Problem Determination
Java Heapdumps and OOMs

May 21, 2019
Kevin Grigorenko
IBM WAS SWAT Team
kevin.grigorenko@us.ibm.com

Agenda

- IBM Memory Analyzer Tool
- Heapdump Theory
- OutOfMemory Analysis
- Object Query Language
- IBM Extensions for Memory Analyzer
- Interactive Diagnostic Data Explorer (IDDE)

Memory Analyzer Tool (MAT)

- MAT is an open source project originally donated to Eclipse by SAP. It was designed for memory leak detection and footprint analysis, but is now used more broadly.
- **Resources:**
  - MAT Project Website: http://eclipse.org/mat/
  - Standalone download (32- or 64-bit) or update site for Eclipse/RAD: http://eclipse.org/mat/downloads.php
  - MAT Forum: http://www.eclipse.org/forums/index.php?t=thread&frm_id=186
  - Reporting a bug: https://bugs.eclipse.org/bugs/enter_bug.cgi?product=MAT
  - Source code (EPL License, instructions): http://dev.eclipse.org/svnroot/tools/org.eclipse.mat/trunk/

IBM & MAT

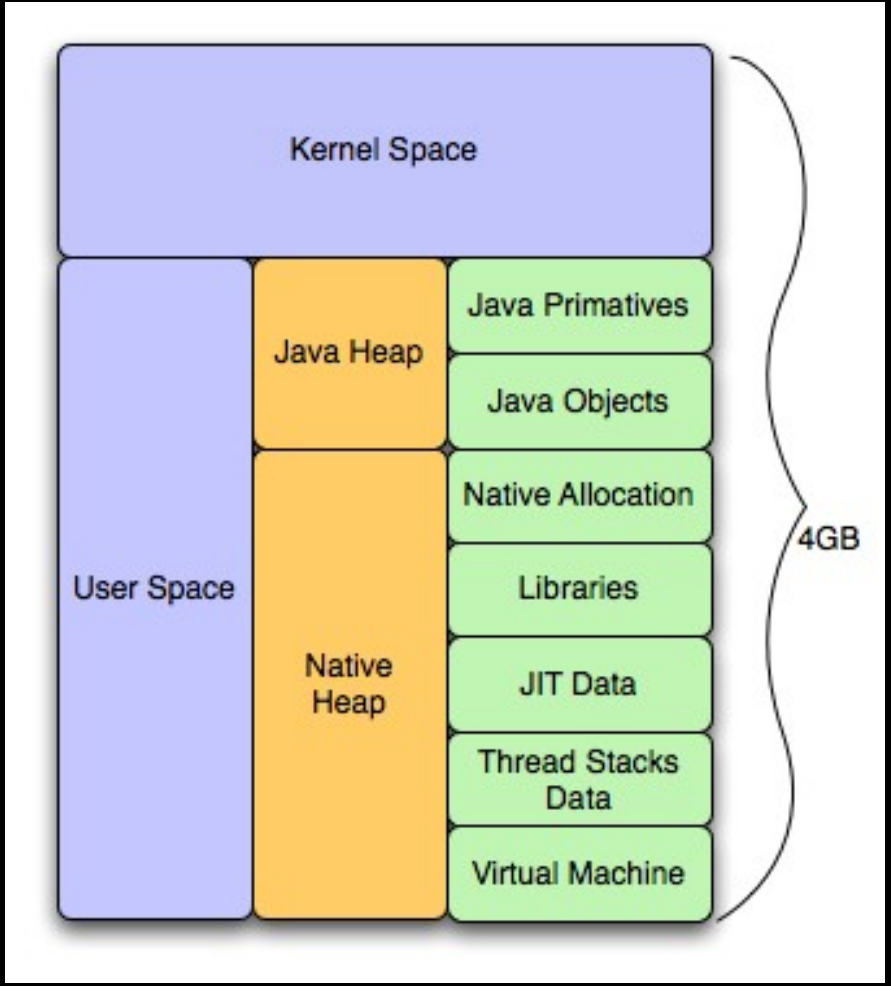
- MAT initially only supported HPROF heapdumps from HotSpot (Sun/Oracle) based JVMs.
- The IBM Java team became a participant in the Eclipse MAT open source project and added support for IBM JVM based dumps (PHD and system dump), however this requires installing the IBM DTFJ plugin adapter.
- IBM ships a version of MAT which already includes this plugin. This is called the IBM Monitoring and Diagnostic Tools for Java – Memory Analyzer Tool.
- This tool is available in the IBM Support Assistant (ISA) and fully supported by IBM (i.e. you can open a PMR on bugs with the tool).
- The ISA platform only runs in 32-bit mode. The best way to get around this:
  - Download the standalone MAT build from eclipse.org: http://eclipse.org/mat/downloads.php
  - Click Help > Install New Software > In the "Work with:" textbox at the top, paste: http://download.boulder.ibm.com/ibmdl/pub/software/isa/isa410/production/ And press Enter (This will take some time to load)
  - Install label.component.tools.jvm > Diagnostic Tool Framework for Java
  - While you're there, also check all the IBM Extensions for Memory Analyzer* plugins

IBM Extensions for Memory Analyzer (IEMA)

- IBM provides the free IBM Extensions for Memory Analyzer (IEMA) that provide product specific knowledge:
  - Generic Java applications
  - WebSphere Application Server
  - WebSphere eXtreme Scale
  - CICS Transaction Gateway
  - ...and more to come
- IEMA also provides “always on” extensions and status reports of aspects of the applications.
- IEMA is covered later in the slides.

Heapdumps, System dumps, Core dumps

- A system dump, or core dump, is a file created by an operating system which is a snapshot of the entire address space
  - Javacores should be referred to as javadumps to avoid confusion with system core dumps
- A heapdump is a file created by the JVM which is a snapshot of the Java heap
- An IBM heapdump does not contain memory contents (Strings, integers, variable names, etc.), but a system dump (or HPROF dump) does



Comparison of Dump Data Availability

Dump Format	Approx. Size on Disk	Objects, Classes, Class Loaders	Thread Details	Field Names	Field and Array Refs	Primitive Fields	Primitive Array Contents	Accurate GC Roots	Native Memory and Threads
IBM PHD	20% of Java heap size	✓	with Javacore	✗	✓	✗	✗	✗	✗
HPROF	Java heap size	✓	✓	✓	✓	✓	✓	✓	✗
System dump	Java heap size + 30%	✓	✓	✓	✓	✓	✓	✓	✓

Getting a Portable Heapdump (PHD - IBM JVM)

- Automatically produced on OOM (up to 4 times)
  - Control with -Xdump or -Xtrace
    - http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/topic/com.ibm.java.doc.diagnostics.60/diag/tools/dumpagents_syntax.html
    - -Xdump:heap:events=systhrow,filter=java/lang/OutOfMemoryError,range=1..4,request=exclusive+compact+prepwalk
- Use -Xdump:heap:user to take one on kill -3/Ctrl+Break
- System dump → jextract → jdmpview → heapdump
- Wsadmin (Mbean has a limit, can be increased):
  - AdminControl.invoke(AdminControl.completeObjectName("type=JVM,process=server1,*"), "generateHeapDump")
- Programmatically with com.ibm.jvm.Dump.HeapDump()
- On Java < 5 with JAVA_DUMP_OPTS envar
- From within MAT → File → Acquire Heap Dump

Getting a Heapdump (HPROF - Oracle JVM)

- Automatically produced on OOM with -XX:+HeapDumpOnOutOfMemoryError
  - http://www-01.ibm.com/support/docview.wss?uid=swg21242314
  - On recent releases, one heapdump per JVM run; previously, no limit.
- Ctrl+Break or kill -3 with -XX:+HeapDumpOnCtrlBreak
- Java 5: jmap -dump:format=b
- Java 6: jmap -dumpformat=b,file=<filename> <pid>
- Jconsole with HotSpotDiagnostic Mbean dumpHeap
- System dump (e.g. gcore) and extract with jmap
- http://wiki.eclipse.org/index.php/MemoryAnalyzer#Getting_a_Heap_Dump

Getting a System Dump

- Ensure proper ulimits! AIX, Linux
- Automatically produced on a crash
- Create a system dump on OOM instead of phd:
  - -Xdump:heap:none
    -Xdump:java+system:events=systhrow,filter=java/lang/OutOfMemoryError,range=1..4,request=exclusive+prepwalk
- Wsadmin:
  AdminControl.invoke(AdminControl.completeObjectName("type=JVM,process=server1,*"), "generateSystemDump")
- Programmatically with com.ibm.jvm.Dump.SystemDump()
- AIX=gencore, Linux=gcore, z/OS=SVCDUMP, Windows=userdump.exe
- Then run: <WAS>/java/jre/bin/jextract $DUMP and load the ZIP in MAT
- System dumps usually compress to 25% of original size.

Getting a System Dump (Continued)

- IBM Health Center can acquire a dump
- The trace engine allows system and PHD dumps to be triggered on method entry or exit. This produces a system dump when the Example.trigger() method is called
  - -Xtrace:maximal=mt,trigger=method{com/ibm/example/Example.trigger,sysdump}
- Set a range to take dumps between the first and 5th method invocations:
  - -Xtrace:maximal=mt,trigger=method{com/ibm/example/Example.trigger,sysdump,,5,1}
- Jextract is no longer needed with DDR in Java 5 >= SR12 (WAS >= 6.1.0.33), Java 6 >= SR9 (WAS >= 7.0.0.15), Java 626 (WAS 8)
- The strategic direction is system dumps. In WAS 8.0.0.2, one is created on the first OOM.
- Ensure enough physical memory for best performance.

Getting a System Dump (Linux)

- “Linux does not provide an operating system API for generating a system dump from a running process. The JVM produces system dumps on Linux by using the fork() API to start an identical process to the parent JVM process. The JVM then generates a SIGSEGV signal in the child process. The SIGSEGV signal causes Linux to create a system dump for the child process. The parent JVM processes and renames the system dump, as required, by the -Xdump options, and might add additional data into the dump file.

