Improving Application Performance With Monitoring and Profiling Tools

Jaroslav Bachorík, Sun Microsystems Inc.
Gregg Sporar, Sun Microsystems Inc.

ID#TS-6000

Describe, demonstrate, and discuss some of the tools available for monitoring and profiling Java™ platform applications. Answer the question: “Which tools do I use for which situations?”

GOAL

Agenda

- What Is the Problem?
- Disclaimers...
- CPU Tools
- Memory Tools
- Multipurpose Tools
- Summary
- Resources

Agenda

- What Is the Problem?
- Disclaimers...
- CPU Tools
- Memory Tools
- Multipurpose Tools
- Summary
- Resources

What Is The Problem?

- So Many Tools From Which To Choose...
  - Operating System Tools
  - JDK™ Tools
  - Third Party Tools

An Embarrassment of Riches....

Agenda

- What Is the Problem?
- Disclaimers
- CPU Tools
- Memory Tools
- Multipurpose Tools
- Summary
- Resources

Disclaimers

- Sun's Java Virtual Machine (JVM™)
- Mostly JDK version 6, with some mention of JDK version 1.4.2 and 5
- We Are Not The Experts (well, maybe on some of the tools we are :-))

Agenda

- What Is the Problem?
- Disclaimers...
- CPU Tools
- Memory Tools
- Multipurpose Tools
- Summary
- Resources

DTrace

An operating system tool for dynamic tracing

```
# ./jagg.d
Tracing... Hit Ctrl-C to end.
^C
                                        Greeting.greet                        3
                          java/io/BufferedWriter.newLine                      3
                             java/io/PrintStream.newLine                      3
                             java/io/PrintStream.print                        3
                             java/io/PrintStream.println                      3
                                java/lang/Thread.currentThread                3
                                java/lang/Thread.sleep                        3
                     java/io/BufferedOutputStream.write                       6
                          java/io/BufferedWriter.flushBuffer                  6
                          java/io/BufferedWriter.min                          6
                          java/io/BufferedWriter.write                        6
                        java/io/FileOutputStream.write                        6
                        java/io/FileOutputStream.writeBytes                   6
                      java/io/OutputStreamWriter.flushBuffer                  6
                      java/io/OutputStreamWriter.write                        6
                                  java/io/Writer.write                        6
                                java/lang/Object.<init>                       6
                                java/lang/String.getChars                     6
                                java/lang/String.indexOf                      6
[...output truncated...]
                     java/io/BufferedOutputStream.flush                       9
                     java/io/BufferedOutputStream.flushBuffer                 9
                             java/io/PrintStream.write                        9
                          java/io/BufferedWriter.ensureOpen                  12
                             java/io/PrintStream.ensureOpen                  12
                                java/lang/System.arraycopy                   12
                             java/nio/ByteBuffer.array                       12
                     java/nio/charset/CoderResult.isUnderflow                12
                                  java/nio/Buffer.position                   18
                             java/nio/CharBuffer.arrayOffset                 18
                             java/nio/ByteBuffer.arrayOffset                 24
```

DTrace (cont'd)

An operating system tool for dynamic tracing

- Strengths
  - *D*ynamic
    - Turn it on when you need it
    - Turn it off when you don't
  - Shows everything
  - Accuracy via instrumentation
  - Not just Java platform...
- Limitations
  - Latest Solaris™ Operating System and OS X only
  - Instrumentation overhead
  - Works best with JDK version 6 and up
  - Steep learning curve
    - Chime
    - D-Light

Sun Studio Collector/Analyzer

A third party tool for CPU sampling

Performance Analyzer [test.1.er]

File View Timeline Help

Find Text:

Functions | Callers-Callees | Source | Lines | Disassembly | PCs | Timeline | LeakList | Statistics | Experiments

Summary | Event | Legend | Leak

Data for Current Timeline Selection

Experiment Name:	test.1.er
Event Type:	Clock Profiling Data
Leaf Function:	__lwp_cond_wait
Timestamp (sec.):	15.205705
LWP:	7
Thread:	7
CPU:	(unknown)
Duration (msec.):	14400.073
Micro State:	User Lock

