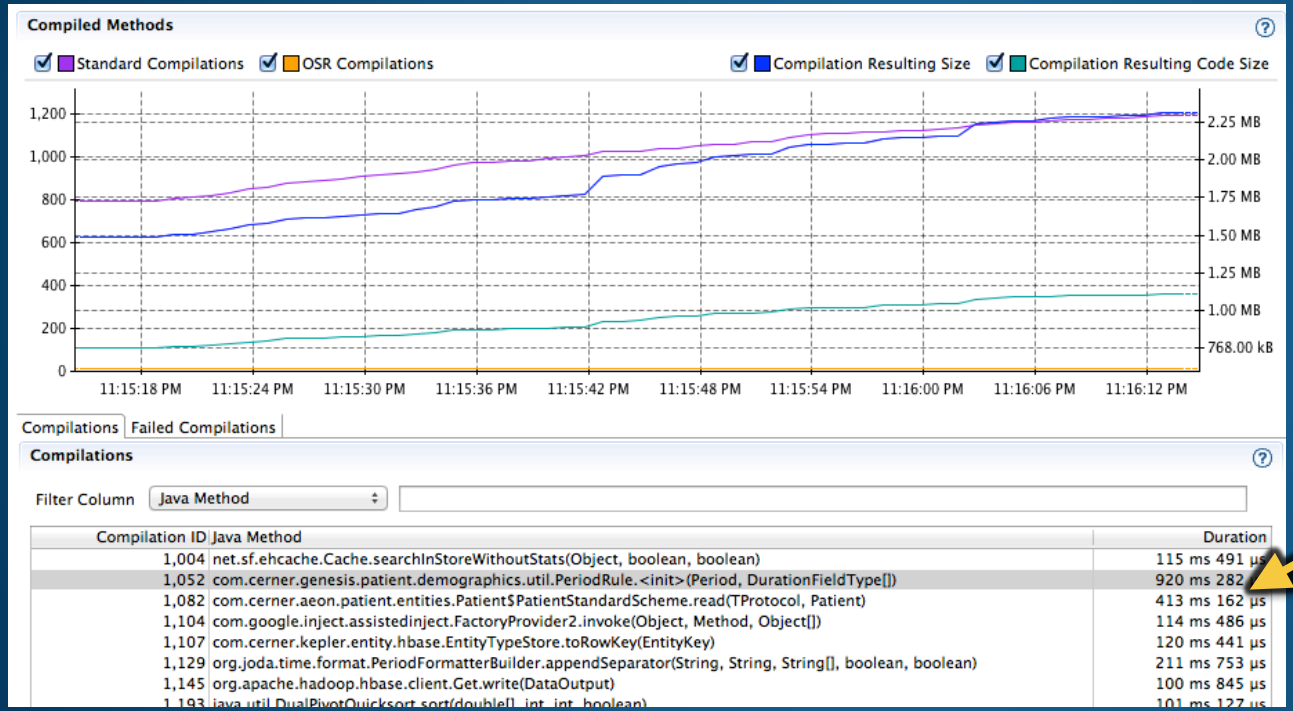


Java Flight Recorder

GC pause time

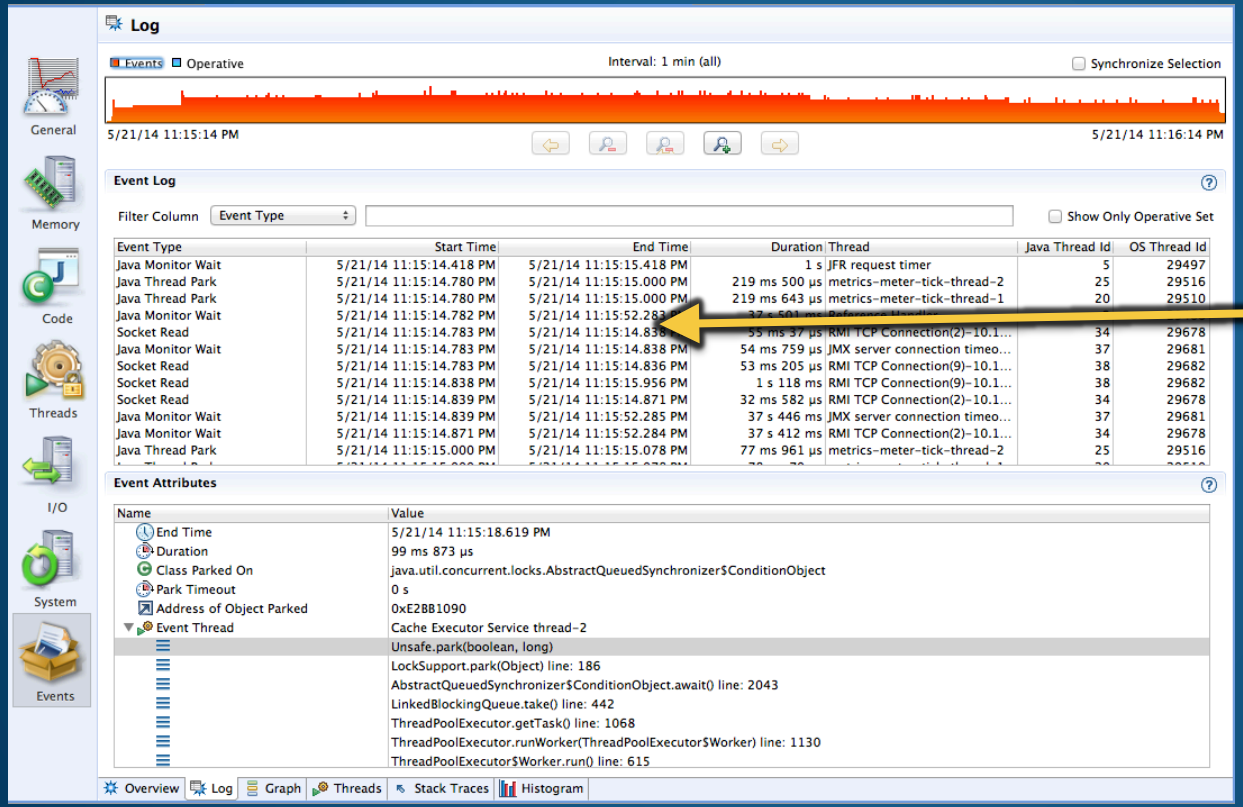


Java Flight Recorder



Compilation time

Java Flight Recorder



Timeline of events

Summary

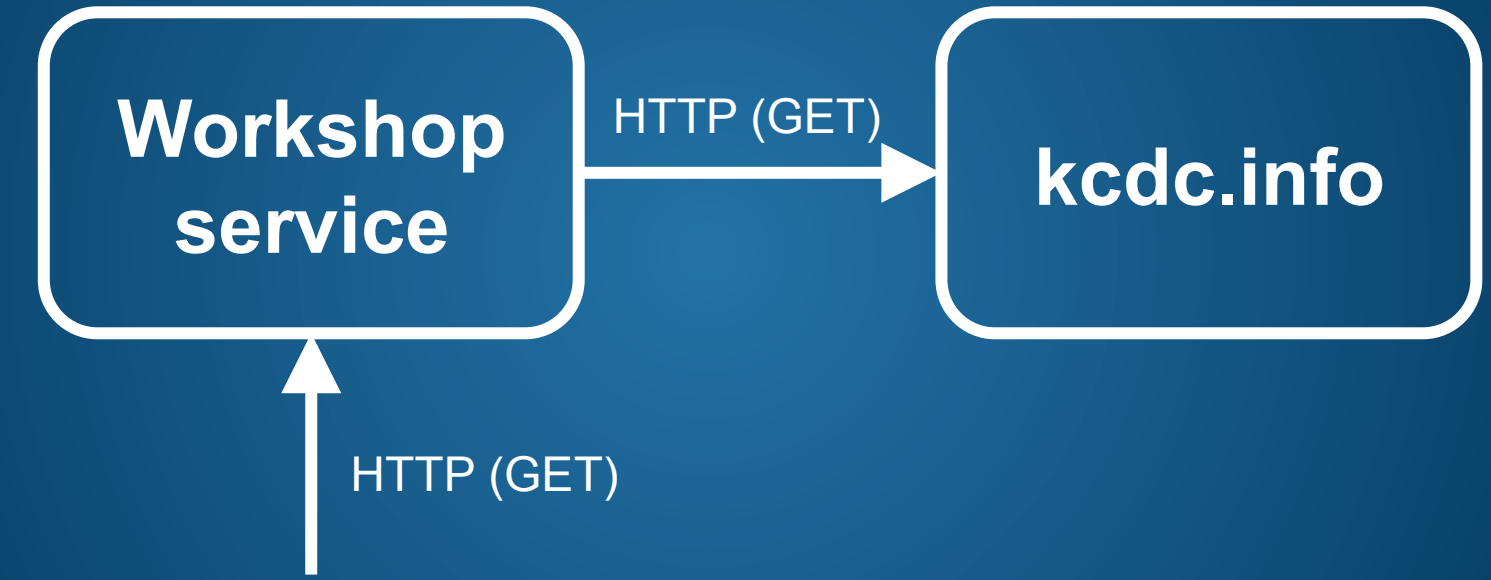
Know your command line tools

Capture metrics while it is still sluggish

Use Flight Recorder in dev to get details

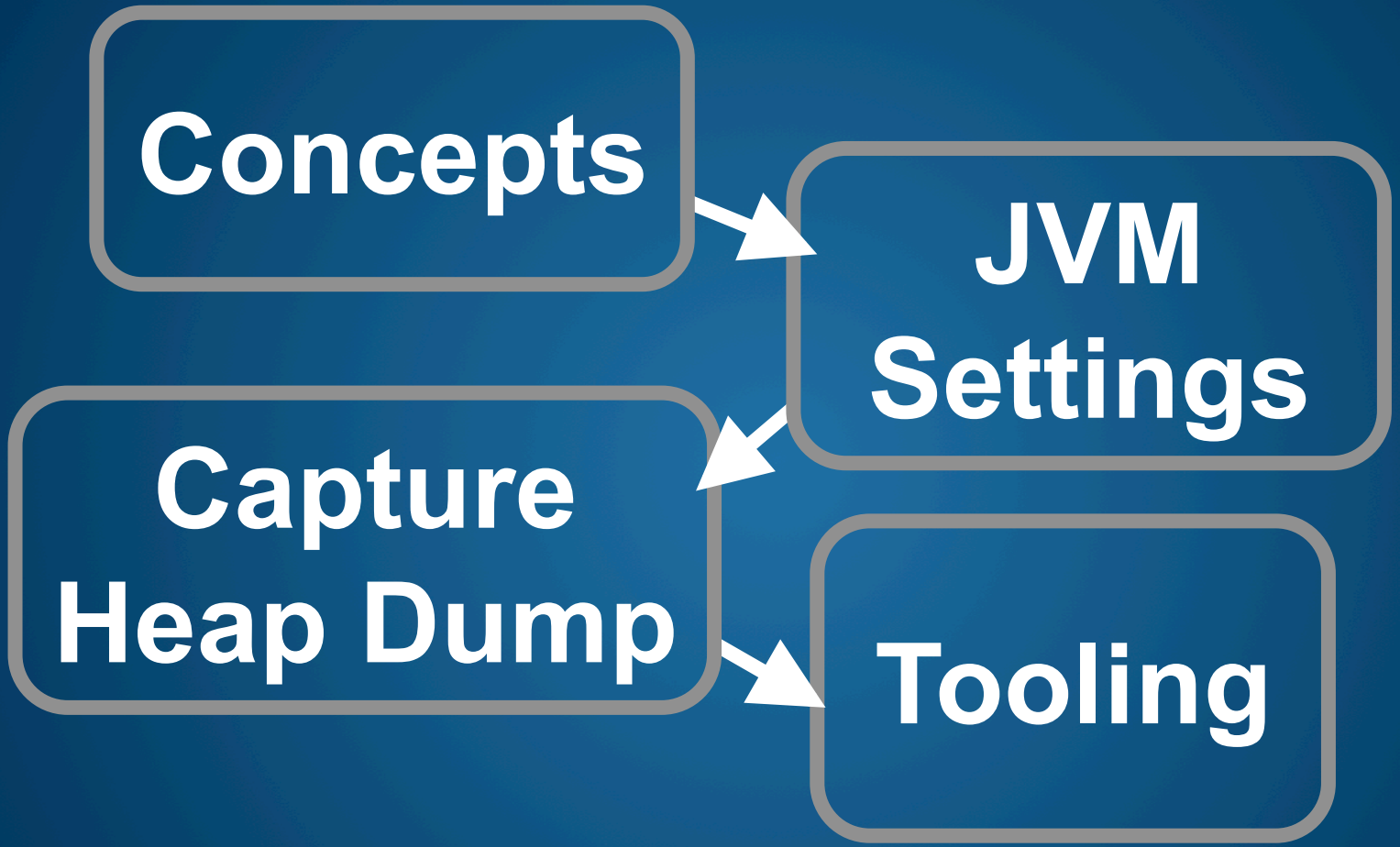
Identify compute / latency events

Workshop


Workshop service
HTTP (GET)
kcdc.info
HTTP (GET)
Query conference information

What's in your JVM?

Heap Analysis


Concepts
JVM Settings
Capture Heap Dump
Tooling

CONCEPTS

Dead objects are garbage.

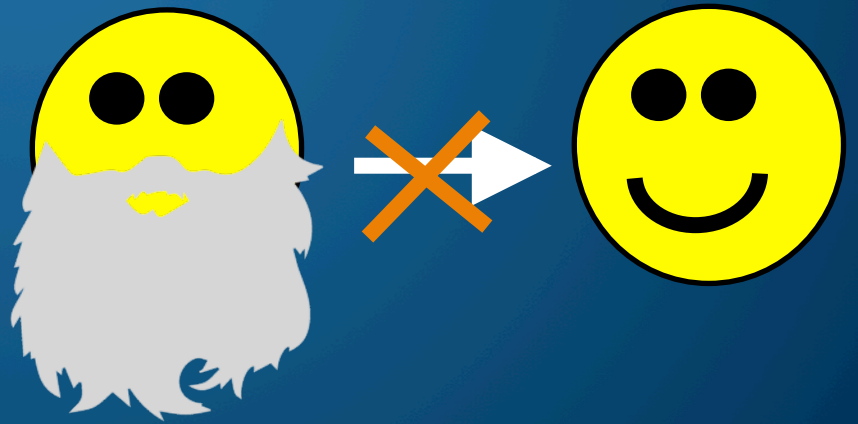
Weak Generational Hypothesis

#1

Most objects die young.