  The system dump for the child process contains an exact copy of the memory areas used in the parent. The SDK dump viewer can obtain information about the Java threads, classes, and heap from the system dump. However, the dump viewer, and other system dump debuggers show only the single native thread that was running in the child process.”
  - http://publib.boulder.ibm.com/infocenter/java7sdk/v7r0/topic/com.ibm.java.lnx.70.doc/diag/tools/dumpagents_platform_nonzos.html

Getting a System Dump (Linux)

- The Linux kernel.core_pattern setting (available in Linux 2.5 and later kernels) can be used to specify the name and path for system dumps.
  - However, this may interfere with the JVM's core naming scheme.
- When getting a system dump manually on Linux using the gcore command, by default, the produced core will be named core.$PID.
- Because the core was not created by the JVM itself, MAT will not know when the core was created. Therefore, it is recommended to rename the core with the timestamp in the name:
  -
```
#!/bin/sh
PID=$1
SEQ=$2
PREFIX=$3
if [ -z "$PREFIX" ]; then
  PREFIX="./"
fi
if [ -z "$SEQ" ]; then
  SEQ=1
fi
COREFILE="${PREFIX}core.`date +%Y%m%d.%H%M%S`.$PID.000$SEQ.dmp"
gcore -o $COREFILE $PID
echo "Renaming core file. Your process has now continued running."
# gcore adds the PID to the end of the file, so just remove that
mv $COREFILE.$PID $COREFILE
```
-

-Xdump Agents

Event	Description	Filtering	Example
gpf	GPF (Crash)		-Xdump:system:events=gpf
user	User generated signal (SIGQUIT or Ctrl-Break)		-Xdump:system:events=user
vmstop	VM shutdown, including call to System.exit()	exit code	-Xdump:system:events=vmstop,filter=#0..#10
load	Class load	class name	-Xdump:system:events=load,filter=com/ibm/example/Example
unload	Class unload	class name	-Xdump:system:events=unload,filter=com/ibm/example/Example
throw	An exception being thrown	exception name	-Xdump:system:events=throw,filter=java/net/ConnectException
catch	An exception being caught	exception name	-Xdump:system:events=catch,filter=java/net/ConnectException
systhrow	A Java exception is about to be thrown by the JVM	exception name	-Xdump:system:events=systhrow, filter=java/lang/OutOfMemoryError,range=1..4
allocation	A Java object is allocated	size of object	-Xdump:system:events=allocate,filter=#5m

- Exceptions can also be filtered on throwing method using '#'
  - -Xdump:system:events=throw,filter=java/lang/NullPointerException#com/ibm/example/Example.bad

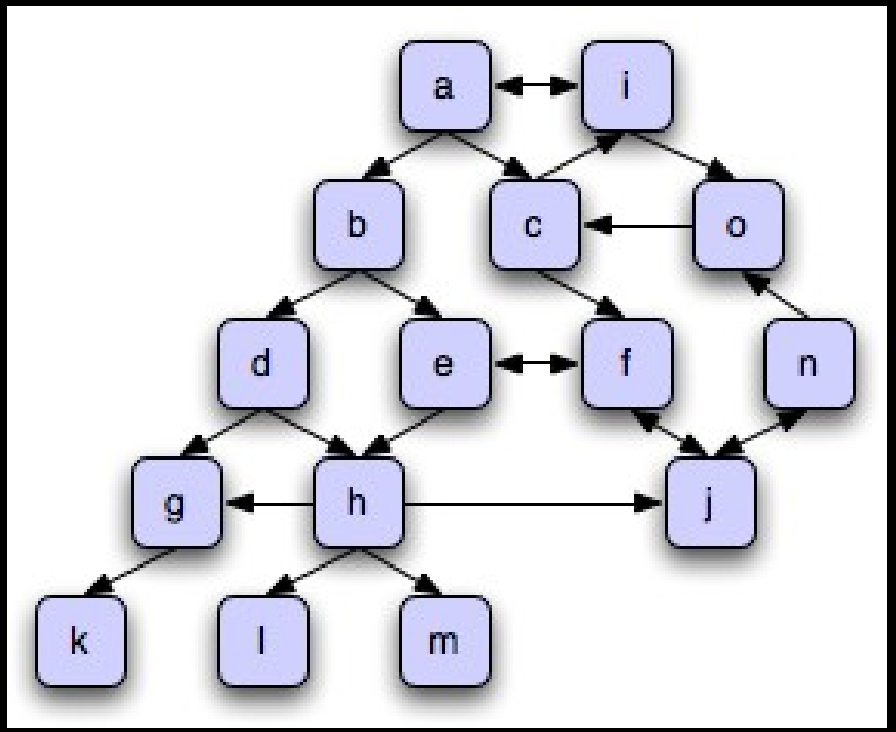
What caused an OutOfMemoryError

- Verbose garbage collection is critical and should be enabled on all production systems (less than 1% overhead):
  - https://www.ibm.com/developerworks/mydeveloperworks/blogs/troubleshootingjava/entry/verbose_gc_performance?lang=en
- An OutOfMemoryError may be caused by native heap exhaustion. That's outside the scope of this presentation, but, at minimum, look for a reason message in the javacore/logs, and see if the top frame of the current thread is in native code.
  - MAT may still be very useful in the case of a native OOM.
- If the OOM is caused by a large object allocation, that allocation won't be in the heapdump!

Heapdump Theory

Heapdump Theory

- The Java heap is a directed graph of references (digraph)
  - Each reference may have primitives (integers, longs, doubles, etc.) which can be seen in Windows → Inspector and click on an object
- Incoming references can be thought of as “parents” and outgoing references as “children”
  - The reason these terms aren't used is because a child can point back to a parent, directly or indirectly.

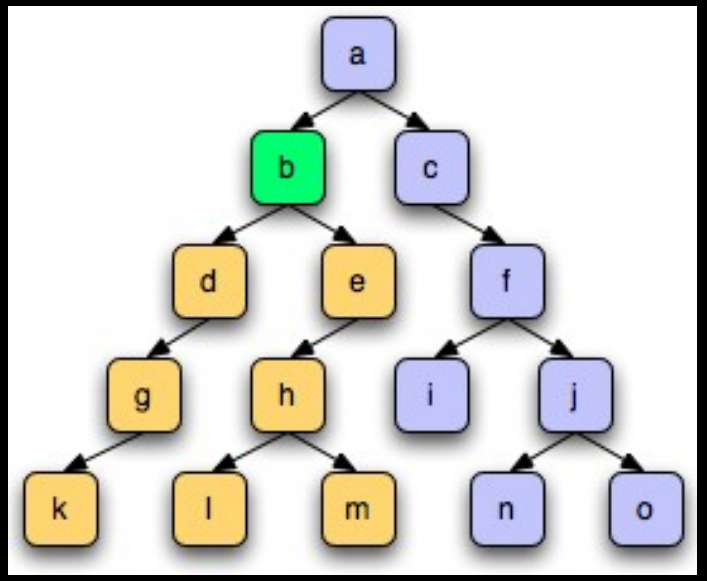


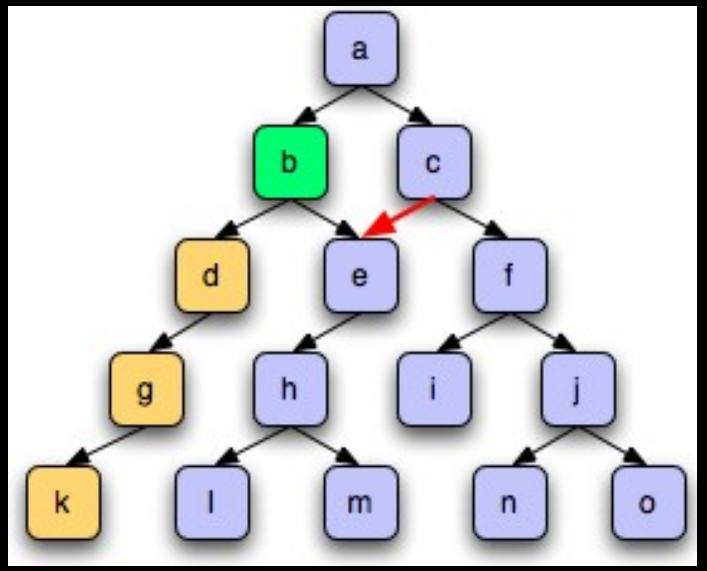
Heapdump Theory

- Shallow heap is the size of an Object and its primitives.
- Retained heap is the shallow heap plus the retained heaps of lifetime-dependent outgoing references.
- Object address may change if moved around by GC
- The dominator X of an object Y is the "root" object that retains Y
- The dominator tree is the heap split into mutually exclusive dominators
- A “GC Root” is an object which has a reference to it from outside the Java heap.
  - e.g. native threads, registers, JNI, stack objects (locals)
  - http://wiki.eclipse.org/index.php/MemoryAnalyzer#Garbage_Collection_Roots

Retained Sets

- Retained set demonstration
  - Top diagram: The green object (B) "retains" the orange objects. The orange objects are lifetime-dependent on B.
  - Bottom diagram: Introduce C which references E. Now, B's retained set/size has been reduced to D, G, K. If B was Gced, E, H, L, M would only be Gced if C was too.