Call Stack for Selected Event

- __lwp_cond_wait + 0x00000008
- _lwp_cond_wait + 0x0000000C
- Monitor::wait(int,long) + 0x000003B8
- CompileBroker::compiler_thread_loop() + 0x00000214
- JavaThread::run() + 0x00000288
- _start + 0x00000134
- _lwp_start + 0x00000000

Sun Studio Collector/Analyzer (cont'd)

A third party tool for CPU sampling

- Strengths
  - Low overhead - uses sampling
  - Source line level of detail
  - Powerful visualization tools
  - Detailed CPU data
  - Not just Java platform...
- Limitations
  - Solaris OS and Linux Only
  - Uses sampling
  - Requires start/stop of the application
  - Postmortem only
  - Analysis must be done on a system with same byte order

jps, jinfo, and jstack

JDK tools for basic JVM software information

```
"Finalizer" daemon prio=8 tid=0x28988400 nid=0x1318 in Object.wait() [0x28f5f000
..0x28f5fb00]
   java.lang.Thread.State: WAITING (on object monitor)
        at java.lang.Object.wait(Native Method)
        - waiting on <0x057fa1b0> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:116)
        - locked <0x057fa1b0> (a java.lang.ref.ReferenceQueue$Lock)
        at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:132)
        at java.lang.ref.Finalizer$FinalizerThread.run(Finalizer.java:159)

   Locked ownable synchronizers:
        - None

"Reference Handler" daemon prio=10 tid=0x28984800 nid=0x119c in Object.wait() [0
x28d5f000..0x28d5fb80]
   java.lang.Thread.State: WAITING (on object monitor)
        at java.lang.Object.wait(Native Method)
        - waiting on <0x0549ac20> (a java.lang.ref.Reference$Lock)
        at java.lang.Object.wait(Object.java:485)
        at java.lang.ref.Reference$ReferenceHandler.run(Reference.java:116)
        - locked <0x0549ac20> (a java.lang.ref.Reference$Lock)

   Locked ownable synchronizers:
        - None

"VM Thread" prio=10 tid=0x28977800 nid=0xaf0 runnable

"Gang worker#0 (Parallel GC Threads)" prio=10 tid=0x0013cc00 nid=0xcd0 runnable


"Gang worker#1 (Parallel GC Threads)" prio=10 tid=0x0013e000 nid=0xac0 runnable


"Concurrent Mark-Sweep GC Thread" prio=10 tid=0x0135a000 nid=0x1334 runnable
"VM Periodic Task Thread" prio=10 tid=0x289b1000 nid=0x1390 waiting on condition


JNI global references: 1783
```

jps, jinfo, and jstack (cont'd)

JDK tools for basic JVM software information

- Strengths
  - Solaris OS, Linux, Windows
  - Dynamic
  - Remote access
  - JDK version 1.4.2 and up
  - Very low overhead
  - Script-able command line user interface
  - Helps identify deadlocks (jstack)
- Limitations
  - Not all jinfo and jstack features available on Windows
  - Rudimentary command line user interface

BTrace

A third party tool for dynamic tracing of Java technology

```
@BTrace public class WebServiceTracker {
   // store webservice entry time in this thread local
   @TLS private static long startTime;

   @OnMethod(
     clazz="@javax.jws.WebService",
     method="@javax.jws.WebMethod"
   )
   public static void onWebserviceEntry() {
       print("entering webservice ");
       println(strcat(strcat(name(probeClass()), "."), probeMethod()));
       startTime = timeMillis();
   }

   @OnMethod(
     clazz="@javax.jws.WebService",
     method="@javax.jws.WebMethod",
     location=@Location(Kind.RETURN)
   )
   public static void onWebserviceReturn() {
       println(strcat("Time taken (msec) ", str(timeMillis() - startTime)));
       println("==========================");
   }
```

BTrace (cont'd)