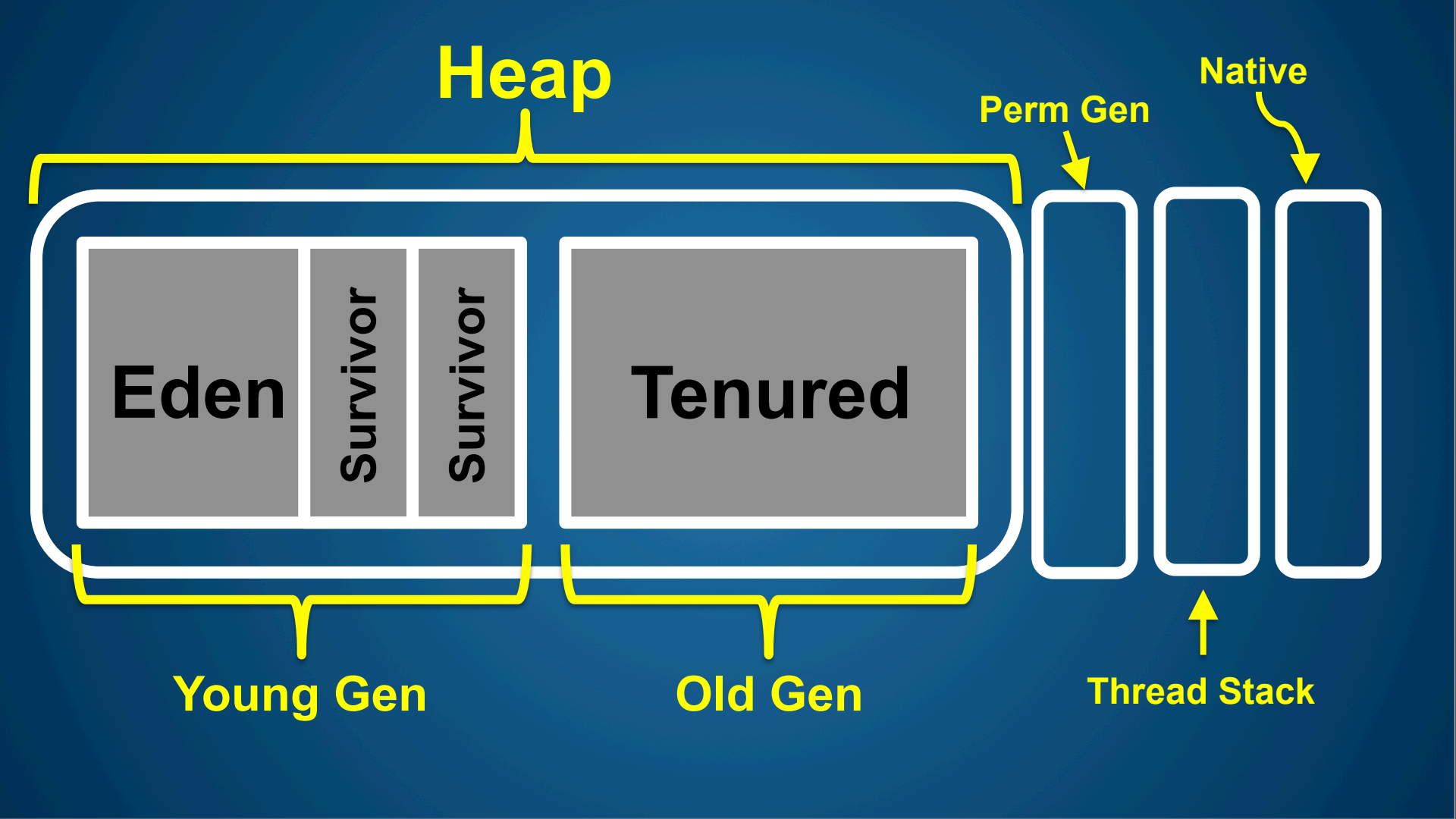
#2

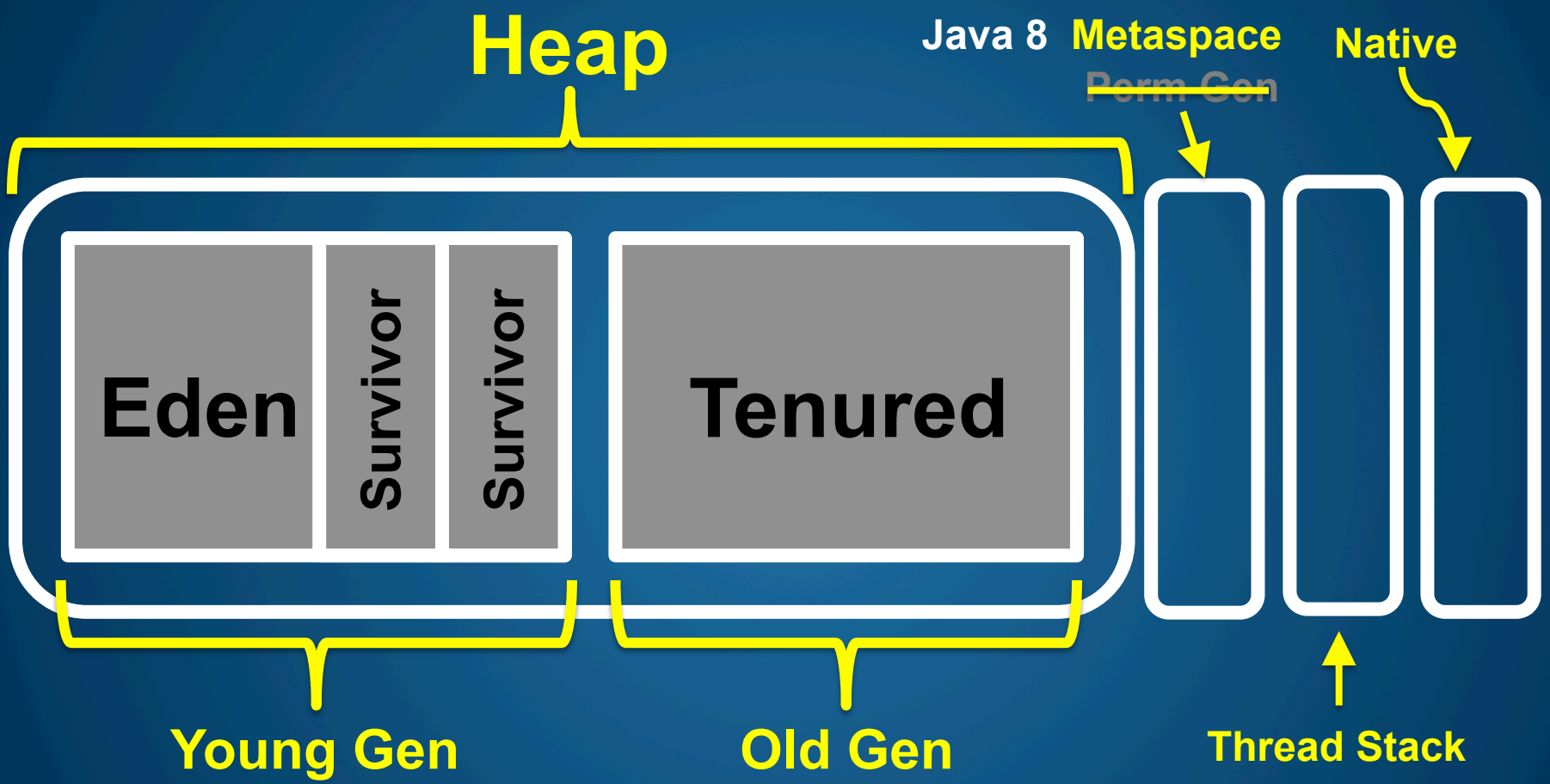
Few old objects reference young objects.