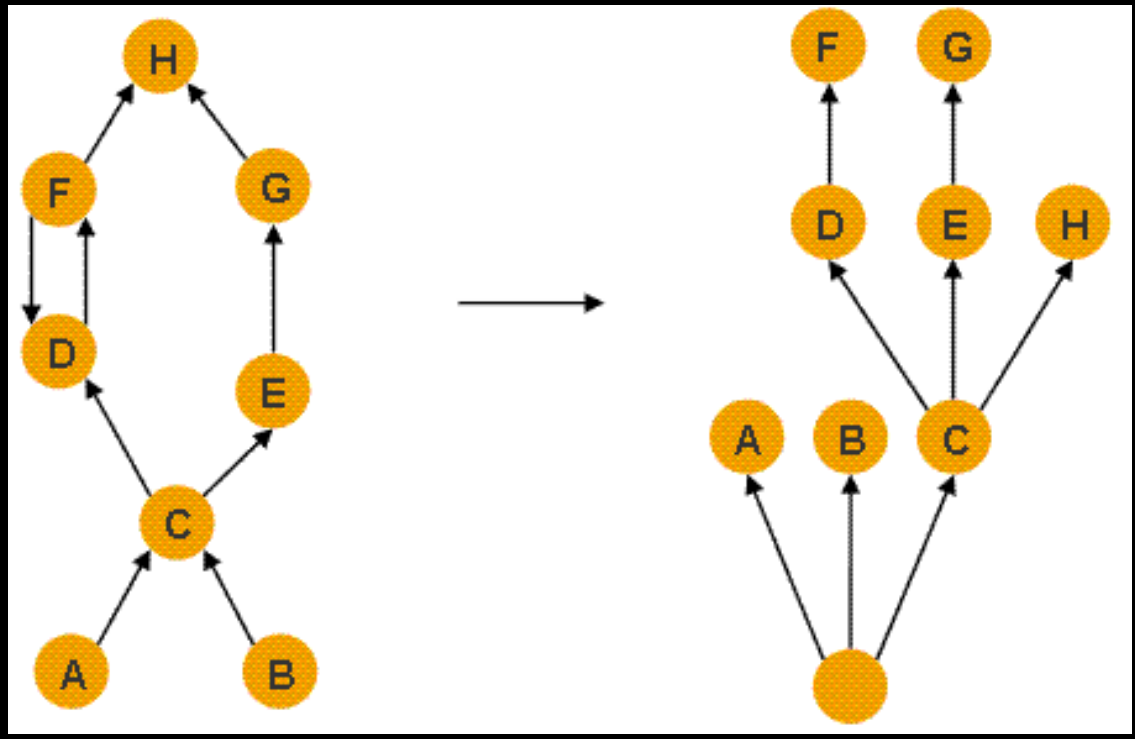


Retained Sets (Continued)

- Customized Retained Set
  - You can create a customized retained set by class using Open Query Browser > Show Retained Set OR Java Basics > Customized Retained Set. For example, if you want to know how much total heap is consumed by some class X, then just pass that to the query. The sum of shallow heaps is the amount held by the class.
  - In the previous example, if you want to see the retained set of B **if** C didn't have that reference to E, then you can use the exclude (-x) option to do this.
    - This is useful if an object is "watching" another object and you know that B is the "primary" object.

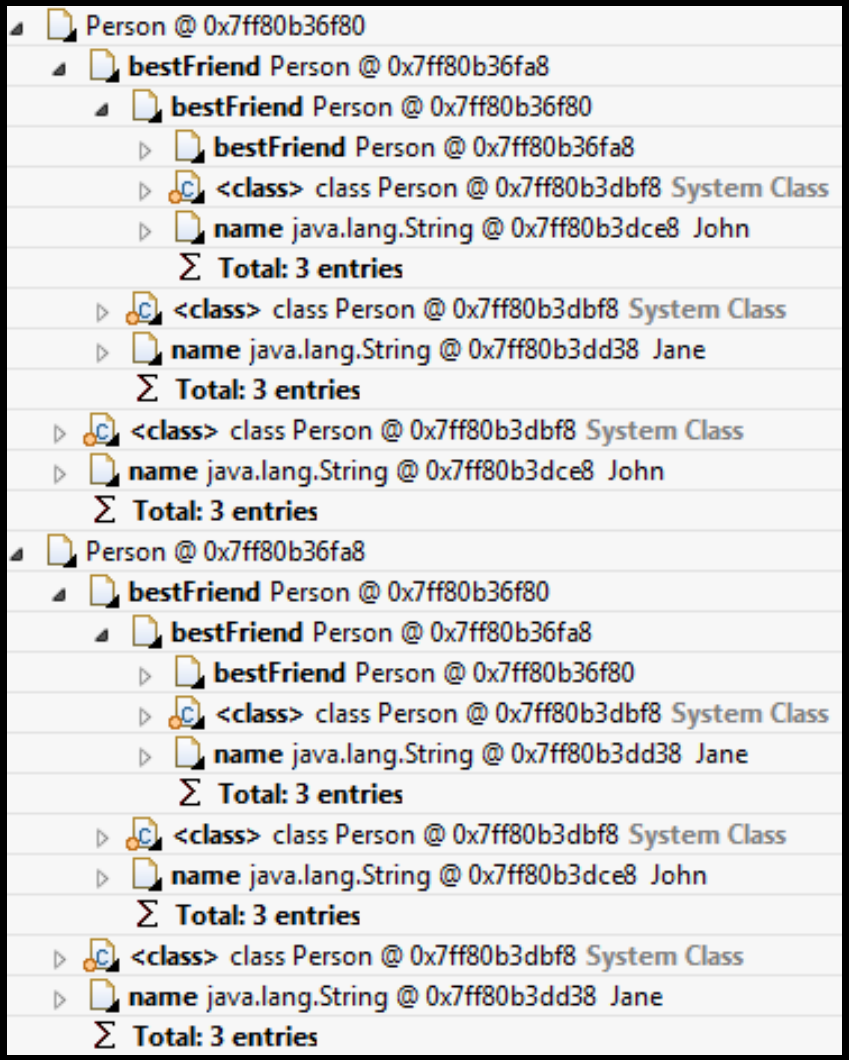
Dominator Tree

- Transform object graph to identify the biggest chunks of retained memory and the keep-alive dependencies among objects.
- An object x dominates an object y if every path in the object graph from the start (or the root) node to y must go through x.
- The immediate dominator x of some object y is the dominator closest to the object y.
- A dominator tree is built out of the object graph. In the dominator tree each object is the immediate dominator of its children, so dependencies between the objects are easily identified.
- The objects belonging to the sub-tree of x (i.e. the objects dominated by x) represent the retained set of x.
- If x is the immediate dominator of y , then the immediate dominator of x also dominates y , and so on.
- The edges in the dominator tree do not directly correspond to object references from the object graph.



Finding Objects

- Open Query Browser > List Objects > with incoming| outgoing references
- Specify a class to get all instances, or a particular object address
- The arrow decorator in a view will show whether references are incoming or outgoing.
- If the name is prefixed with “class “ then that is the static instance of that class (1 per classloader)
  - In general, do not follow this down when navigating outgoing references (unless it's very large in which case there might be a static cache)
- “Show objects by class” lets you group references by class.
  - For example, from the histogram, if class X is the biggest, right click and Show Objects by class → by incoming references will show which objects (grouped by class) reference instances of class X
- Watch out for loops! (Check the address)



Tips & Tricks

Background Information

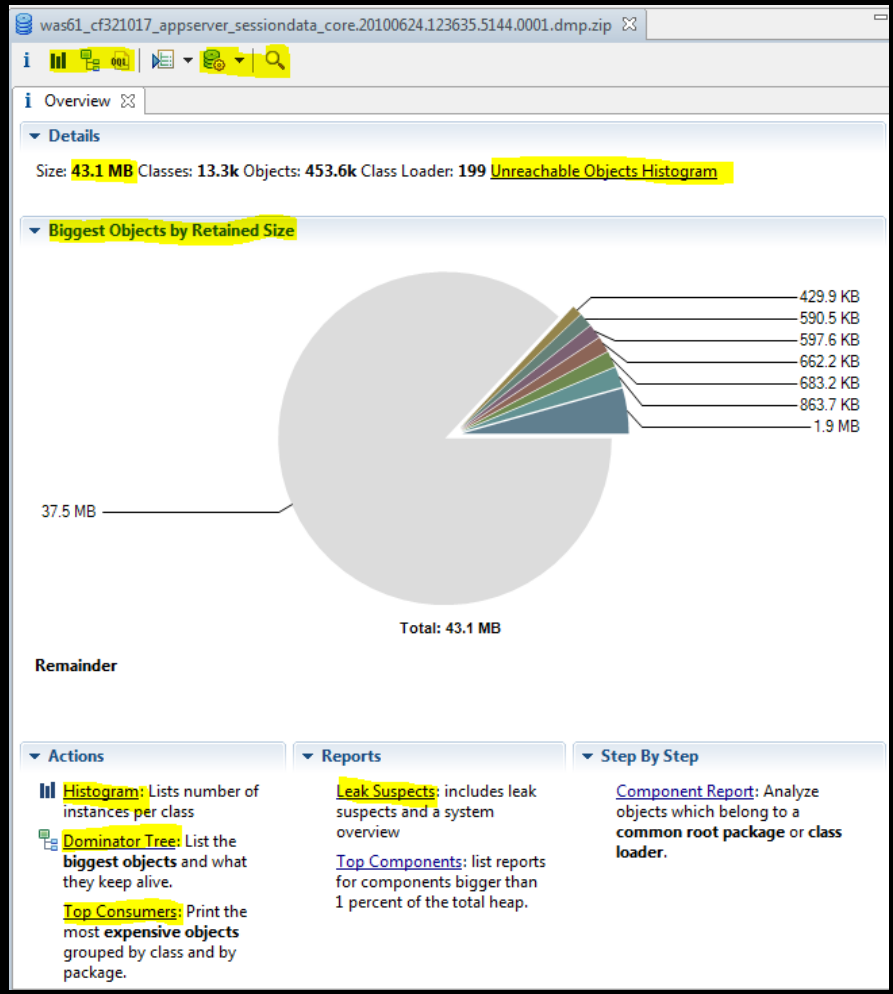
- For Standalone MAT, set JVM parameters (e.g. -Xmx) in MemoryAnalyzer.ini
  - For ISA max heap: http://www-01.ibm.com/support/docview.wss?uid=swg21403571
- For HPROF, thread Stacks are not available until Java 6 Update >= 14 and Java 7
- “Unreachable objects” are objects that are eligible for garbage collection
  - In MAT, see a histogram of these by clicking “Unreachable Objects Histogram”
- Some reports can be exported as HTML using a button at the top (e.g. to send to developers)

Background Information

- MAT writes “swap” files into the same directory as the heapdump so that it doesn't have to load the whole heapdump
  - There are limits to this and so you may still get OutOfMemoryErrors loading heapdumps. If so, use 64 bit.
  - These also make reloading a heapdump very fast. You can delete them but you will lose the fast reload.
- The first row of a table result set allows filters
  - Just type something in, it will automatically surround with .*
- Compare two dumps by loading both, clicking histogram on the newer one then clicking the Compare button at the top right

First Steps in MAT

- In the “Details” section, note “Size” which is the size of the live Java heap at the time of the heapdump.
  - If it's not making sense (to verbosegc), click Unreachable Objects Histogram and note the Total in the “Shallow Heap” column- this amount would be GCed if it could.
    - For example, a lot of garbage in tenured and no Full GCs for a while.
- In the “Biggest Objects by Retained Size” section, the pie chart represents the top hitters from the Dominator Tree report.
  - Left click on a pie portion and List Objects → with outgoing references to see objects held by the dominator.

