Java is a trademark of Sun Microsystems, Inc.

JavaOne℠

Debugging Your Production JVM Machine

Ken Sipe
Perficient (PRFT)
kensipe@gmail.com

Abstract

> Learn the tools and techniques used to monitor, trace and debugging running Java applications.

Agenda

- Java Memory Management
- Memory Management Tools
  - Command-line Tools
  - VisualVM
- Btrace
- Summary
- Resources

Agenda

- Java Memory Management
- Memory Management Tools
  - Command-line Tools
  - VisualVM
- Btrace
- Summary
- Resources

Don't Worry…

- C (malloc / free)
- C++ (new / delete)
- Java (new / gc)
- Memory Allocation / Deallocation in Java is automatic

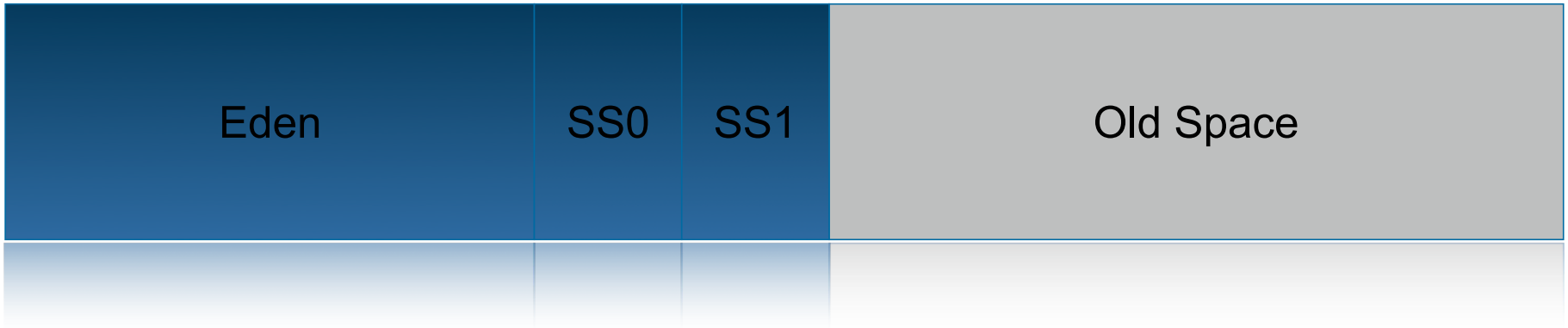
Object Lifetimes

- Most objects live very short lives
  - 80-98% of all newly allocated objects die
    - within a few million instructions
    - Before another megabyte is allocated
- Old objects tend to live a long…. time

Memory Spaces

- New Space
  - Where all objects are new (sort of…)
- Old Space
  - Where all objects are old