A third party tool for dynamic tracing of Java technology

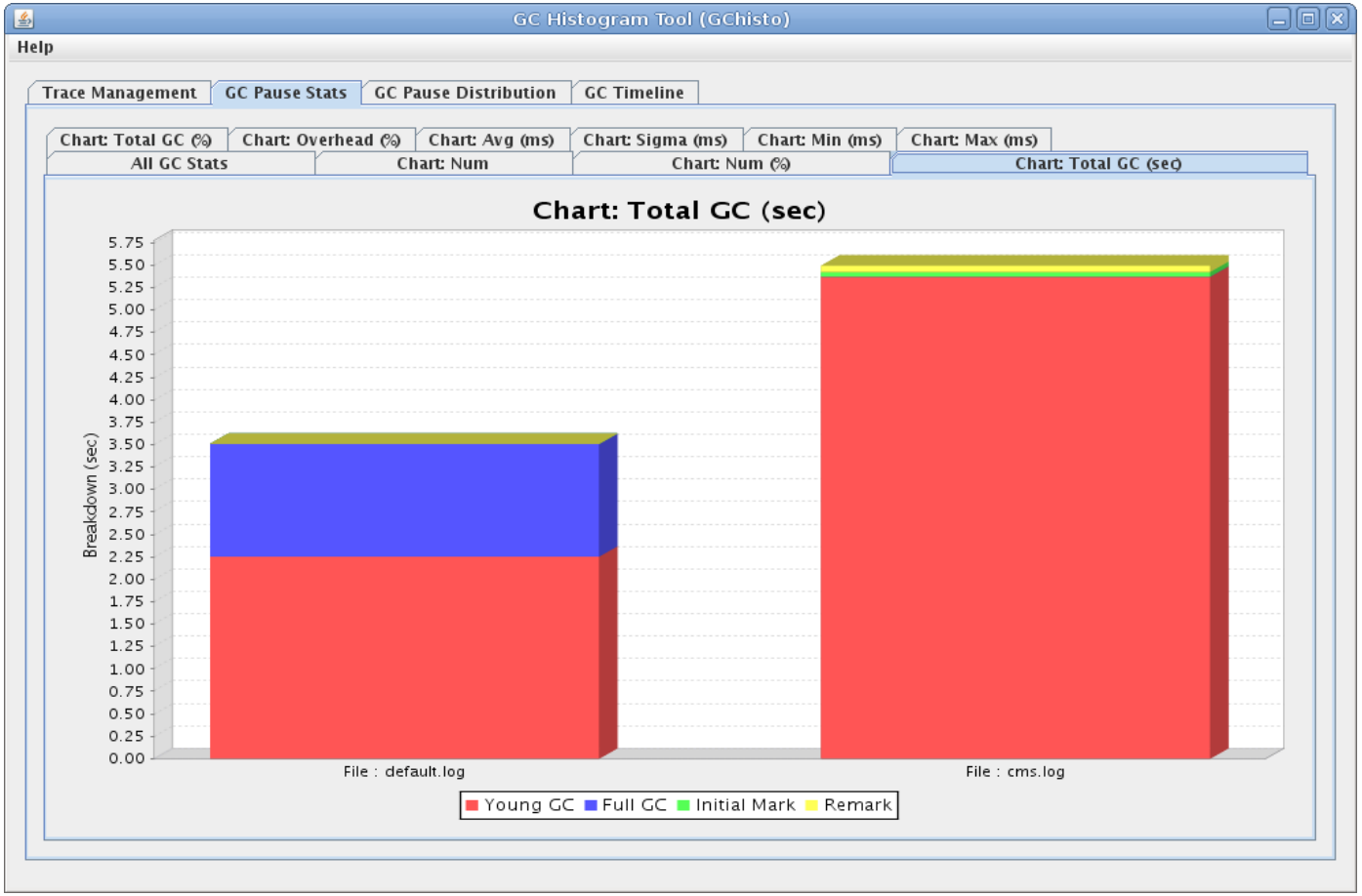
- Strengths
  - Solaris OS, Linux, Windows, OS X
  - Accuracy via instrumentation
  - Dynamic
  - Integration with DTrace where available
  - Scripts written in Java language
- Limitations
  - Requires JDK version 6
  - Instrumentation overhead
  - Java code *only*
  - Bleeding edge technology – not mature

Agenda

- What Is the Problem?
- Disclaimers...
- CPU Tools
- Memory Tools
- Multipurpose Tools
- Summary
- Resources

GChisto

A third party tool for analyzing garbage collector log files



GChisto (cont'd)