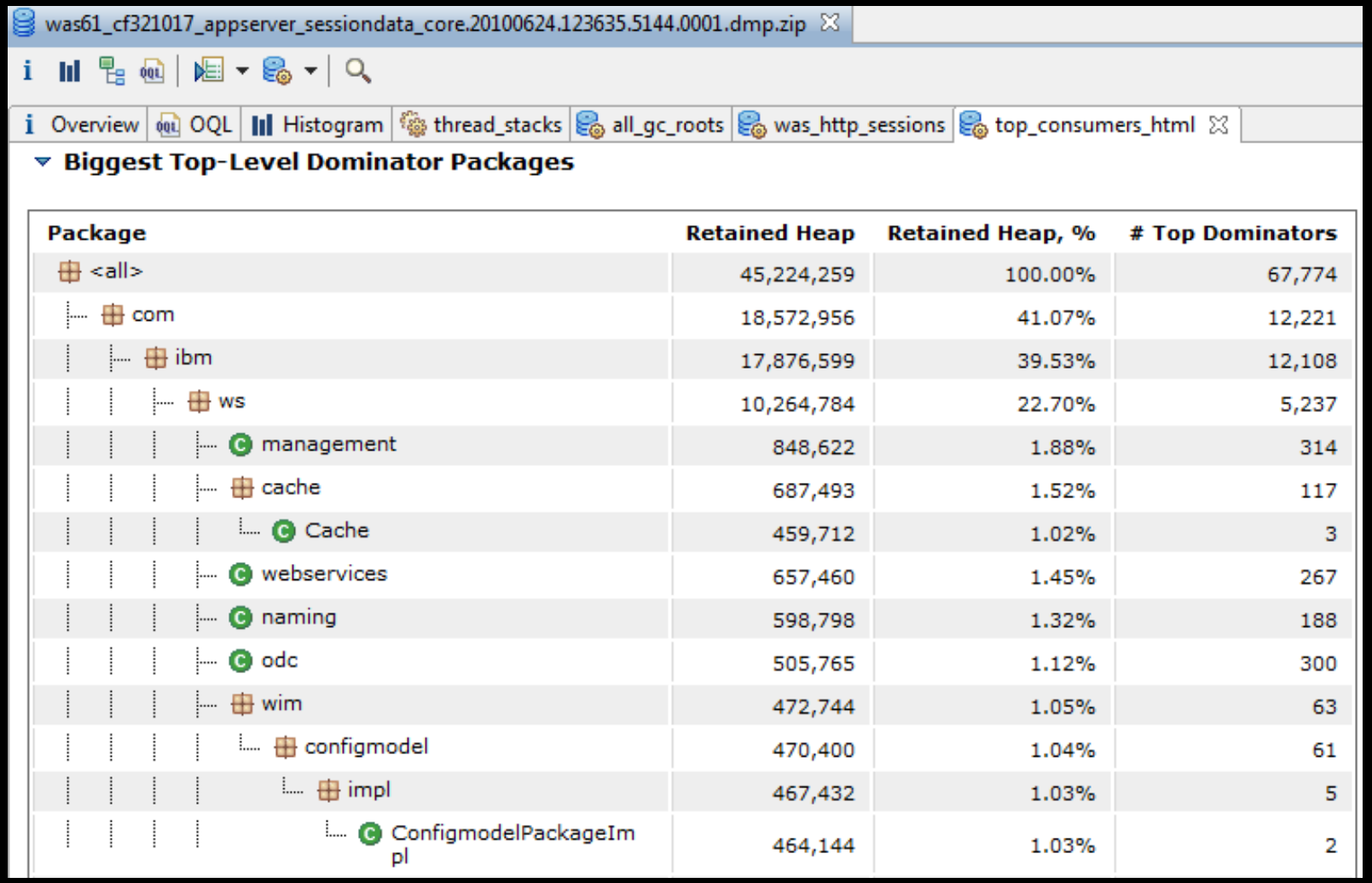


First Steps in MAT

- In the “Reports” section, click “Leak Suspects”
- In the “Actions” section:
  - Click Histogram
    - Gives you what is taking up the heap by class
  - Click Dominator Tree
    - Gives you what “large” objects are taking up the heap
  - Click Top Consumers
    - Scroll down to the “Biggest Top-Level Dominator Packages” section
    - Gives you what is taking up the heap by package
- Open Query Browser > Leak Identification > Big Drops in Dominator Tree

Top Consumers

- Groups the dominator tree by package



Package	Retained Heap	Retained Heap, %	# Top Dominators
<all>	45,224,259	100.00%	67,774
com	18,572,956	41.07%	12,221
ibm	17,876,599	39.53%	12,108
ws	10,264,784	22.70%	5,237
management	848,622	1.88%	314
cache	687,493	1.52%	117
Cache	459,712	1.02%	3
webservices	657,460	1.45%	267
naming	598,798	1.32%	188
odc	505,765	1.12%	300
wim	472,744	1.05%	63
configmodel	470,400	1.04%	61
impl	467,432	1.03%	5
ConfigmodelPackageImpl	464,144	1.03%	2

Tips

- There is no easy way to get the object's generation
- When viewing a String, MAT doesn't show very large ones. Right click → Copy → Save value to file
- Strings are actually Java classes that have 1 outgoing reference to a primitive array of chars
- Table results have a useful Export button (e.g. CSV)
- Running from a script (headless mode):
  - java -Xmx3g -jar .../plugins/org.eclipse.equinox.launcher*.jar -consoleLog -application org.eclipse.mat.api.parse mydump.dmp org.eclipse.mat.api:suspects org.eclipse.mat.api:overview org.eclipse.mat.api:top_components

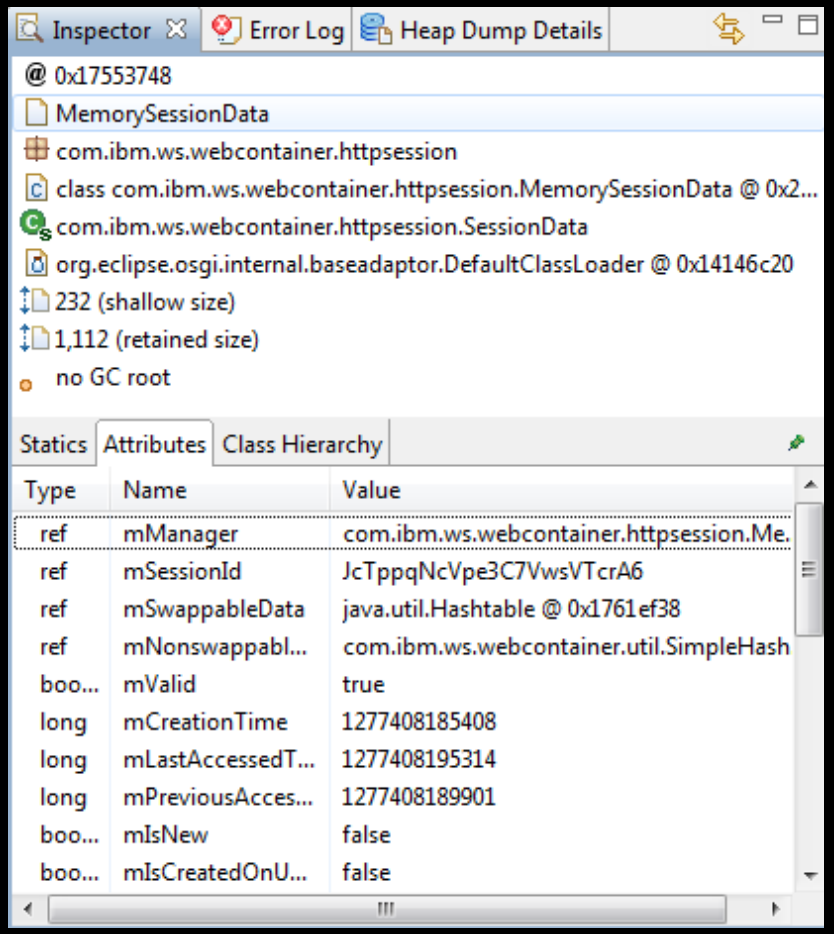
Sizing Applications

- No straightforward method because objects can be strewn throughout the heap (e.g. sessions in the session manager, caches, etc.)
- Dominators by class loader with a system dump is a good start
- IEMA provides the WAS Overview which finds all application classloaders' usage

Debugging with System Dumps

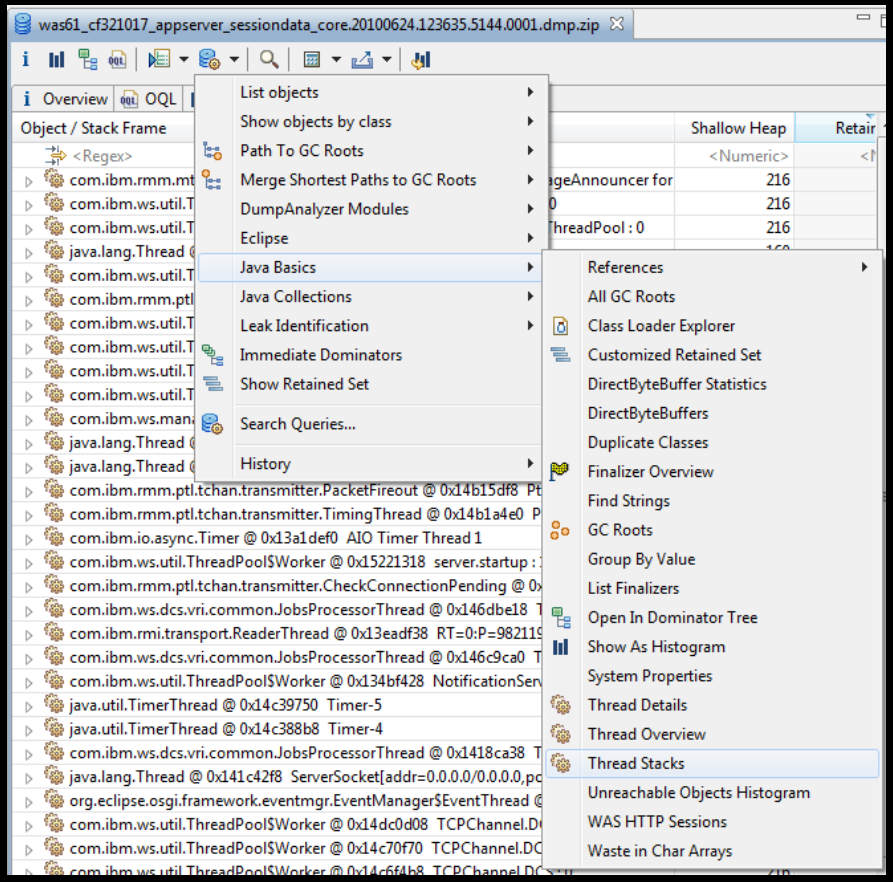
Inspector

- Window > Inspector
- The object will change when you click on something, or sometimes even if you just hover over an object



Thread Stacks and Frame Locals

- Java Basics > Thread Stacks and press OK
- Expand a thread and expand a stack. Any local object references on the frame will show up!