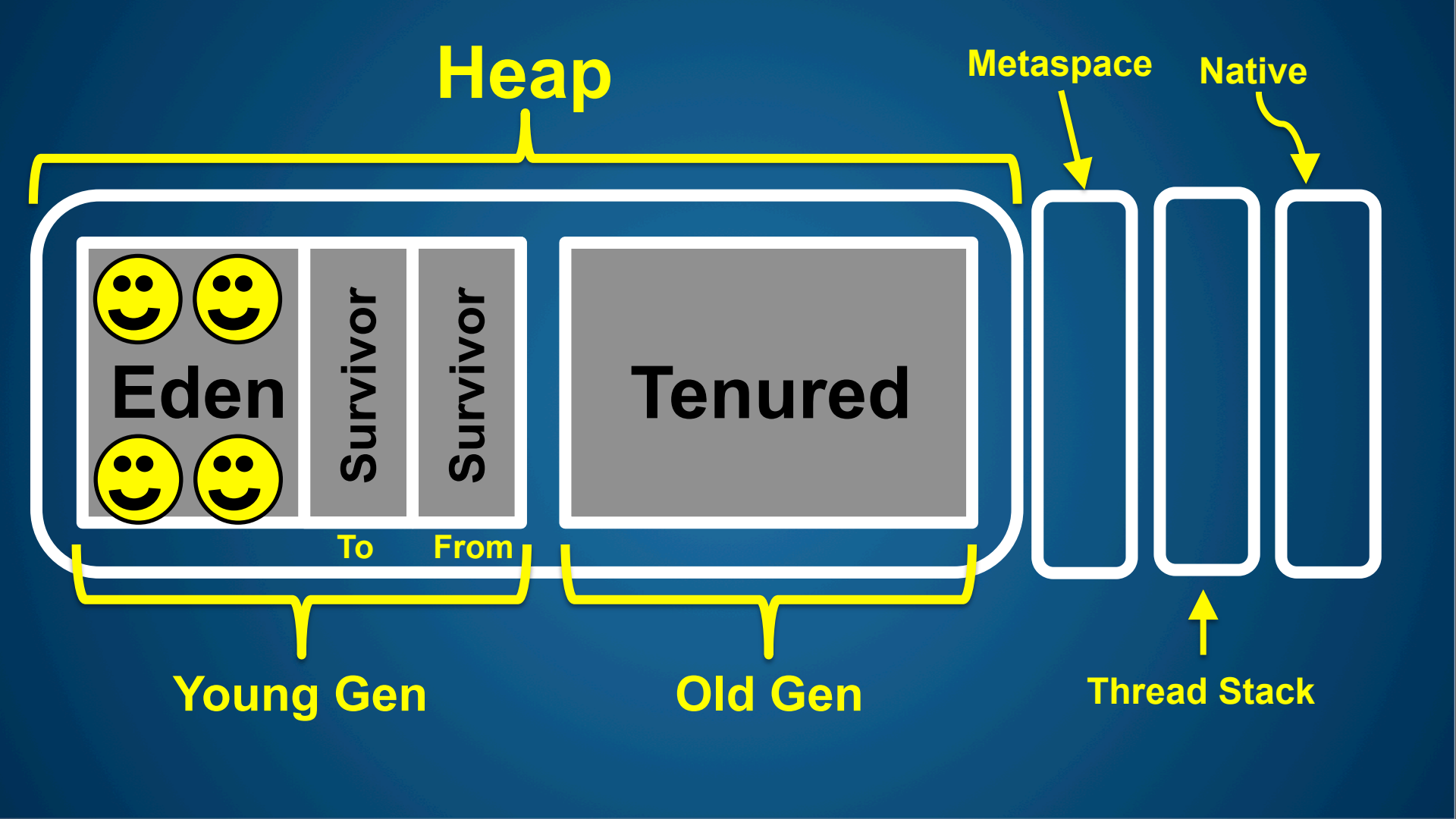


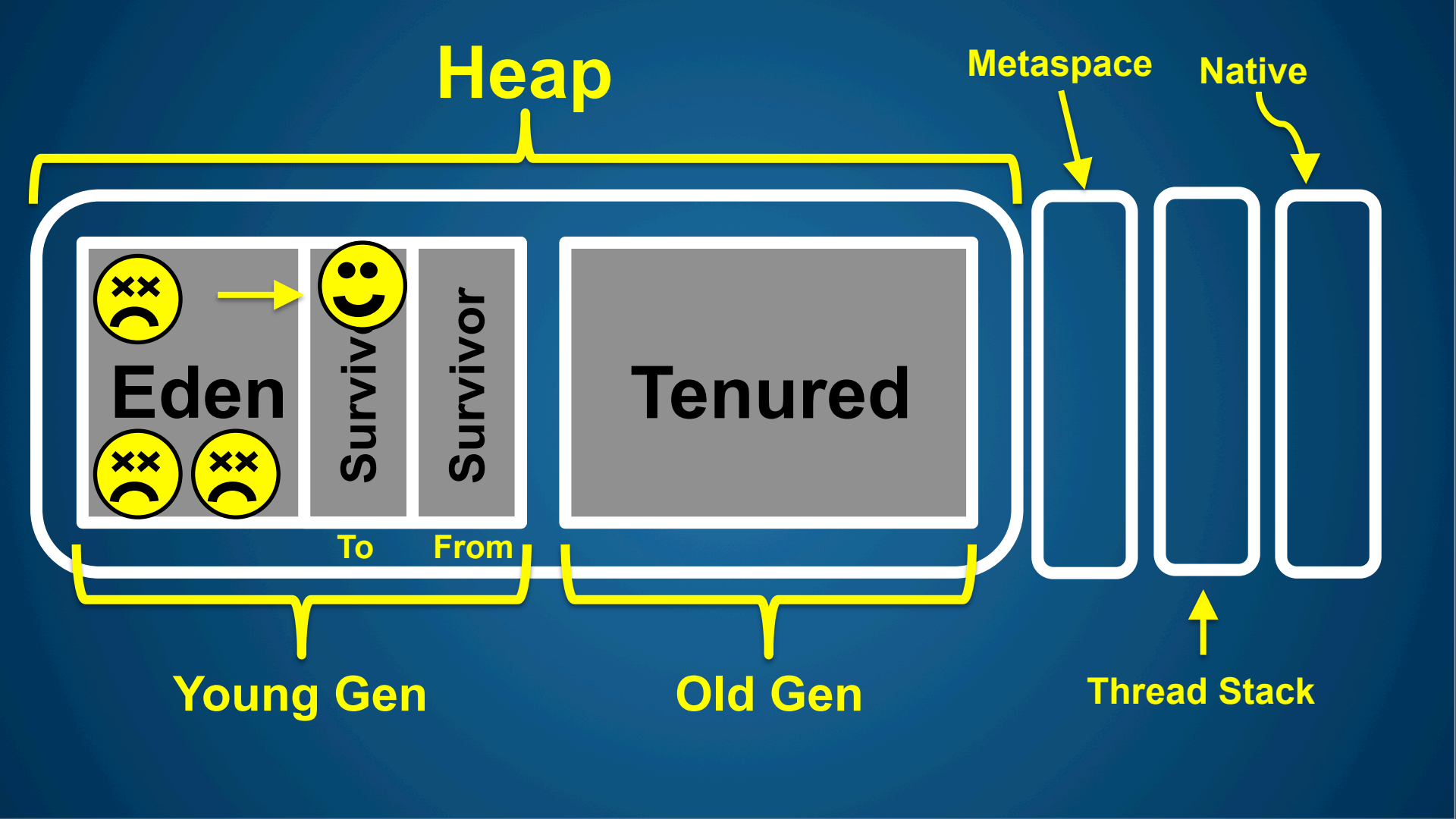
HotSpot VM is based on this hypothesis.