A third party tool for analyzing garbage collector log files

- Strengths
  - Solaris OS, Linux, Windows
  - Comparison views
    - which garbage collector to use
    - how to tune it
  - Plug-able log readers
  - Live or postmortem analysis
- Limitations
  - Pause times only
  - Relatively new – some important features still TBD
  - Limited testing on older JVM software logs
  - Don't forget:
    - -XX:+PrintGCTimeStamps
    - -XX:+PrintGCDetails

jmap

A JDK tool for examining memory usage

```
 num     #instances         #bytes  class name
----------------------------------------------
   1:        175993       33166432  [C
   2:        103139       11511640  <constMethodKlass>
   3:         15924       10867624  [I
   4:         12055        9746696  [B
   5:        103139        8259760  <methodKlass>
   6:        152147        7052352  <symbolKlass>
   7:          9290        5686024  <constantPoolKlass>
   8:        196753        4722072  java.lang.String
   9:          9290        4417984  <instanceKlassKlass>
  10:        171578        4117872  java.util.HashMap$Entry
  11:          8660        3801328  <constantPoolCacheKlass>
  12:         32519        3507520  [Ljava.util.HashMap$Entry;
  13:         34700        2367880  [Ljava.lang.Object;
  14:         32181        1287240  java.util.HashMap
  15:         15363         989656  [S
  16:          9938         954048  java.lang.Class
  17:         27309         873888  com.sun.org.apache.xerces.internal.dom.Defer
redAttrImpl
  18:         30963         743112  java.util.Hashtable$Entry
  19:         15266         702448  [[I
  20:           496         536904  [Ljava.util.concurrent.ConcurrentHashMap$Has
hEntry;
  21:         21957         526968  java.lang.ref.WeakReference
  22:         11803         472120  java.util.WeakHashMap$Entry
  23:         18254         438096  java.util.ArrayList
  24:         12755         408160  org.openide.util.WeakListenerImpl$ListenerRe
ference
  25:         15101         362424  org.netbeans.spi.editor.highlighting.support
.OffsetsBag$Mark
  26:          6382         357392  org.openide.filesystems.MultiFileObject
  27:          2607         334416  [Ljava.util.Hashtable$Entry;
  28:          4131         330480  java.lang.reflect.Method
  29:         16720         267520  java.util.jar.Attributes$Name
  30:          9736         233664  org.openide.util.WeakListenerImpl$ProxyListe
-- More  --
```

jmap (cont'd)

A JDK tool for examining memory usage

- Strengths
  - Solaris OS, Linux, Windows
  - Dynamic
  - Remote access
  - No overhead
  - Fast
  - Easy to use
  - Script-able command line user interface
  - Can be used with jhat
- Limitations
  - JDK version 1.4.2 support: Solaris OS only
  - Some features not available on Windows
  - Rudimentary command line user interface

jhat

A JDK tool for doing heap inspection

http://localhost:7000/

All Classes (excluding platform)

Package <Arrays>

class [Lcom.sun.source.tree.ExpressionTree; [0x272b8af0]
class [Lcom.sun.source.tree.Tree; [0x272b8718]
class [Lcom.sun.tools.javac.code.BoundKind; [0x272d89a0]
class [Lcom.sun.tools.javac.code.Lint$LintCategory; [0x27117898]
class [Lcom.sun.tools.javac.code.Scope$Entry; [0x271730c8]
class [Lcom.sun.tools.javac.code.Source; [0x270fb990]
class [Lcom.sun.tools.javac.code.Symbol$VarSymbol; [0x272e2708]
class [Lcom.sun.tools.javac.code.Symbol; [0x272e0d68]
class [Lcom.sun.tools.javac.code.Type; [0x27170270]
class [Lcom.sun.tools.javac.jvm.Code$LocalVar; [0x272edfe8]
class [Lcom.sun.tools.javac.jvm.Code$StackMapFormat; [0x27260b40]
class [Lcom.sun.tools.javac.jvm.Items$Item; [0x272f36f0]

jhat (cont'd)