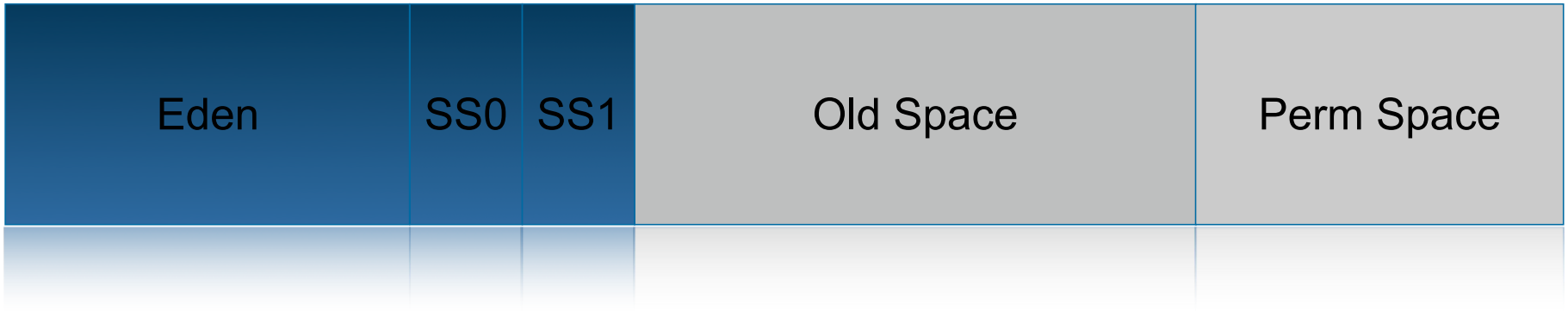
Memory Spaces



> New Space Division

- Eden
  - Where all objects are created
- Survivor Space 0
  - Provide object aging
- Survivor Space 1

Perm Spaces



- Permanent Space
  - class information
  - static information

GC Responsibility

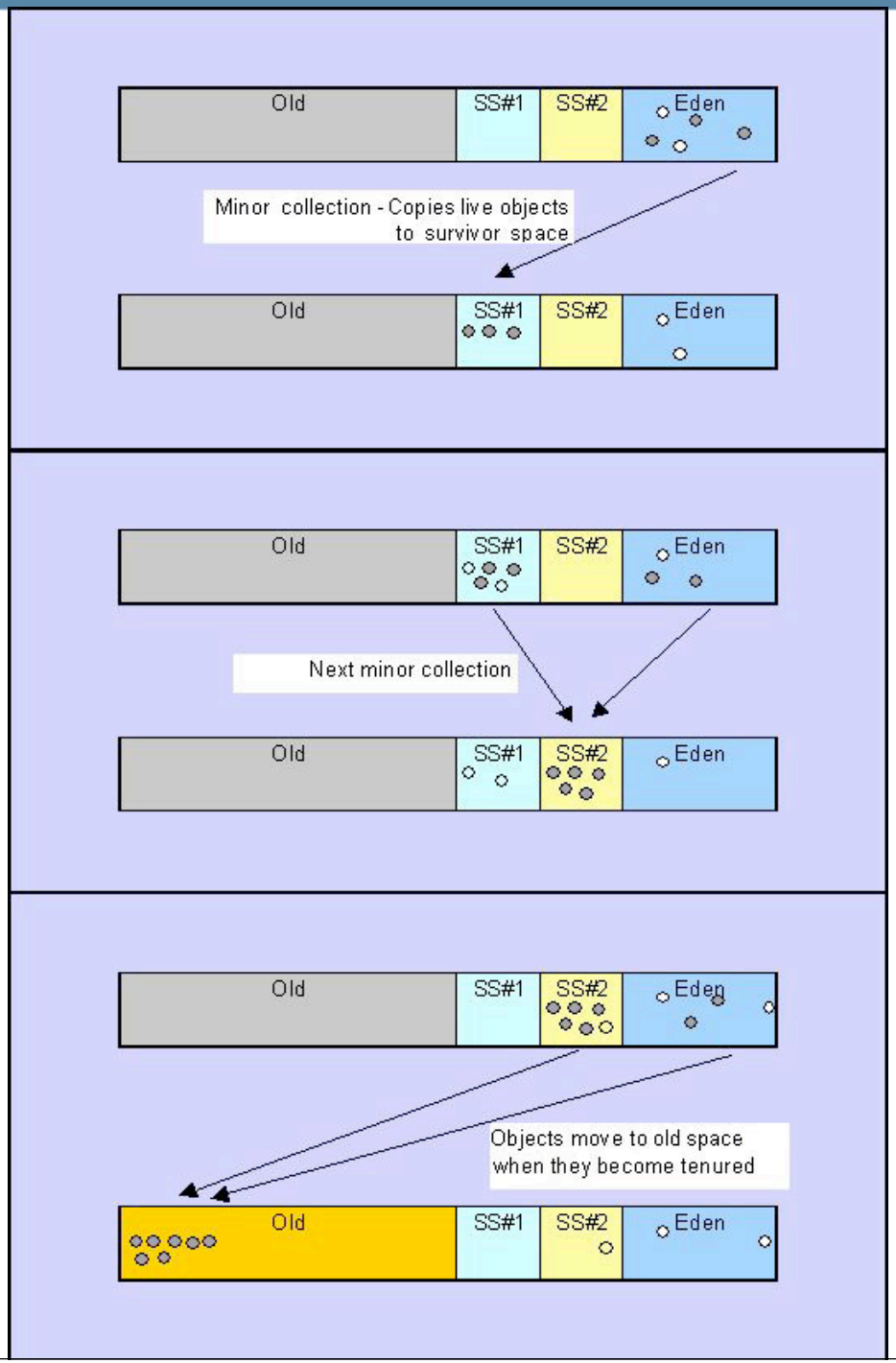
- Heap Walking from GC Roots
- Mark / Sweep
  - Garbage detection (mark)
    - Sort out the live ones from the dead ones
    - Reference counting
  - Garbage reclamation (sweep)
    - Make space available