Object Query Language

Object Query Language

- Object Query Language (OQL) button at the top
  - Similar to SQL. Hit F1 for decent help and examples.
- Get an outgoing references tree of all instances of $CLASS (with an optional condition)
  - SELECT * FROM INSTANCEOF <CLASS> WHERE …
    - Use INSTANCEOF to include subclasses
- OQL is most useful with system dumps because you can reference the fields
  - SELECT
    dbb,
    dbb.capacity,
    snapshot.getObject(inbounds(dbb)[0]),
    snapshot.getObject(inbounds(dbb)[1])
    FROM INSTANCEOF java.nio.DirectByteBuffer dbb
    WHERE
    (
    (dbb.viewedBuffer=null) AND
    (dbb.att=null)and(inbounds(dbb).length>1)
)

OQL (Continued)

- The field reference can be nested. For example, let's say you have class X which has an object reference (named obj) to class Y which has an integer field called size. Let's say you want to find all instances of X with size greater than 10:
  - SELECT * from com.package.X x where x.obj.size > 10
  - This uses the concept of an alias, x, given to the class
- If you select a reference object, e.g.:
  - SELECT x.obj from com.package.X x
  - Then you need OBJECTS in front to get the normal view:
  - SELECT OBJECTS x.obj from com.package.X x

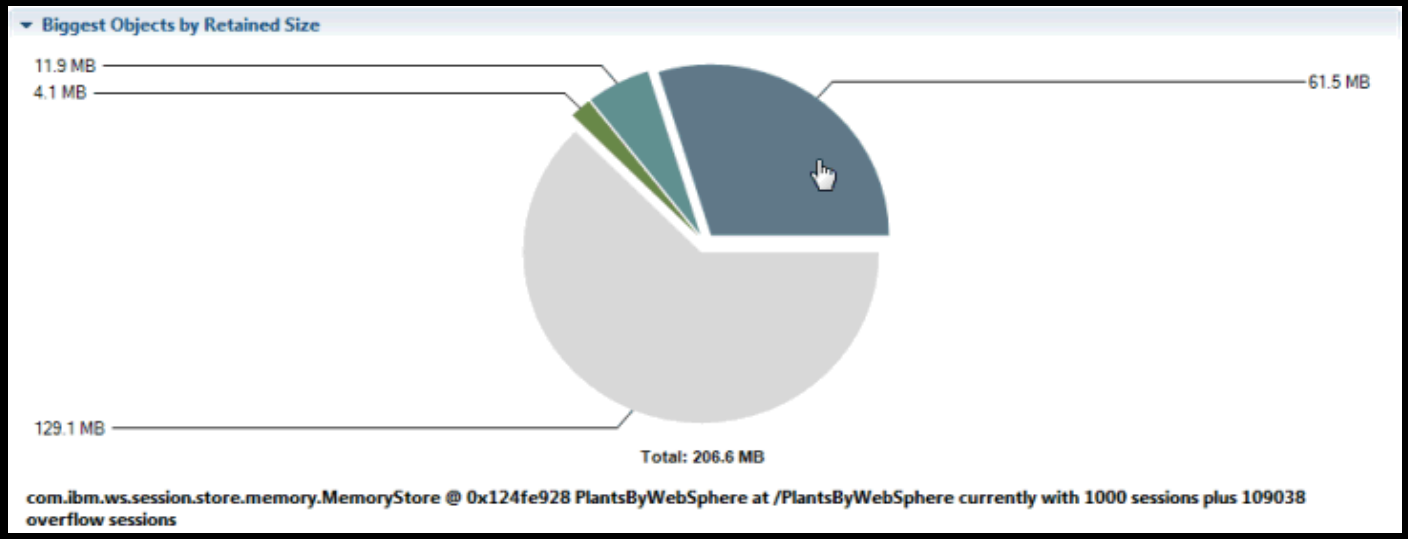
OQL (Continued)

- Displaying String contents using toString()
  - SELECT toString(t.name) FROM INSTANCEOF java.lang.Thread t
  - SELECT * FROM INSTANCEOF java.lang.Thread t WHERE (toString(t.name) = "Thread-1")
- Displaying particular columns, including "built ins":
  - SELECT t.@displayName, t.@retainedHeapSize AS "Retained Size" FROM INSTANCEOF java.lang.Thread t WHERE (toString(t.name) = "Thread-1")
- Other Interesting Functions: dominators(), outbounds(), inbounds(), dominatorof()

IBM Extensions for Memory Analyzer

Always On Extensions

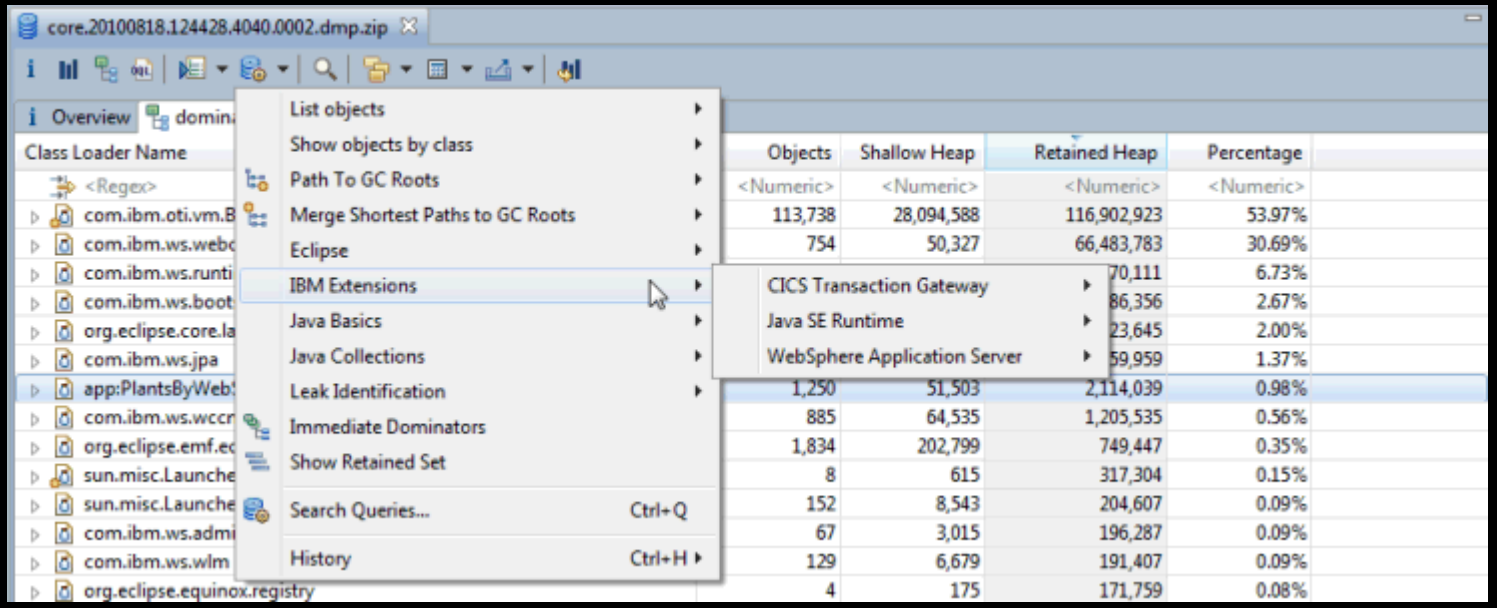
- Top right: MemoryStore object shows how many sessions and what app