A JDK tool for doing heap inspection

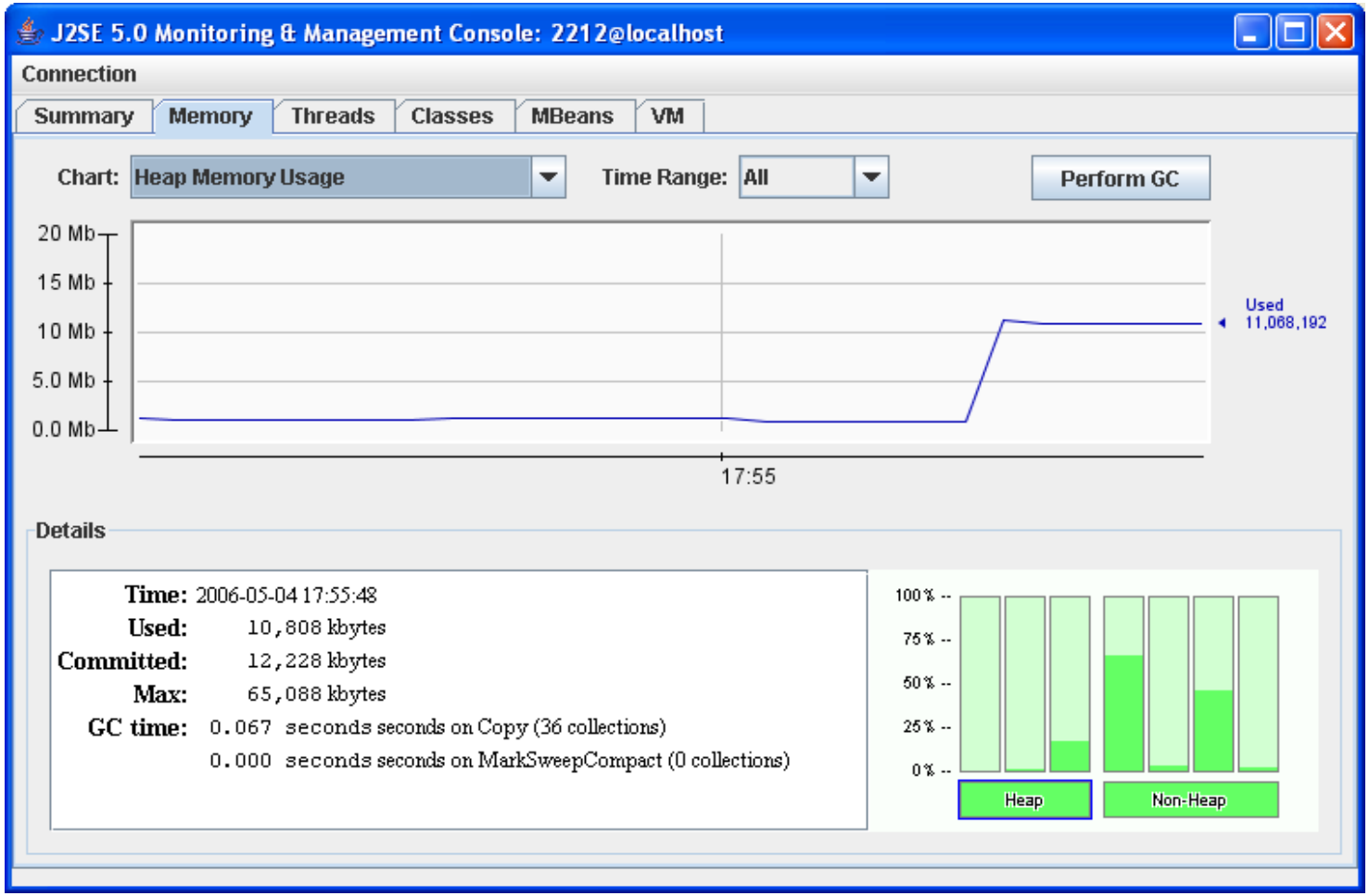
- Strengths
  - Solaris OS, Linux, Windows
  - Shows *everything*
    - *Only* tool for solving PermGen problems
  - **O**bject **Q**uery **L**anguage
  - JDK version 6 can read binary heap dumps created by older JVM versions
  - Can be used with jmap
- Limitations
  - Postmortem only
  - Rudimentary user interface
  - OQL learning curve
  - Too much information...