Minor Garbage Collection

- Minor gc (scavenge)
  - When eden is “full” a minor gc is invoked
  - Sweeps through eden and the current survivor space, removing the dead and moving the living to survivor space or old
  - Ss0 and ss1 switch which is “current”
  - A new tenuring age is calculated

Major Garbage Collection

- Major gc
  - When old is “full”
  - All spaces are garbage collected including perm space
  - All other activities in the jvm are suspended


Old
SS#1
SS#2
Eden
Minor collection - Copies live objects to survivor space
Old
SS#1
SS#2
Eden
Old
SS#1
SS#2
Eden
Next minor collection
Old
SS#1
SS#2
Eden
Old
SS#1
SS#2
Eden
Objects move to old space when they become tenured
Old
SS#1
SS#2
Eden

Agenda

- Java Memory Management
- Memory Management Tools
  - Command-line Tools
  - VisualVM
- Btrace
- Summary
- Resources

Java Memory Tools

- JPS
  - Getting the Process ID (PID)
- Jstat
  - jstat -gcutil <pid> 250 7

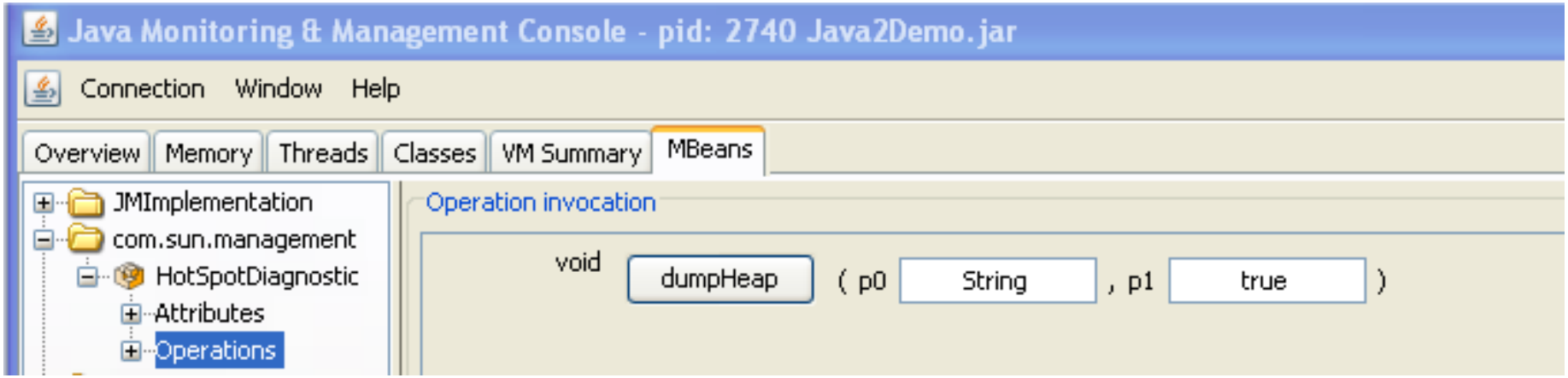
Looking at the Heap

- %JAVA_HOME%/bin/jmap – histo:live <pid>
  - Looking at all the “live” objects
- %JAVA_HOME%/bin/jmap – histo <pid>
  - Looking at all objects