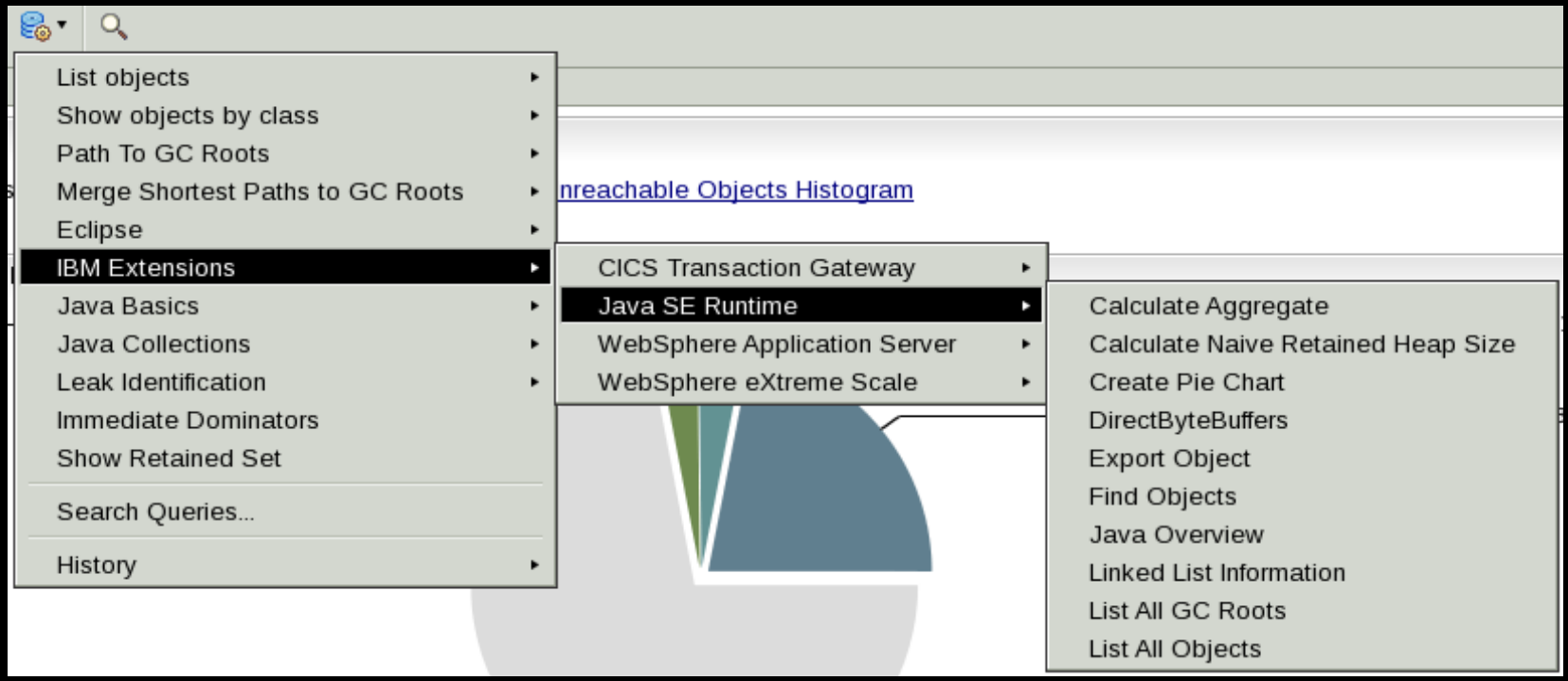
- Bottom right: Classloader shown by name/application

<Regex>	<Numeric>	<Numeric>	<Numeric>	<Numeric>
com.ibm.oti.vm.BootstrapClassLoader @ 0x10064630	113,738	28,094,588	116,902,923	53.97%
com.ibm.ws.webcontainer	754	50,327	66,483,783	30.69%
com.ibm.ws.runtime	27,149	1,330,047	14,570,111	6.73%
com.ibm.ws.bootstrap.ExtClassLoader @ 0x105354e0	1,149	67,279	5,786,356	2.67%
org.eclipse.core.launcher.Main$StartupClassLoader @ 0x1015ad00	9,068	423,455	4,323,645	2.00%
com.ibm.ws.jpa	281	16,599	2,959,959	1.37%
app:PlantsByWebSphere	1,250	51,503	2,114,039	0.98%
com.ibm.ws.wccmbase	885	64,535	1,205,535	0.56%
org.eclipse.emf.ecore	1,834	202,799	749,447	0.35%

Accessing the Queries

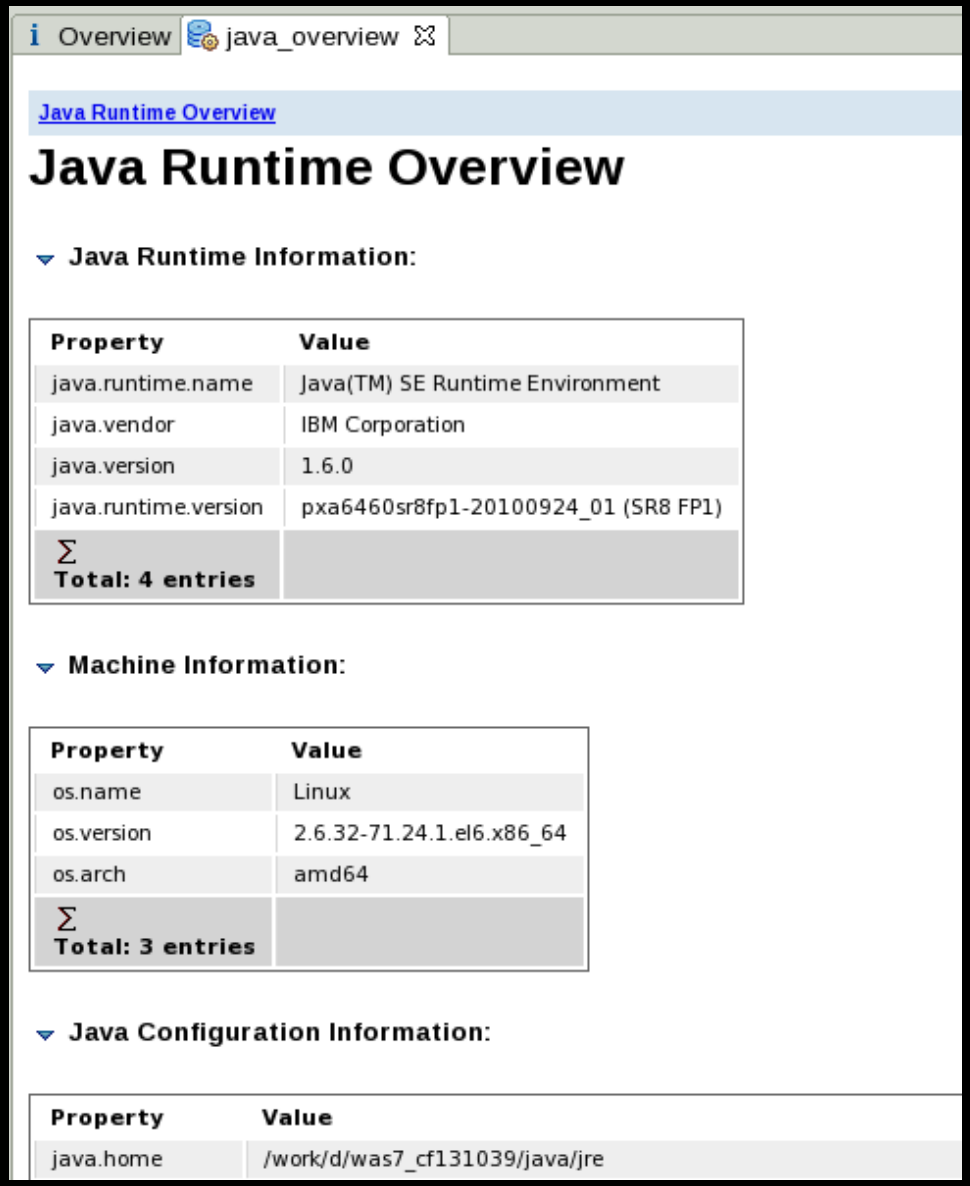


Java Extensions



- Highlights
  - DirectByteBuffers: Shows native memory held by DBBs and by whom
  - Java Overview: Shows things like the command line arguments
  - Export Object: Export a subset of the object graph as text to a file
  - List All GC Roots and List All Objects: Mimics HeapAnalyzer functionality

Java Overview



DirectByteBuffers

directbytebuffers | directbytebuffers

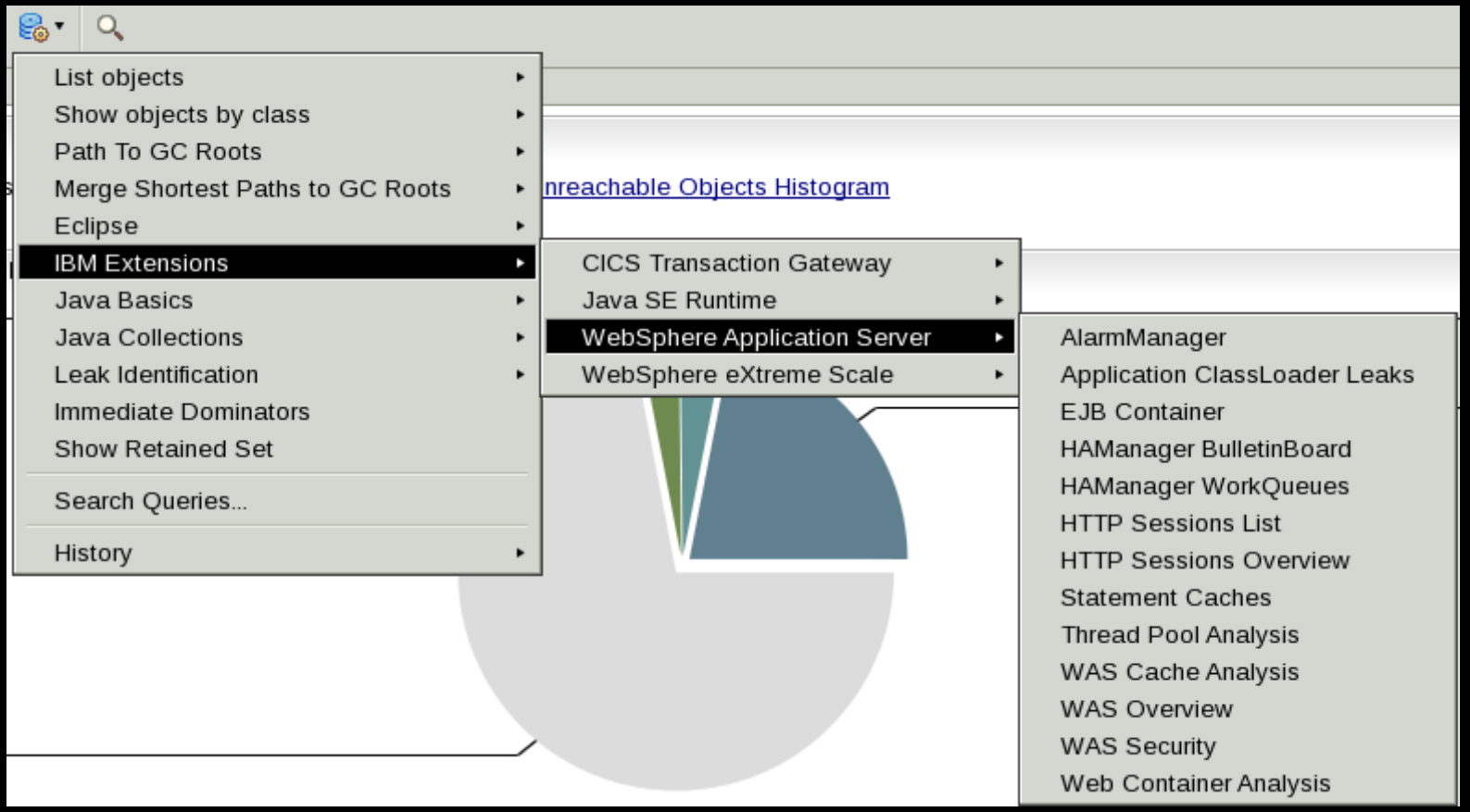
Class Name	Shallow Heap	Retained Heap	Capacity	IsViewed
<Regex>	<Numeric>	<Numeric>	<Numeric>	<Regex>
java.nio.DirectByteBuffer @ 0x3873ef8	72	72	1,052,672	false
java.nio.DirectByteBuffer @ 0x3873a40	72	72	1,052,672	false
java.nio.DirectByteBuffer @ 0x2c4b0a8	72	72	1,052,672	false
java.nio.DirectByteBuffer @ 0x2c4abd8	72	72	1,052,672	false
java.nio.DirectByteBuffer @ 0x24150f0	72	72	70,656	false
java.nio.DirectByteBuffer @ 0x2402bb8	72	72	70,656	false
java.nio.DirectByteBuffer @ 0x4dd9ca0	72	72	12,288	false
java.nio.DirectByteBuffer @ 0x4dd9b28	72	72	12,288	false
java.nio.DirectByteBuffer @ 0x4dd99b0	72	72	12,288	false
java.nio.DirectByteBuffer @ 0x4dd9838	72	72	12,288	false
java.nio.DirectByteBuffer @ 0x4dd96c0	72	72	12,288	false
java.nio.DirectByteBuffer @ 0x4dd9548	72	72	12,288	false
java.nio.DirectByteBuffer @ 0x4dd93d0	72	72	12,288	false
java.nio.DirectByteBuffer @ 0x4dd9258	72			
java.nio.DirectByteBuffer @ 0x4dd2458	72			
java.nio.DirectByteBuffer @ 0x4dd22e0	72			
java.nio.DirectByteBuffer @ 0x4dd2168	72			
java.nio.DirectByteBuffer @ 0x4dd1ff0	72			
java.nio.DirectByteBuffer @ 0x4dd1e78	72			
java.nio.DirectByteBuffer @ 0x4dd1d00	72			
java.nio.DirectByteBuffer @ 0x4dd1b88	72			
java.nio.DirectByteBuffer @ 0x4dd1a10	72			
java.nio.DirectByteBuffer @ 0x4dd1898	72			
java.nio.DirectByteBuffer @ 0x4dd1720	72			
Total: 24 of 631 entries	**45,432**		**9,385,696**	