Agenda

- What Is the Problem?
- Disclaimers...
- CPU Tools
- Memory Tools
- Multipurpose Tools
- Summary
- Resources

jconsole

A JDK tool for monitoring Java technology applications



jconsole (cont'd)

A JDK tool for monitoring Java technology applications

- Strengths
  - Solaris OS, Linux, Windows
  - JDK version 1.4.2 and up
  - Dynamic
  - Remote access
  - Support for JMX MBeans
  - Easy to use
- Limitations
  - Mostly high level information
  - Command line flag required for JDK version 1.4.2 and JDK version 5.0

NetBeans IDE Profiling Tools

A third party tool for dynamic bytecode instrumentation

NetBeans IDE 6.0

File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help

<default config>

Projects | Files | Profiler

Controls

Status

Type: CPU
Configuration: Analyze Performance
Status: Running

Profiling Results

Take Snapshot | Live Results

Reset Collected Results

Saved Snapshots

AnagramGame

Open | Delete | Save As... | Load...

View

VM Telemetry | Threads

Basic Telemetry

Instrumented: 15 681 Methods
Filter: Profile all classes
Threads: 8
Total Memory: 5 177 344 B
Used Memory: 2 813 784 B
Time Spent in GC: 0,0%

CPU: 06:28:26 odp. *

View: Methods

Call Tree - Method	Time [%]	Time	Invocations
All threads		1047 ms (100%)	1
main		1006 ms (100%)	1
com.toy.anagrams.ui.Anagrams.**main** (String[])		1006 ms (100%)	1
com.toy.anagrams.ui.Anagrams.**<init>** ()		963 ms (95,7%)	1
java.awt.Window.**setVisible** (boolean)		43.1 ms (4,3%)	1
Self time		0.040 ms (0%)	1
AWT-EventQueue-0		41.4 ms (100%)	1
com.toy.anagrams.ui.Anagrams$3.**actionPerformed** (java.awt.event.ActionEve		37.4 ms (90,4%)	8
com.toy.anagrams.ui.Anagrams.**access$200** (com.toy.anagrams.ui.Anagran		37.4 ms (90,3%)	8
com.toy.anagrams.ui.Anagrams.**nextTrialActionPerformed** (java.awt.		37.4 ms (90,2%)	8
javax.swing.text.JTextComponent.**setText** (String)		33.6 ms (81,3%)	16
javax.swing.JComponent.**requestFocusInWindow** ()		2.92 ms (7,1%)	8
javax.swing.JLabel.**setText** (String)		0.595 ms (1,4%)	8
Self time		0.066 ms (0,2%)	8
javax.swing.JRootPane.**setDefaultButton** (javax.swing.JButton)		0.026 ms (0,1%)	8

Hot Spots - Method	Self time [%]	Self time	Invocations
javax.swing.text.AbstractDocument.**insertUpdate** (javax.swing.text.AbstractDocument...		0.076 ms (0,2%)	8
java.awt.Component.**requestFocusHelper** (boolean, boolean, sun.awt.CausedFocusEv...		0.075 ms (0,2%)	11
java.awt.Rectangle.**<init>** (int, int, int, int)		0.074 ms (0,2%)	122
sun.font.FontDesignMetrics.**getLatinCharWidth** (char)		0.073 ms (0,2%)	237
java.util.WeakHashMap.**get** (Object)		0.073 ms (0,2%)	31
javax.swing.RepaintManager.**extendDirtyRegion** (java.awt.Component, int, int, int, int)		0.073 ms (0,2%)	66
com.toy.anagrams.ui.Anagrams.**nextTrialActionPerformed** (java.awt.event.ActionEv...		0.066 ms (0,2%)	8
javax.swing.text.AbstractDocument.**getLength** ()		0.066 ms (0,2%)	94
java.util.IdentityHashMap.**get** (Object)		0.065 ms (0,2%)	66
javax.swing.text.PlainView.**lineToRect** (java.awt.Shape, int)		0.065 ms (0,2%)	32
java.awt.Component.**isVisible** ()		0.064 ms (0,2%)	237
javax.swing.text.Utilities.**isComposedTextElement** (javax.swing.text.Document, int)		0.063 ms (0,2%)	8
javax.swing.RepaintManager.**currentManager** (java.awt.Component)		0.061 ms (0,1%)	76
javax.swing.text.SegmentCache.**getSegment** ()		0.060 ms (0,1%)	56
java.awt.Frame.**getExtendedState** ()		0.060 ms (0,1%)	10

[Method Name Filter]

Call Tree | Hot Spots | Combined | Info

Output

anagrams (profile) running...