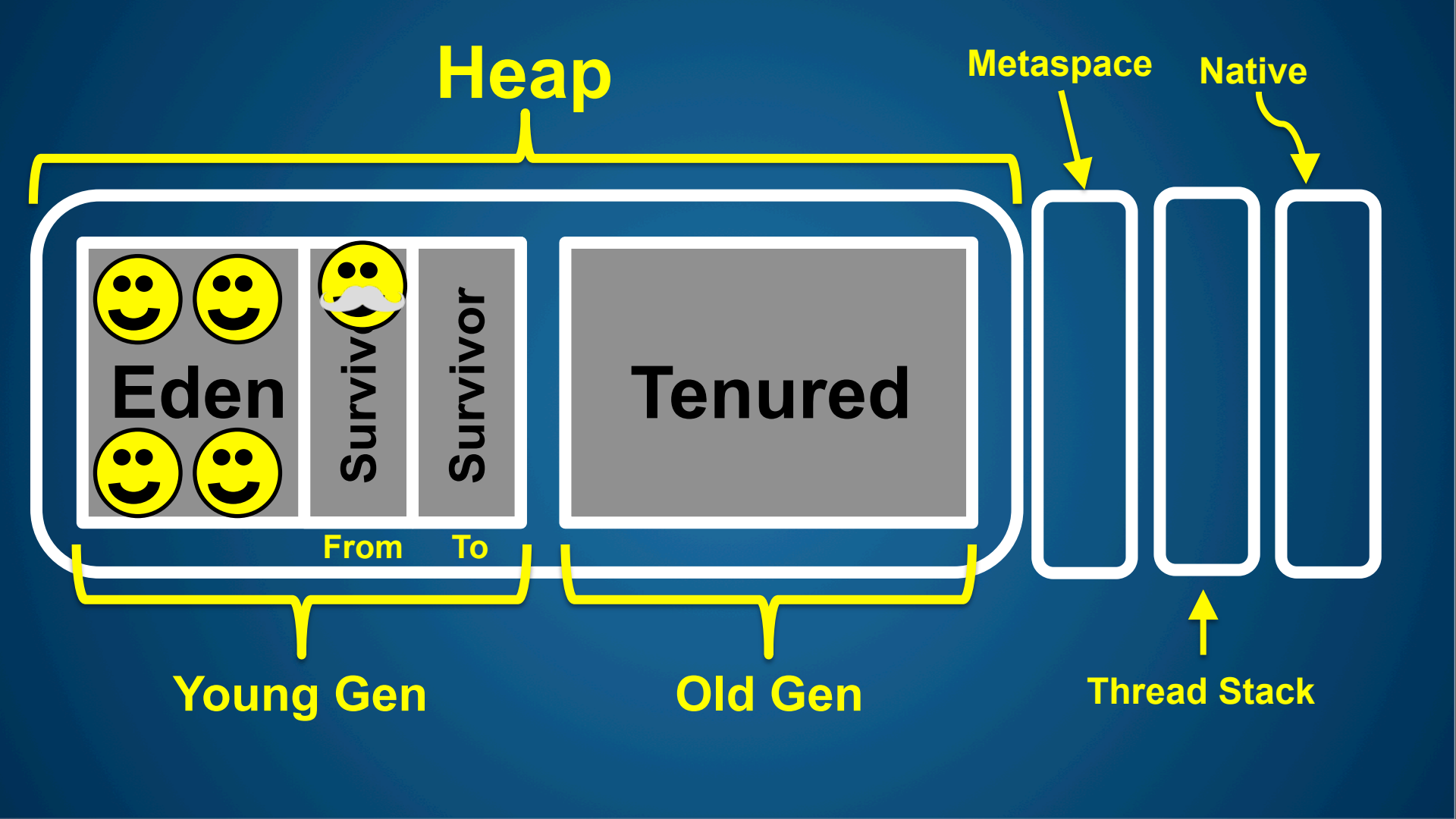
Uses a generational garbage collector.

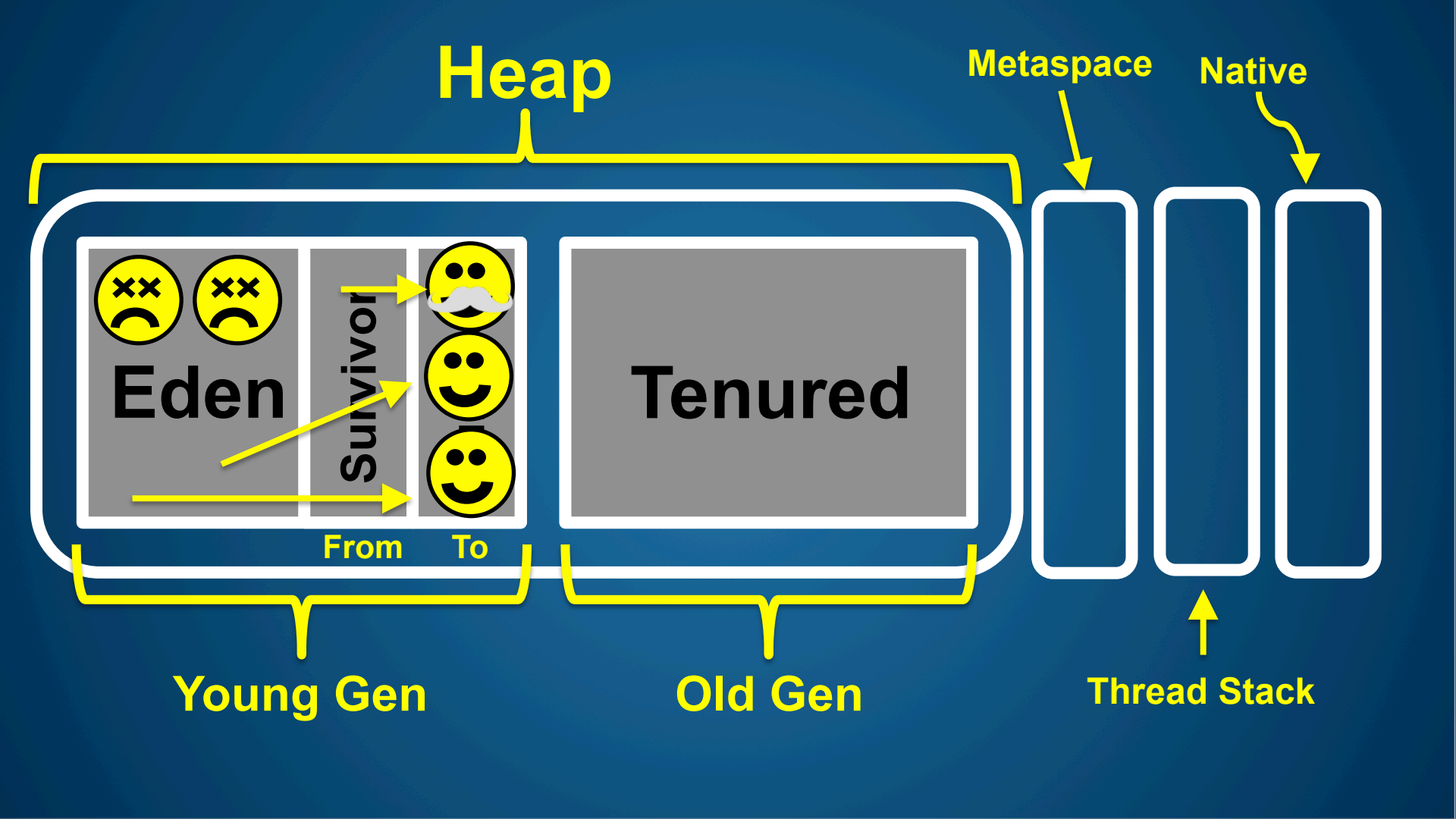

Heap
Perm Gen
Native
Eden
Survivor
Survivor
Tenured
Young Gen
Old Gen
Thread Stack