directbytebuffers | directbytebuffers

```
Alignment size of 4096 bytes, and word size of 8 bytes.
637 total instances of java.nio.DirectByteBuffer
635 total non-viewed* DirectByteBuffers. Sum capacity (with overhead)=9403040 (8.96
631 total non-viewed*, non-phantomed** DirectByteBuffers. Sum capacity (with overhea
Maximum non-viewed*, non-phantomed** DirectByteBuffer = 1052672 (1.00 MB)
=> Sum DirectByteBuffer capacity available for GC: 17344 (16.93 KB)
=> Sum DirectByteBuffer capacity not available for GC: 9385696 (8.95 MB)

---------
Histogram of Incoming References (*, **)
4 instances incoming from com.ibm.ws.recoverylog.spi.LogFileHandle=4210688 (4.01 MB)
157 instances incoming from com.ibm.ws.util.ThreadPool$Worker=1929216 (1.83 MB)
```

WAS Extensions



- Highlights
  - WAS Overview: Shows WAS version, uptime, and experimental pie chart of what components (e.g. applications, sessions, etc.) are taking up the heap
  - Application ClassLoader Leaks: Find potential classloader leaks
  - HTTP Sessions List: Show all HTTP sessions, size, timeout, attributes, etc.

WAS Overview

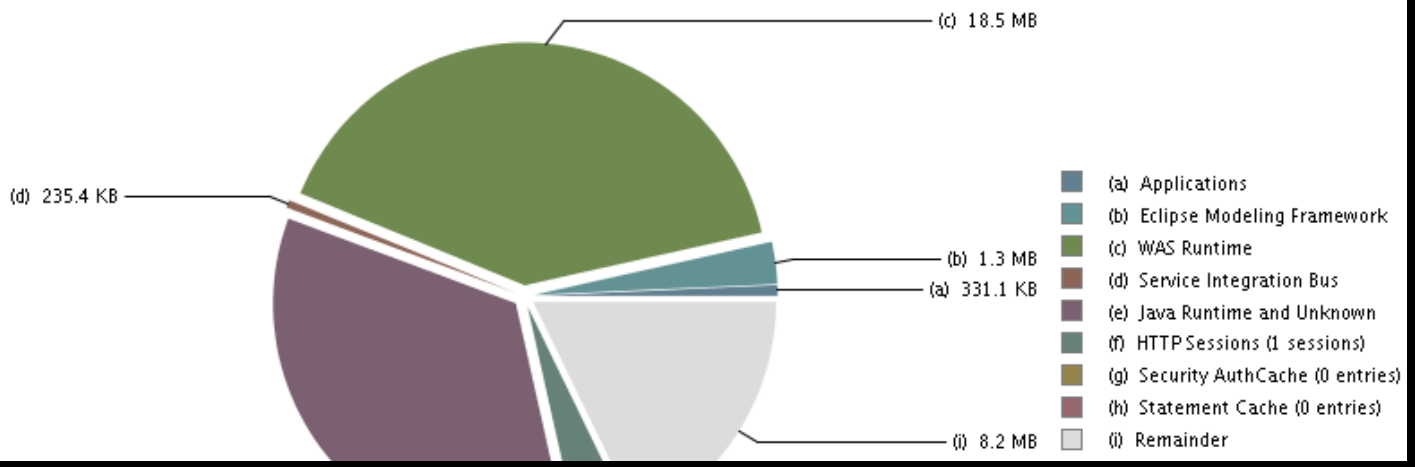
was_overview

Object	Name	Version	Build Date	Build Level	Install Directory
0x00009d25	IBM WebSphere Application Server - ND	7.0.0.13	10/2/10	cf131039.07	/work/d/was7_cf131039

Server Information

Description	Value
Server Name	server1
Node	oc8110753153Node01
Cell	oc8110753153Cell01
Full Server Name	oc8110753153Cell01\oc8110753153Node01\server1
Time Zone	America/New_York
Bit Mode	64 bit
Inferred Startup Time	Sun Apr 10 13:24:44.958 EDT 2011
Σ Total: 7 entries	

What is consuming the Java heap? [Experimental]



HTTP Session Analysis

http_sessions_list

Class Name	Shallow	Retained	AppName	SessionID	User Name	Timeout	IsOverflow	IsValid	Created
<Regex>	lumeric>	<Numeric>	<Regex>	<Regex>	<Regex>	Numeric>	<Regex>	<Regex	<Regex>
com.ibm.ws.session.store.memory.MemorySession @ 0x31	112	1,776,496	default_hostswat	-U4y-y-o1BLKhNDl4vN8zuK	anonymous	1,800	false	true	Sun Apr 10 10:29:00 PDT 2011
Key=sess2600,Value=com.ibm.Sessions$SimpleHashtal	32	216							
com.ibm.Sessions$SimpleHashtable @ 0x4125510	24	184							
Key=sess2601,Value=com.ibm.Sessions$SimpleHashtal	32	216							
Key=sess2602,Value=com.ibm.Sessions$SimpleHashtal	32	216							
Key=sess2603,Value=com.ibm.Sessions$SimpleHashtal	32	216							

http_sessions_overview

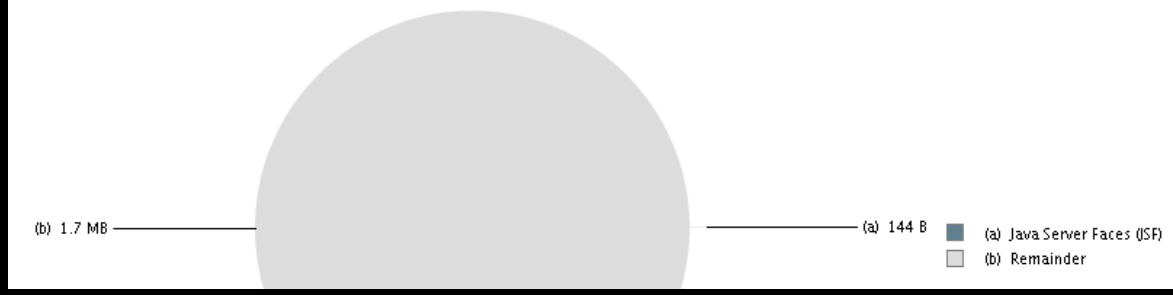
WebSphere Application Server HTTP Sessions Overview