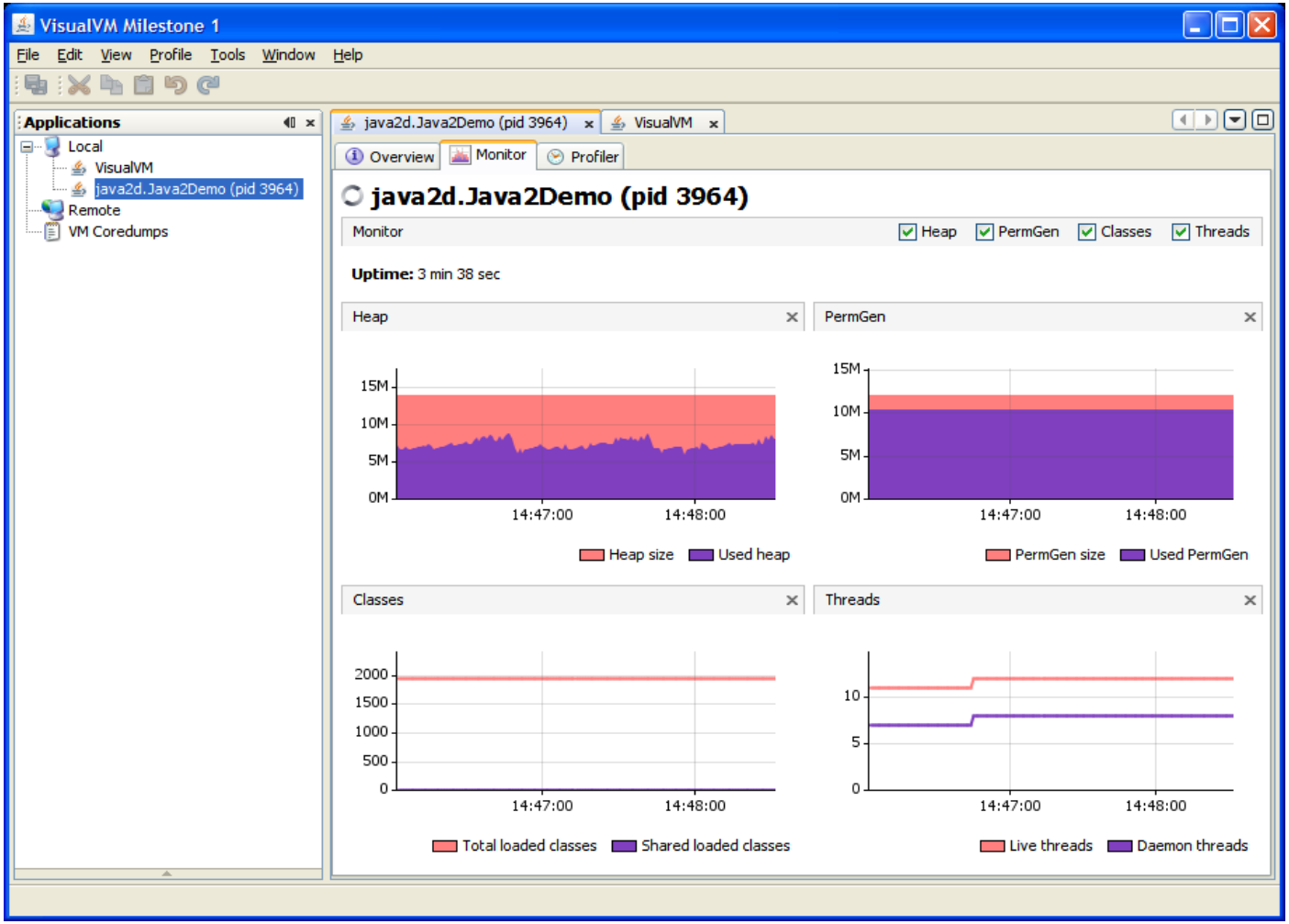
NetBeans IDE Profiling Tools (cont'd)