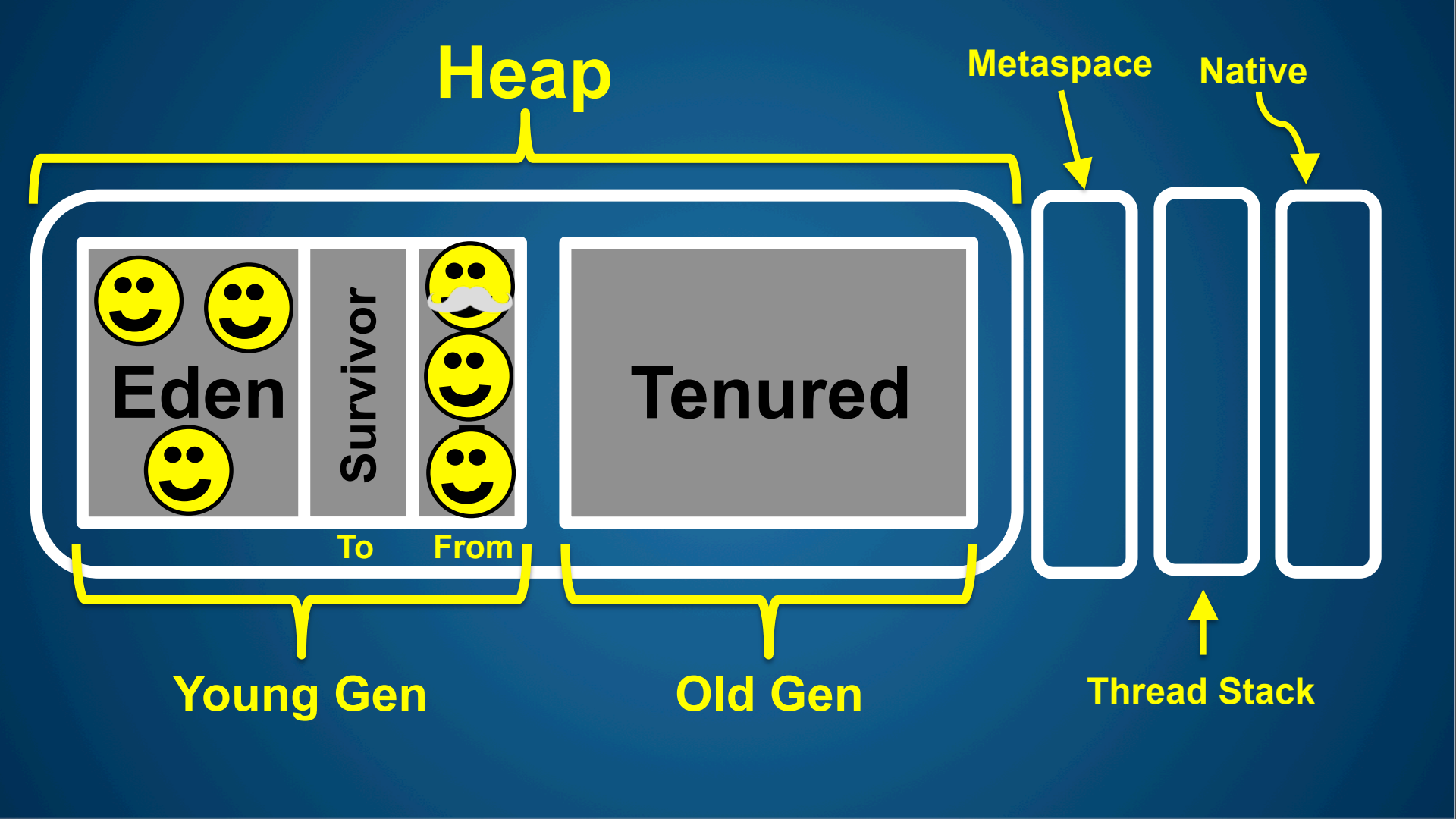

Heap
Java 8 Metaspace
Perm Gen
Native
Eden
Survivor
Survivor
Tenured
Young Gen
Old Gen
Thread Stack

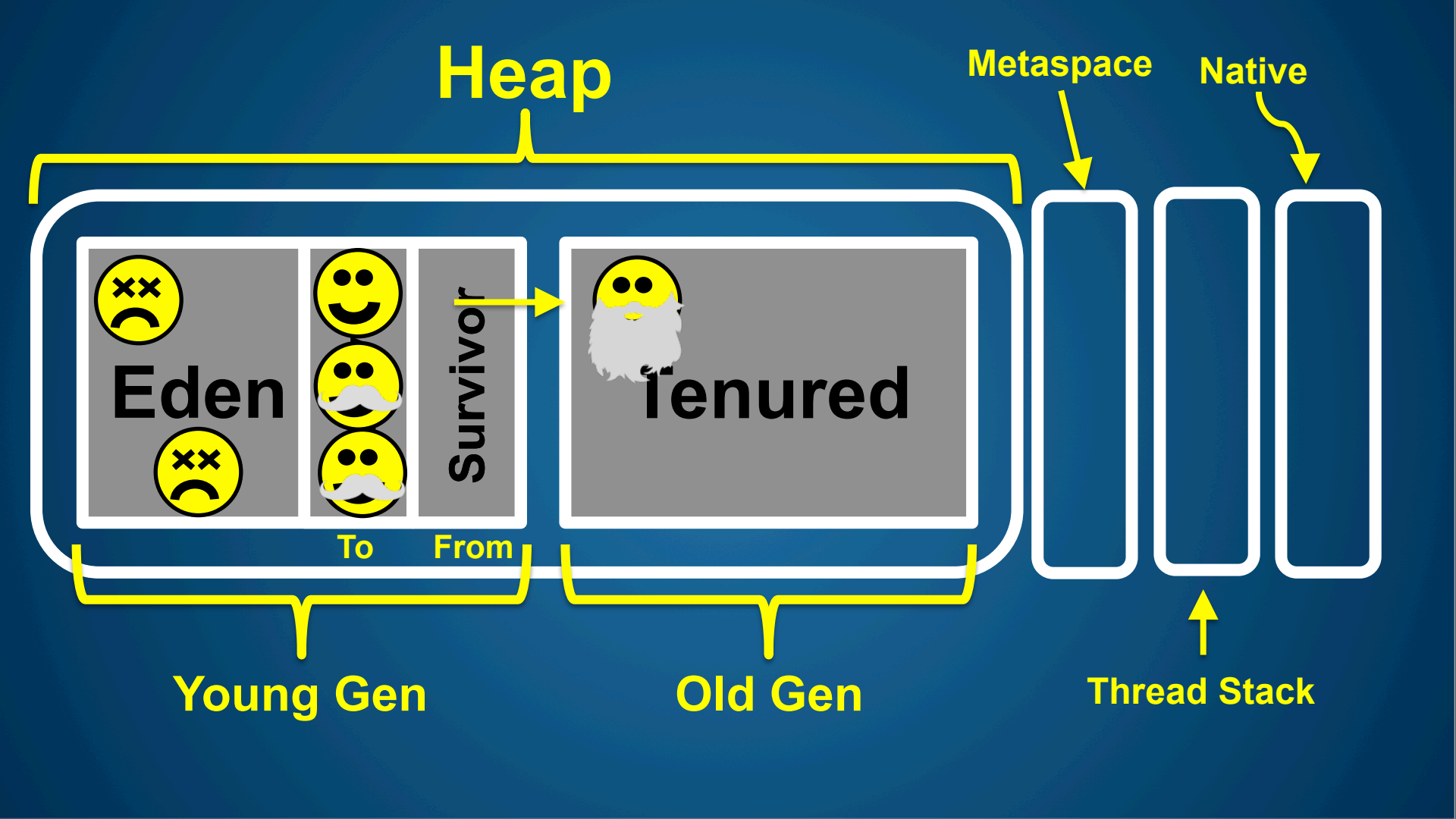

Heap
Metaspace
Native
Eden
Survivor
Survivor
Tenured
To
From
Young Gen
Old Gen
Thread Stack


Heap
Metaspace
Native
Eden
Survivor
Survivor
Tenured
To
From
Young Gen
Old Gen
Thread Stack


Heap
Metaspace
Native
Eden
Survivor
Survivor
From
To
Tenured
Young Gen
Old Gen
Thread Stack


Heap
Metaspace
Native
Eden
Survivor
From
To
Tenured
Young Gen
Old Gen
Thread Stack


Heap
Metaspace
Native
Eden
Survivor
Tenured
To
From
Young Gen
Old Gen
Thread Stack


Heap
Metaspace
Native
Eden
Survivor
Tenured
To
From
Young Gen
Old Gen
Thread Stack

Avoid STW.

Stop The World
(pausing application threads)

1) Total heap size

2) Ratio of heap for young generation

3) Know how to view and measure collection

Serial

CMS

Throughput

G1

Serial Collector

Don’t use this.

Single thread, small heap application

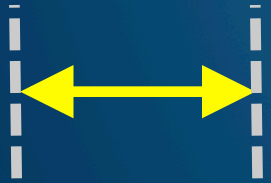
Freezes application threads

Throughput (Parallel) Collector

It's the default.

Uses multiple threads

Pauses on minor / major GC



Supports adaptive sizing

Throughput: The Two Operations

#1 Multiple threads to collect Young Generation

Makes it faster than Serial Collector

#2 Multiple threads to collect Old Generation

CMS: Concurrent Mark Sweep

Uses multiple threads

Designed to eliminate the long pauses from the full GC cycles

Low pause collector

(Pauses on minor GC, execute full GC on background)

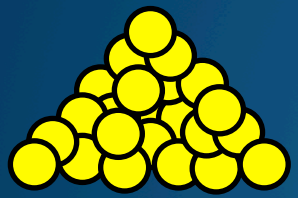
CMS: The Three Operations

#1 Collect the young generation (stopping all application threads)

#2 Run concurrent cycle to clean data out of the old generation

#3 If necessary, perform full GC

G1: Garbage First (introduced in Java 7u4)



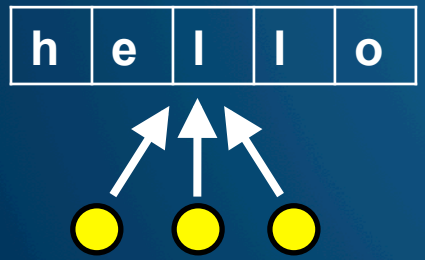
Designed to support large heaps (↑ 4 GB)

Divides heap into regions

(isolate to clean up regions that are mostly garbage)

Duplicate Strings Strings Strings

Strings are generally the largest consumer in your heap (besides their char[] reference)



G1 has support for string de-duplication (different than String.intern())

Java 8 u20

JVM SETTINGS

JVM Args: Garbage Collector

Throughput

`-XX:+UseParallelGC`

Concurrent Mark Sweep

`-XX:+UseConcMarkSweepGC`

`-XX:+UseParNewGC` (diff algorithm to collect young gen)