WebSphere Application Server HTTP Sessions Overview

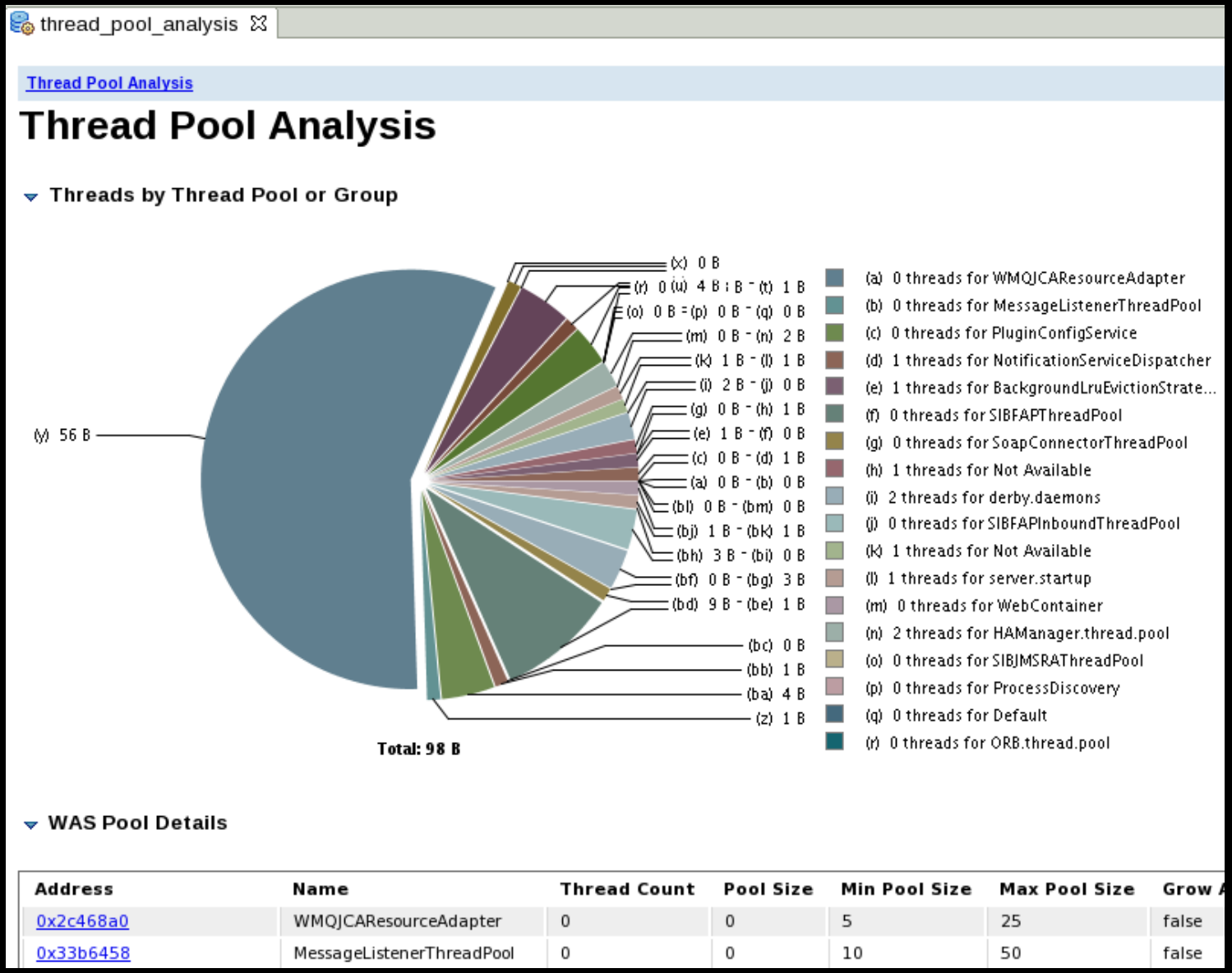
HTTP Sessions Information

Description	Value
Total Retained Heap of HTTP Sessions	1.69 MB
Number of HTTP sessions	1
Number of Unique Users	1
Number of Unique Applications	1
Oldest Session Created On	Sun Apr 10 10:29:00 PDT 2011
Newest Session Created On	Sun Apr 10 10:29:00 PDT 2011
Minimum Session Timeout	30 minutes
Maximum Session Timeout	30 minutes
Approximate Java Server Faces (JSF) usage	144.0 B
∑ Total: 9 entries	

Retained Heap of HTTP Sessions



Thread Pool Analysis



Address	Name	Thread Count	Pool Size	Min Pool Size	Max Pool Size	Grow A
0x2c468a0	WMQJCAResourceAdapter	0	0	5	25	false
0x33b6458	MessageListenerThreadPool	0	0	10	50	false

Web Application Analysis

Address	Virtual Host Name	Web Group Name	Web App Name	Loader	Destroyed	Current Sessions
0x31326a8	default_host	/swat/*	WasSwat	war:WasSwat/WasSwatWeb.war	false	1
0x4ce7f58	default_host	/PlantsByWebSphere/docs/*	PlantsByWebSphere	war:PlantsByWebSphere/PlantsGallery.war	false	0
0x387edd8	default_host	/SamplesGallery/*	SamplesGallery	war:SamplesGallery/GalleryMenu.war	false	0
0x34d7530	default_host	/IBM_WS_SYS_RESPONSESERVLET/*	ibmasyncrsp	war:ibmasyncrsp/ibmasyncrsp.war	false	0
0x3473550	default_host	/WSsamples/*	SamplesGallery	war:SamplesGallery/Gallery.war	false	0
0x393b678	default_host	/*	DefaultApplication	war:DefaultApplication/DefaultWebApplication.war	false	0
0x3f07608	default_host	/PlantsByWebSphere/*	PlantsByWebSphere	war:PlantsByWebSphere/PlantsByWebSphere.war	false	0
Σ Total: 7 entries						

Interactive Diagnostic Data Explorer

IDDE

- IDDE is different from MAT in that it is not designed for heap graph analysis.
- This means that the IDDE load time, particularly in recent versions of IBM Java with direct dump reading, are very fast.
- However, you do not get retained set analysis, etc.
- Also, some objects seen by IDDE may be garbage (MAT does a garbage collection on start).
- IDDE only supports IBM dumps (not HPROF).
- Also support extensions, although different API than MAT.
-

Extending Memory Analyzer

Extension Points

- See http://wiki.eclipse.org/MemoryAnalyzer/Extending_Memory_Analyzer
- org.eclipse.mat.report.query
  - This adds a menu item that executes your code and creates new tab(s) of output in any form you want: Tree, Text, HTML, Pie Charts, etc.
- org.eclipse.mat.api.nameResolver
  - Provide readable description of an object in some of MAT's view (like toString())

API

- ISnapshot – Represents one dump
  - Each object and class has a unique Integer ID. Most methods will return an int or array of ints. Then you can call snapshot.getObject with the int to get an IObject representing the item
  - getGCRoots – List of all GC roots
  - getClasses – List of all classes (or search by name)
  - getInboundRefererIds – List of incoming references
  - getOutboundReferentIds – List of outgoing references
  - getHeapSize – Shallow heap size of the object
  - getRetainedHeapSize – Retained heap size of the object

API

- IObject – represents an item in the heap
  - getObjectAddress – This is the address of the object in the Java heap
  - getClazz – Get the class of an object
  - getUsedHeapSize/getRetainedHeapSize – Same as ISnapshot
  - getDisplayName – The class, address, and name resolver
  - resolveValue – For an IBM system dump or HPROF dump, given a name of a field, find the object representing that field. This identifier can have periods which separate going down that tree of items.

API

- IResult – What a query returns
  - TextResult – Plain text or HTML content
  - ObjectListResult – Grid of results with in/outbound refs
  - SectionSpec – Separate results into sections
    - Add QuerySpecs
  - PieFactory...build() - Generate a pie chart
  - ListResult – Table of items
  - CompositeResult – Display results in separate tabs

Query Extension

```
@Name("My Query") // This will be the menu item
@Category("IBM Extensions/WebSphere Application Server") // Subfolders
@Help("Short description\n\n")
public class MyQuery implements org.eclipse.mat.query.IQuery {
    @Argument
    public ISnapshot snapshot;

    public IResult execute(IProgressListener listener) throws Exception {
        String someResult = "# roots=" + snapshot.getGCRoots().length;
        return new TextResult(someResult, true);
    }
}
```

```
<plugin>
    <extension point="org.eclipse.mat.report.query">
        <query impl="MyQuery" />
    </extension>
</plugin>
```

Name Resolver Extension

```
@Subject("com.example.MyClass") // Describes which application class
public class MyClassNameResolver { // Extension class for MAT
    public String resolve(IObject object) { // IObject represents object in dump
        IObject name = (IObject) object.resolveValue("nameField"); // Read field
        return name == null ? null : name.getClassSpecificName(); // printable
    }
}
```

```
<plugin>
    <extension point="org.eclipse.mat.api.nameResolver">
        <resolver impl="MyClassNameResolver" />
    </extension>
</plugin>
```

Extending with Reports

```
<extension point="org.eclipse.mat.report.report">
    <report id="wasanalysis"
        name="My Report" description="Description" file="META-INF/reports/my
report.xml" />
</extension>
```

myreport.xml:

```
<section name="My Report" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.eclipse.org/mat/report.xsd"
xsi:schemaLocation="http://www.eclipse.org/mat/report.xsd
../../../org.eclipse.mat.report/schema/report.xsd">

<param key="html.collapsed" value="false" />
<param key="filename_suffix" value="MyReport" />
<query name="My Report">
<command>my_query_name</command>
</query>

</section>
```

Conclusion

- The Memory Analyzer Tool (MAT) is quite advanced, but attempts at automating or simplifying heapdump analysis have generally failed. MAT balances the inherent complexity of the Java object graph with a good UI, a powerful extensibility model, precise calculations, a useful query language, and loads of other features.
- Core dump based debugging is an important future direction that middleware problem determination is moving towards.

Other Heapdump Tools

- HeapAnalyzer (HA) [ISA]
  - http://www.alphaworks.ibm.com/tech/heapanalyzer
  - User-friendly but inaccurate total size calculation and does not read system dumps
- Memory Dump Diagnostic for Java (MDD4J) [ISA]
  - Deprecated
- Heap Analysis Tool (HAT)
  - https://hat.dev.java.net/
- Other: HeapRoots, svcdump.jar, some profiler tools can analyze heapdumps

Other Links

- Be careful when diagnosing Java memory leaks:
  - https://www.ibm.com/developerworks/mydeveloperworks/blogs/kevgrig/entry/be_careful_when_diagnosing_java_memory_leaks17?lang=en
- How to use MAT to compare dumps:
  - https://www.ibm.com/developerworks/mydeveloperworks/blogs/kevgrig/entry/how_to_use_the_memory_analyzer_tool_mat_to_compare_heapdumps_and_system_dumps20?lang=en
- IEMA in ISA:
  - https://www.ibm.com/developerworks/mydeveloperworks/blogs/kevgrig/entry/the_ibm_extensions_for_memory_analyzer_are_now_available_through_the_ibm_support_assistant24?lang=en