A third party tool for dynamic bytecode instrumentation

- Strengths
  - Solaris OS, Linux, Windows, OS X
  - Integrated :-)
  - Accuracy via instrumentation
  - Easy to use
    - Part of development work flow
    - One click
  - Feature-rich
    - Thread, CPU, and memory profiling
    - Root method selection
    - Source line level of detail
    - Support for load testing
    - Live and postmortem memory profiling
- Limitations
  - JDK version 5 and higher only
  - Instrumentation overhead
  - Not helpful for memory problems with PermGen
  - Requires NetBeans IDE

Java VisualVM

A JDK tool for monitoring and profiling Java technology applications



Java VisualVM (cont'd)

A JDK tool for monitoring and profiling Java technology applications

- Strengths
  - Solaris OS, Linux, Windows, Mac OS X
  - One tool that incorporates functionality from:
    - jps, jinfo, and jstack
    - jconsole
    - jstat
    - NetBeans IDE profiling tools
  - Easily extended
    - BTrace
    - GChisto
    - GlassFish
    - Terracotta
- Limitations
  - Very new – not mature yet
  - Limited features for JDK version 1.4.2 and JDK version 5.0
  - No option to see your source code

Agenda

- What Is the Problem?
- Disclaimers...
- CPU Tools
- Memory Tools
- Multipurpose Tools
- Summary
- Resources

Summary

- Lots of tools...
  - Strengths
  - Limitations
- Some consolidation occurring

Agenda

- What Is the Problem?
- Disclaimers...
- CPU Tools
- Memory Tools
- Multipurpose Tools
- Summary
- Resources

Resources

- JavaOne℠ 2008 Event Sessions
  - Tuesday:
    - TS-5716 - D-I-Y (Diagnose-It-Yourself): Adaptive Monitoring for Sun Java Real-Time System
    - TS-7392 - Maximizing Enterprise Java Performance on Multi-core Platforms
    - BOF-5552 - Java™ Platform Observability by Bytecode Instrumentation
  - Wednesday:
    - TS-6000 - Improving Application Performance with Monitoring and Profiling Tools
    - LAB-8430 - Isolating Performance Bottlenecks and Memory Leaks With the NetBeans™ Profiler
    - TS-7735 - New Approaches to Visualization and Management of Application Server Clusters
  - Thursday:
    - TS-6145 - Using DTrace with Java Technology-Based Applications: Bridging the Observability Gap
    - BOF-4994 - End-to-End Tracing of Ajax/Java Technology-Based Applications, Using Dynamic Tracing (DTrace)

Resources (cont'd)

- JavaOne 2008 Event Sessions
  - Thursday (cont'd):
    - TS-5419 - The Garbage-First Garbage Collector
    - TS-6434 - Java Platform Performance: Case Studies in Bottleneck Identification and Removal
    - LAB-1440 – Performance Troubleshooting
    - BOF-5223 - VisualVM: Integrated and Extensible Troubleshooting Tool for the Java Platform
  - Friday:
    - TS-5729 - Automated Heap Dump Analysis for Developers, Testers, and Support Employees
    - TS-6145 - Using DTrace with Java Technology-Based Applications: Bridging the Observability Gap
    - TS-6000 - Improving Application Performance with Monitoring and Profiling Tools

Resources (cont'd)

- Performance and Troubleshooting Guides
  - http://java.sun.com/performance/reference/whitepapers/6_performance.html
  - http://java.sun.com/javase/6/webnotes/trouble/TSG-VM/html/docinfo.html
- DTrace
  - http://www.sun.com/bigadmin/content/dtrace/
  - http://www.devx.com/Java/Article/33943/0/page/1
- Sun Studio Collector/Analyzer
  - http://developers.sun.com/solaris/articles/javapps.html
  - http://developers.sun.com/solaris/articles/dtrace.html
- BTrace
  - https://btrace.dev.java.net/
- GChisto
  - https://gchisto.dev.java.net/

Resources (cont'd)

- jmap/jhat
  - http://weblogs.java.net/blog/gsporar/archive/2007/04/tracking_down_m.html
  - http://weblogs.java.net/blog/gsporar/archive/2007/05/tracking_down_m_1.html
- NetBeans IDE Profiling Tools
  - http://www.netbeans.org/features/java/profiler.html
  - http://profiler.netbeans.org/
- Java VisualVM
  - https://visualvm.dev.java.net

THANK YOU

Jaroslav Bachorík, Sun Microsystems Inc.
Gregg Sporar, Sun Microsystems Inc.