G1

`-XX:+UseG1GC`

JVM Args: Total Heap Sizing

Minimum and initial heap size

```
-Xms<size>[g|G|m|M|k|K]
```

Maximum heap size

```
-Xmx<size>[g|G|m|M|k|K]
```

```
Xms == Xmx
```

Can be utilized to avoid heap resizing

Helpful when benchmarking specific areas

JVM Args: Adapt

Indicates the max pause time that is tolerable to assist in adaptive sizing (ergonomics)

```
-XX:MaxGCPauseMillis=N
```

String de-duplication (G1 – Java 8 u20+ only)

```
-XX:+UseStringDeduplication
```

JVM Args: Adapt

Enables dumping heap to a file when OutOfMemoryError occurs

```
-XX:-HeapDumpOnOutOfMemoryError
```

Path to dump heap (ex. /tmp)

```
-XX:HeapDumpPath
```

Command to execute on OOME (capture diagnostics and shutdown)

```
-XX:OnOutOfMemoryError="<cmd>;"
```

JVM Args: GC Log

Note: -verbose:gc is NOT necessary when you set –Xloggc (it is implied)

Include more details within your GC log

```
-XX:+PrintGCDetails
```

Have readable date/time strings to correlate to events

```
-XX:+PrintGCDateStamps
```

Specify log file (otherwise will go to stdout)

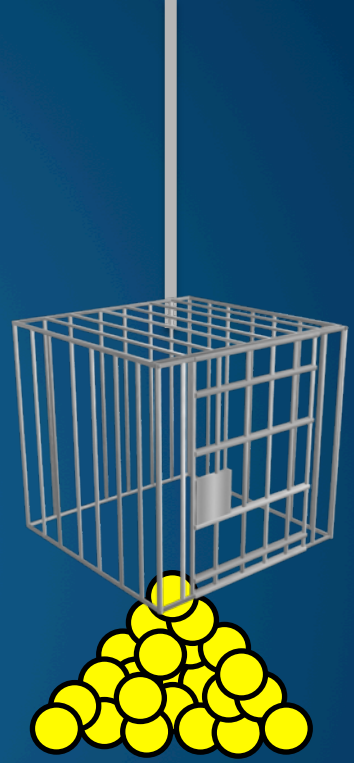
```
-Xloggc:<file>
```

Rotate your GC logs!

```
-XX:GCLogFileSize=50M
-XX:NumberOfGCLogFiles=10
```

```
-XX:+UseGCLogFileRotation
```

CAPTURE A HEAP DUMP



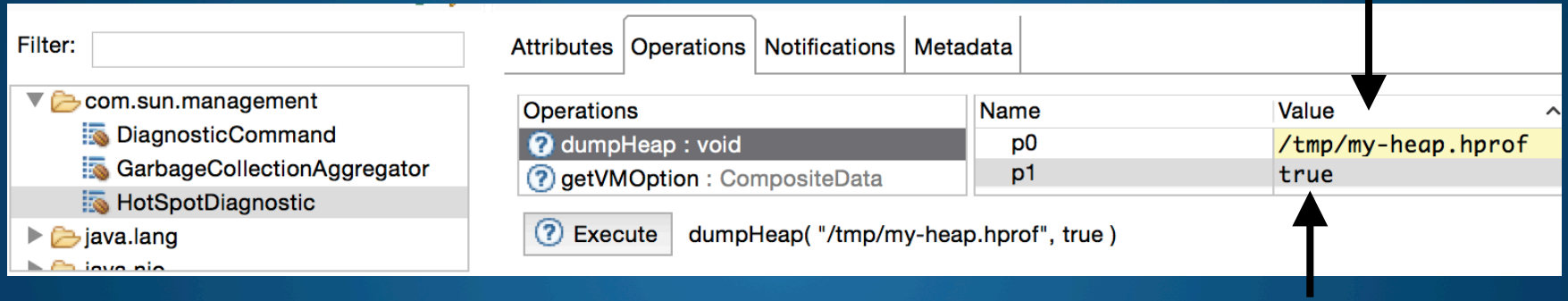
Command Line

```
> jcmd <PID> GC.heap_dump
  hd.hprof

> jmap
  -dump:format=b,file=hd.hprof
  <PID>
```

MBean

File name for the heap dump



True: Only dump live objects

Will create the heap dump with read/write privs for JVM owning user.

Core Dump

```
> sudo gdb --pid=<PID>
> gcore /tmp/jvm.core
> detach
> quit
> jmap
  -dump:format=b,file=hd.hprof
  /usr/bin/java /tmp/jvm.core
```

Java 8 Issue: JDK-8073606
jmap can not get class data for sun/net/ExtendedOptionsImpl$$Lambda

TOOLS

JOverflow

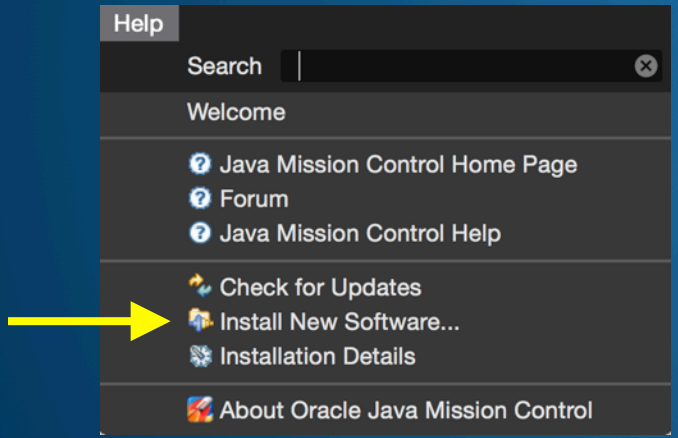
VisualVM

Eclipse Memory Analyzer

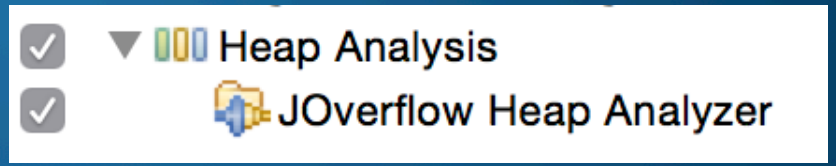
JOverflow (Java Mission Control)

```
> jmc
```

Add plugin to JMC

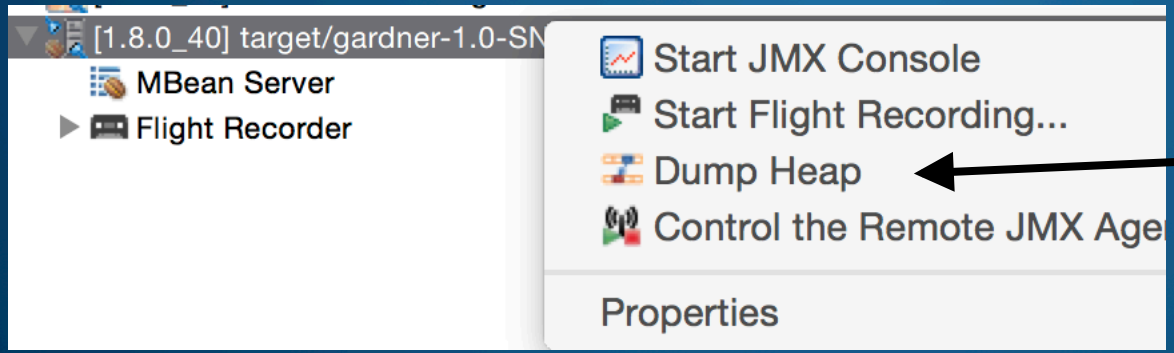


Select and install



JOverflow: Load the heap dump

File → Open File
(select hprof file)



Analyze locally running JVM

JOverflow Analysis

Object Selection	Memory KB	Overhead KB	Objects
All Objects	932,426 (100...	0 (0%)	16,265,498
Duplicate Strings	362,505 (38%)	346,385 (37%)	4,853,598
Arrays with One Element	60,100 (6%)	52,847 (5%)	2,564,281
Duplicate Arrays	47,323 (5%)	46,754 (5%)	1,955,716
Small Collections	47,293 (5%)	32,147 (3%)	599,435
Arrays with Underused Elements	46,118 (4%)	13,117 (1%)	1,949,525
Sparse Large Collections	36,961 (3%)	4,722 (0%)	22,847
Empty Unused Collections	32,764 (3%)	32,764 (3%)	690,425
Boxed Collections	15,119 (1%)	20,606 (2%)	6,127
Sparse Small Collections	4,979 (0%)	508 (0%)	21,794
Empty Arrays	3,270 (0%)	3,270 (0%)	9,493
Long Zero Tail Arrays	727 (0%)	678 (0%)	178
Sparse Arrays	542 (0%)	474 (0%)	1,657

Referrer	Memory KB	Overhead KB	Objects
{ArrayList}	64,355 (6%)	0 (0%)	709,269
BitSet.words	43,547 (4%)	0 (0%)	1,857,925
Unknown GC root	39,682 (4%)	0 (0%)	501,976
com.cerner.aeon.encounter.entities.Location.descrip...	37,084 (3%)	0 (0%)	431,634
com.cerner.aeon.encounter.entities.Location.display	34,981 (3%)	0 (0%)	431,634
{HashMap}	28,379 (3%)	0 (0%)	462,320
com.cerner.aeon.encounter.entities.Location.meaning	24,870 (2%)	0 (0%)	431,634
com.cerner.aeon.encounter.entities.Code.display	22,294 (2%)	0 (0%)	307,953
org.python.core.PyString.string	20,819 (2%)	0 (0%)	96,475
com.cerner.aeon.encounter.entities.Phone.number	15,712 (1%)	0 (0%)	248,799
com.cerner.aeon.encounter.entities.Location.__isset...	15,394 (1%)	0 (0%)	492,613
com.cerner.aeon.encounter.entities.Code.__isset_bit...	14,382 (1%)	0 (0%)	460,238
com.cerner.aeon.encounter.entities.ProviderRelation...	12,947 (1%)	0 (0%)	150,660

Class

Class	Memory ...	Overhead KB	Objects
String	389,576 (4...	0 (0%)	5,017,151
ArrayList	59,865 (6%)	0 (0%)	1,179,967
BitSet	58,060 (6%)	0 (0%)	1,857,925
long[]	44,681 (4%)	0 (0%)	1,861,412
com.cerner.aeon.encounter.e...	30,788 (3%)	0 (0%)	492,613
HashMap	26,111 (2%)	0 (0%)	94,059
com.cerner.aeon.encounter.e...	25,169 (2%)	0 (0%)	460,238
com.cerner.aeon.encounter.e...	17,655 (1%)	0 (0%)	150,660
com.cerner.aeon.encounter.e...	17,494 (1%)	0 (0%)	248,799
HashSet	15,664 (1%)	0 (0%)	7,436
byte[]	14,969 (1%)	0 (0%)	26,532
com.cerner.aeon.encounter.e...	12,697 (1%)	0 (0%)	116,091
com.cerner.aeon.encounter.e...	11,546 (1%)	0 (0%)	492,615

Ancestor referrer

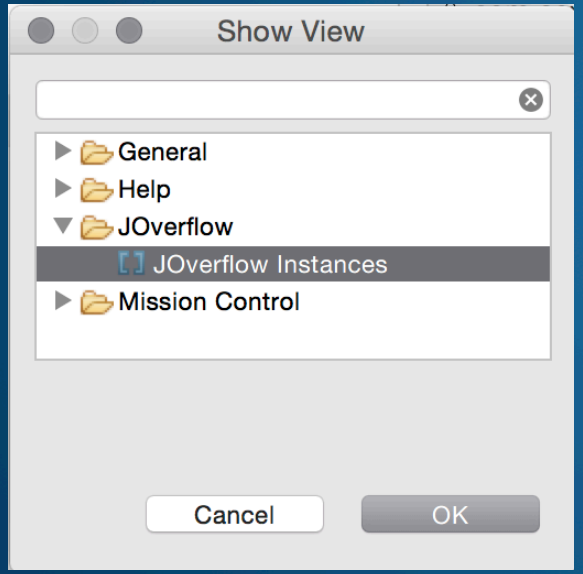
Ancestor prefix

Clear Update

Ancestor referrer	Memory ...	Overhead KB	Objects
{ArrayList}	64,355 (6%)	0 (0%)	709,269
BitSet.words	43,547 (4%)	0 (0%)	1,857,925
N/A	40,708 (4%)	0 (0%)	502,288
com.cerner.aeon.encounter.e...	37,084 (3%)	0 (0%)	431,634
com.cerner.aeon.encounter.e...	34,981 (3%)	0 (0%)	431,634
{HashMap}	28,379 (3%)	0 (0%)	462,320
com.cerner.aeon.encounter.e...	24,870 (2%)	0 (0%)	431,634
com.cerner.aeon.encounter.e...	22,294 (2%)	0 (0%)	307,953
org.python.core.PyString.string	20,819 (2%)	0 (0%)	96,475
com.cerner.aeon.encounter.e...	15,712 (1%)	0 (0%)	248,799
com.cerner.aeon.encounter.e...	15,394 (1%)	0 (0%)	492,613
com.cerner.aeon.encounter.e...	14,382 (1%)	0 (0%)	460,238
com.cerner.aeon.encounter.e...	12,947 (1%)	0 (0%)	150,660

JOverflow: See Instance Data

Window → Show View → Other…



Name	Value	Size
next	null	0
value	com.cerner.aeon.gardner.model.Tool@0x6c0676698	48
url	null	0
repo	null	0
environments	ArrayList@0x6c0676718	32
name	"M+ Kafka Zookeeper Exhibitor"	24
key	"d81aa7ef40a42c77e2b7331720a01cda"	24
hash	1744175625	4

JOverflow Analysis

Object Selection	Memory KB	Overhead KB	Objects
All Objects	932,426 (100...	0 (0%)	16,265,498
Duplicate Strings	362,505 (38%)	346,385 (37%)	4,853,598
Arrays with One Element	60,100 (6%)	52,847 (5%)	2,564,281
Duplicate Arrays	47,323 (5%)	46,754 (5%)	1,955,716
Small Collections	47,293 (5%)	32,147 (3%)	599,435
Arrays with Underused Elements	46,118 (4%)	13,117 (1%)	1,949,525
Sparse Large Collections	36,961 (3%)	4,722 (0%)	22,847
Empty Unused Collections	32,764 (3%)	32,764 (3%)	690,425
Boxed Collections	15,119 (1%)	20,606 (2%)	6,127
Sparse Small Collections	4,979 (0%)	508 (0%)	21,7
Empty Arrays	3,270 (0%)	3,270 (0%)	9,4
Long Zero Tail Arrays	727 (0%)	678 (0%)	1
Sparse Arrays	542 (0%)	474 (0%)	1,6

Referrer	Memory KB	Overhead KB	Objects
{ArrayList}	64,355 (6%)	0 (0%)	709,269
BitSet.words	43,547 (4%)	0 (0%)	1,857,925
Unknown GC root	39,682 (4%)	0 (0%)	501,976
com.cerner.aeon.encounter.entities.Location.descrip...	37,084 (3%)	0 (0%)	431,634
com.cerner.aeon.encounter.entities.Location.display	34,981 (3%)	0 (0%)	431,634
{HashMap}	28,379 (3%)	0 (0%)	462,320
com.cerner.aeon.encounter.entities.Location.meaning	24,870 (2%)	0 (0%)	431,634
com.cerner.aeon.encounter.entities.Code.display	22,294 (2%)	0 (0%)	307,953
org.python.core.PyString.string	20,819 (2%)	0 (0%)	96,475
com.ce...eon.en...ntities.Phone.number	15,712 (1%)	0 (0%)	248,799
...m.c...aeon...counter.a...ies.Location.__isset...	15,394 (1%)	0 (0%)	492,613
...aeon...ncounter.e...es.Code.__isset_bit...	14,382 (1%)	0 (0%)	460,238
...aeon...counter...ies.ProviderRelation...	12,947 (1%)	0 (0%)	150,660

Class

Class	Memory ...	Overhead KB	Objects
String	389,576 (4...	0 (0%)	5,017,151
ArrayList	59,865 (6%)	0 (0%)	1,179,967
BitSet	58,060 (6%)	0 (0%)	1,857,925
long[]	44,681 (4%)	0 (0%)	1,861,412
com.cerner.aeon.encounter.e...	30,788 (3%)	0 (0%)	492,613
HashMap	26,111 (2%)	0 (0%)	94,059
com.cerner.aeon.encounter.e...	25,169 (2%)	0 (0%)	460,238
com.cerner.aeon.encounter.e...	17,655 (1%)	0 (0%)	150,660
com.cerner.aeon.encounter.e...	17,494 (1%)	0 (0%)	248,799
HashSet	15,664 (1%)	0 (0%)	7,436
byte[]	14,969 (1%)	0 (0%)	26,532
com.cerner.aeon.encounter.e...	12,697 (1%)	0 (0%)	116,091
com.cerner.aeon.encounter.e...	11,546 (1%)	0 (0%)	492,615

Ancestor referrer

Ancestor prefix

Clear Update

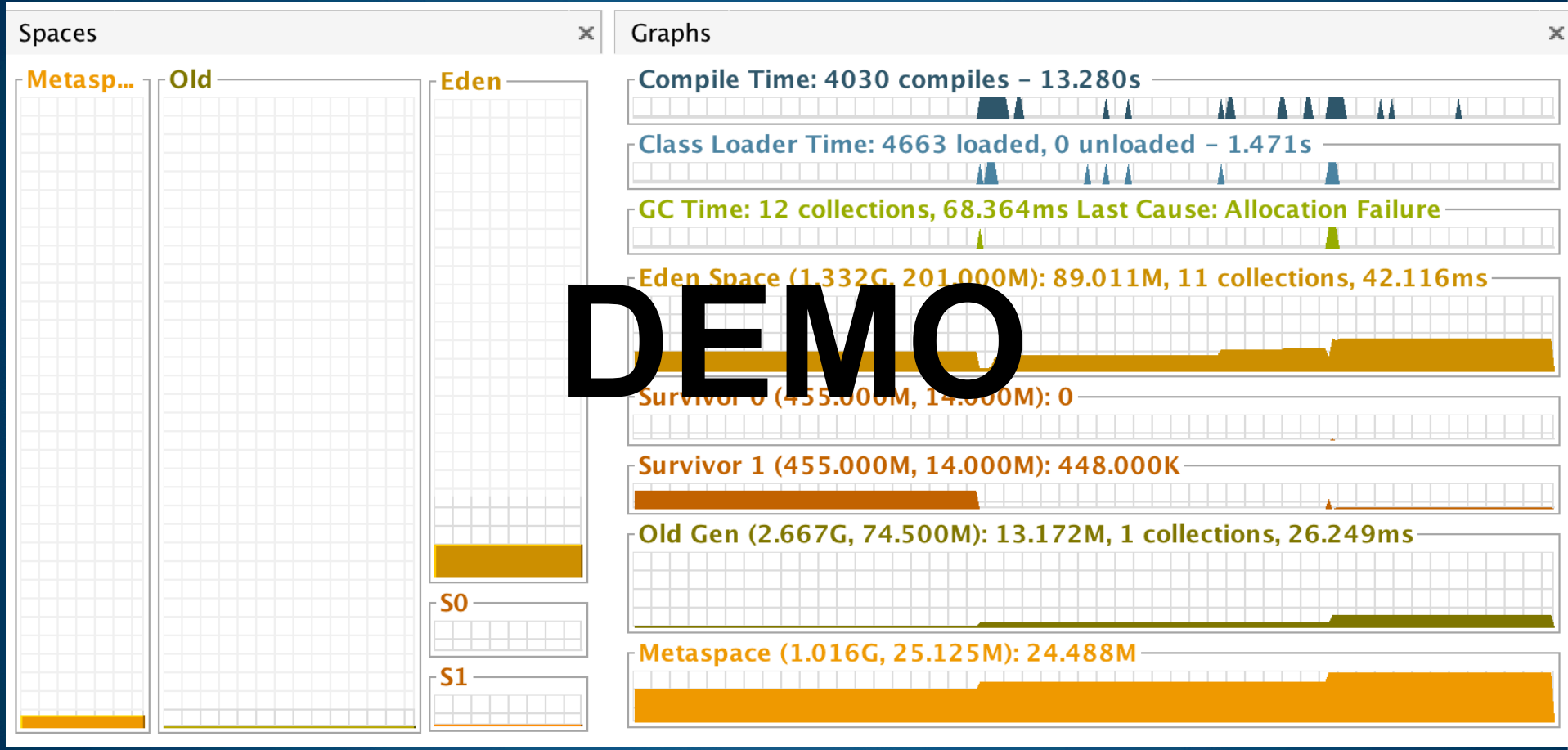
Ancestor referrer	Memory ...	Overhead KB	Objects
{ArrayList}	64,355 (6%)	0 (0%)	709,269
BitSet.words	43,547 (4%)	0 (0%)	1,857,925
N/A	40,708 (4%)	0 (0%)	502,288
com.cerner.aeon.encounter.e...	37,084 (3%)	0 (0%)	431,634
com.cerner.aeon.encounter.e...	34,981 (3%)	0 (0%)	431,634
{HashMap}	28,379 (3%)	0 (0%)	462,320
com.cerner.aeon.encounter.e...	24,870 (2%)	0 (0%)	431,634
com.cerner.aeon.encounter.e...	22,294 (2%)	0 (0%)	307,953
org.python.core.PyString.string	20,819 (2%)	0 (0%)	96,475
com.cerner.aeon.encounter.e...	15,712 (1%)	0 (0%)	248,799
com.cerner.aeon.encounter.e...	15,394 (1%)	0 (0%)	492,613
com.cerner.aeon.encounter.e...	14,382 (1%)	0 (0%)	460,238
com.cerner.aeon.encounter.e...	12,947 (1%)	0 (0%)	150,660

VisualVM

```
> jvisualvm
```

Helpful plugins:

- Visual GC
- * OQL Syntax Support


Spaces
Metasp...
Old
Eden
S0
S1
Graphs
Compile Time: 4030 compiles – 13.280s
Class Loader Time: 4663 loaded, 0 unloaded – 1.471s
GC Time: 12 collections, 68.364ms Last Cause: Allocation Failure
Eden Space (1.332G, 201.000M): 89.011M, 11 collections, 42.116ms
Survivor 0 (455.000M, 14.000M): 0
Survivor 1 (455.000M, 14.000M): 448.000K
Old Gen (2.667G, 74.500M): 13.172M, 1 collections, 26.249ms
Metaspace (1.016G, 25.125M): 24.488M
DEMO

Eclipse Memory Analyzer

Install as:

- Standalone app
- Eclipse Plugin in IDE

Random Tip

Java ThreadLocals can make it helpful to correlate transactional context in a heap dump.


Thread locals
are the gateway
drug to global
memory.

@ 0xe01820a8
NotifyingMemoryStore
net.sf.ehcache.store
class net.sf.ehcache.store.NotifyingMemor…
net.sf.ehcache.store.MemoryStore
sun.misc.Launcher$AppClassLoader @ 0xe…
64 (shallow size)
2,849,504 (retained size)
no GC root

Statics | Attributes | Class Hierarchy | Value

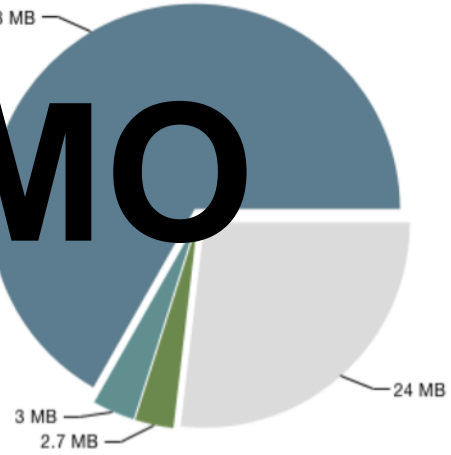
Type	Name	Value
ref	cache	net.sf.ehcache.Cache @ (
boolean	tierPinned	false
ref	lockProvider	null
ref	policy	net.sf.ehcache.store.LruF
ref	status	net.sf.ehcache.Status @ (
int	maximumSize	100000
boolean	elementPin…	true
boolean	cachePinned	false
ref	missRate	net.sf.ehcache.util.ratesta
ref	hitRate	net.sf.ehcache.util.ratesta
ref	poolAccessor	net.sf.ehcache.pool.impl.
ref	map	net.sf.ehcache.store.chm
ref	cache	net.sf.ehcache.Cache @ (
boolean	alwaysPutO…	false
ref	listenerList	null

Overview | Histogram | OQL | Histogram

Details

Size: **89.5 MB** Classes: **10.8k** Objects: **2.1m** Class Loader: **159** Unreachable Objects Histogram

Biggest Objects by Retained Size



DEMO

Total: 89.5 MB

net.sf.ehcache.store.NotifyingMemoryStore @ 0xe01820a8 aeon-concept-tagging-concept-mapping-cache Max Entries:100000 Entries: 1862

Shallow Size: **64 B** Retained Size: **2.7 MB**

Actions

Histogram : Lists number of instances per class

Dominator Tree : List the **biggest objects**

Reports

Leak Suspects : includes leak suspects and a system overview

Top Components : list reports for

Step By Step

Component Report : Analyze objects which belong to a **common root package** or **class loader**.

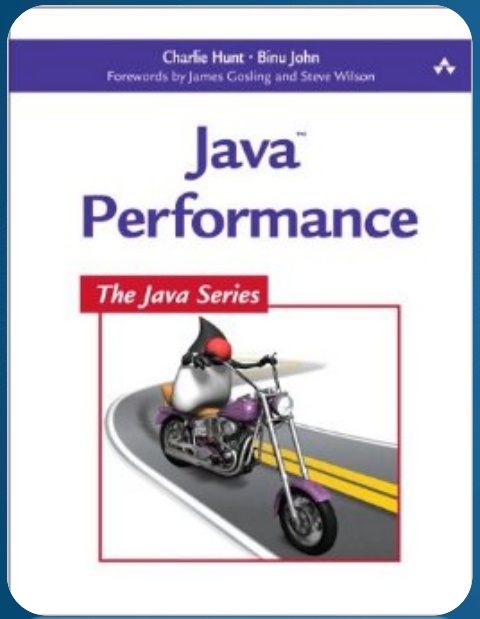
Summarize

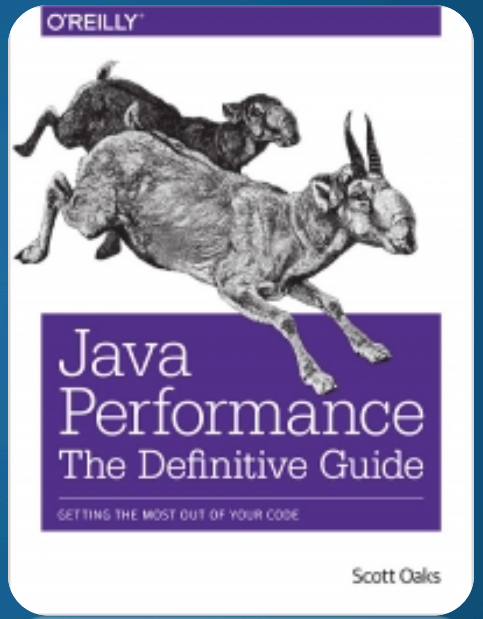
Understand what GC you are using and how it is configured

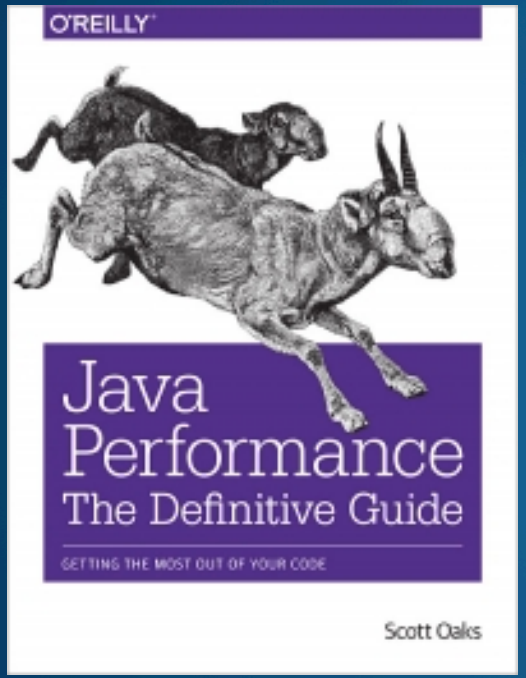
Capture and understand GCs and heap dumps

Know your tools

References


Charlie Hunt · Binu John
Forewords by James Gosling and Steve Wilson
Java™
Performance
The Java Series


O'REILLY
Java
Performance
The Definitive Guide
GETTING THE MOST OUT OF YOUR CODE
Scott Oaks



"Short of rewriting code, tuning the garbage collector is the most important thing that can be done to improve the performance of a Java application." – Scott Oaks