- The difference between is the list of unreachable objects

Taking a Heap Dump

- %JAVA_HOME%/bin/jmap –dump:live,file=heap.out,format=b <pid>
  - Dumps the Heap to a file
- JConsole



JHat

- %JAVA_HOME%/bin/jhat <filename>
  - Starts a web server to investigate the heap
- Queries
  - Show instance count for all classes
  - Show Heap Histogram
  - Show Finalizer
  - Use the Execute Object Query Language (OQL)
    - select s from java.lang.String s where s.count >=100

JMX – Looking at Flags

Java Monitoring & Management Console - pid: 2740 Java2Demo.jar

Connection Window Help

Overview | Memory | Threads | Classes | VM Summary | MBeans

- JMImplementation
- com.sun.management
  - HotSpotDiagnostic
    - Attributes
    - Operations
- java.lang
- java.util.logging

Operation invocation

void	dumpHeap	(p0	heap2.out	, p1	true	)
CompositeData	getVMOption	(p0	tOfMemoryError	)		
void	setVMOption	(p0	tOfMemoryError	, p1	false	)

MAT – Memory Analyzer Tool

Eclipse Memory Analyzer

Inspector

- 0x106dd2f70
- Gregorian
- sun.util.calendar
- class sun.util.calendar.Gregorian @ 0x10
- sun.util.calendar.BaseCalendar
- java.lang.ClassLoader @ 0x0
- 24 (shallow size)
- 24 (retained size)
- no GC root

Statics | Attributes | Class Hierarchy

Type	Name	Value
ref	eras	null

heapdump-1244002540669.hprof

Overview | default_report org.eclipse.mat.api:suspects | dominator_tree | Histogram | dominator_tree

Class name	Shallow Heap	Retained Heap	Percentage
<Regex>	<Numeric>	<Numeric>	<Numeric>
sun.util.calendar.Gregorian @ 0x106dd2f70	24	24	0.00%
sun.text.normalizer.NormalizerImpl$NormTrieImpl @ 0x1072dbb38	16	16	0.00%
sun.text.normalizer.NormalizerImpl$FCDTrieImpl @ 0x1072d3898	16	16	0.00%
sun.text.normalizer.NormalizerImpl$AuxTrieImpl @ 0x1072dbb48	16	16	0.00%
sun.security.x509.X509CertImpl @ 0x1072c9228	128	9,944	0.09%
sun.security.x509.X509CertImpl @ 0x1072c1618	128	7,368	0.07%
sun.security.util.ObjectIdentifier @ 0x1072d0e60	40	88	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c8988	40	88	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c7a38	40	40	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c79e8	40	40	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c79c0	40	40	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c7998	40	40	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c7970	40	40	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c7948	40	40	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c7920	40	40	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c78f8	40	40	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c78d0	40	40	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c7858	40	40	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c7830	40	40	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c77e0	40	40	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c77b8	40	40	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c7790	40	40	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c7768	40	40	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c7740	40	40	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c7718	40	40	0.00%
sun.security.util.ObjectIdentifier @ 0x1072c76f0	40	40	0.00%
Total: 26 of 8,420 entries			

Notes | Navigation History

- OverviewPane
- default_report org.eclipse.mat.api:suspects

VisualVM

- Open Source All-in-One Java Troubleshooting tool
- https://visualvm.dev.java.net/

com.toy.anagrams.ui.Anagrams (pid 2096)

Overview | Monitor | Threads | Profiler | MBeans | heapdump-1204307575240.hprof

com.toy.anagrams.ui.Anagrams (pid 2096)

Profiler

Profile: CPU | Memory | Stop

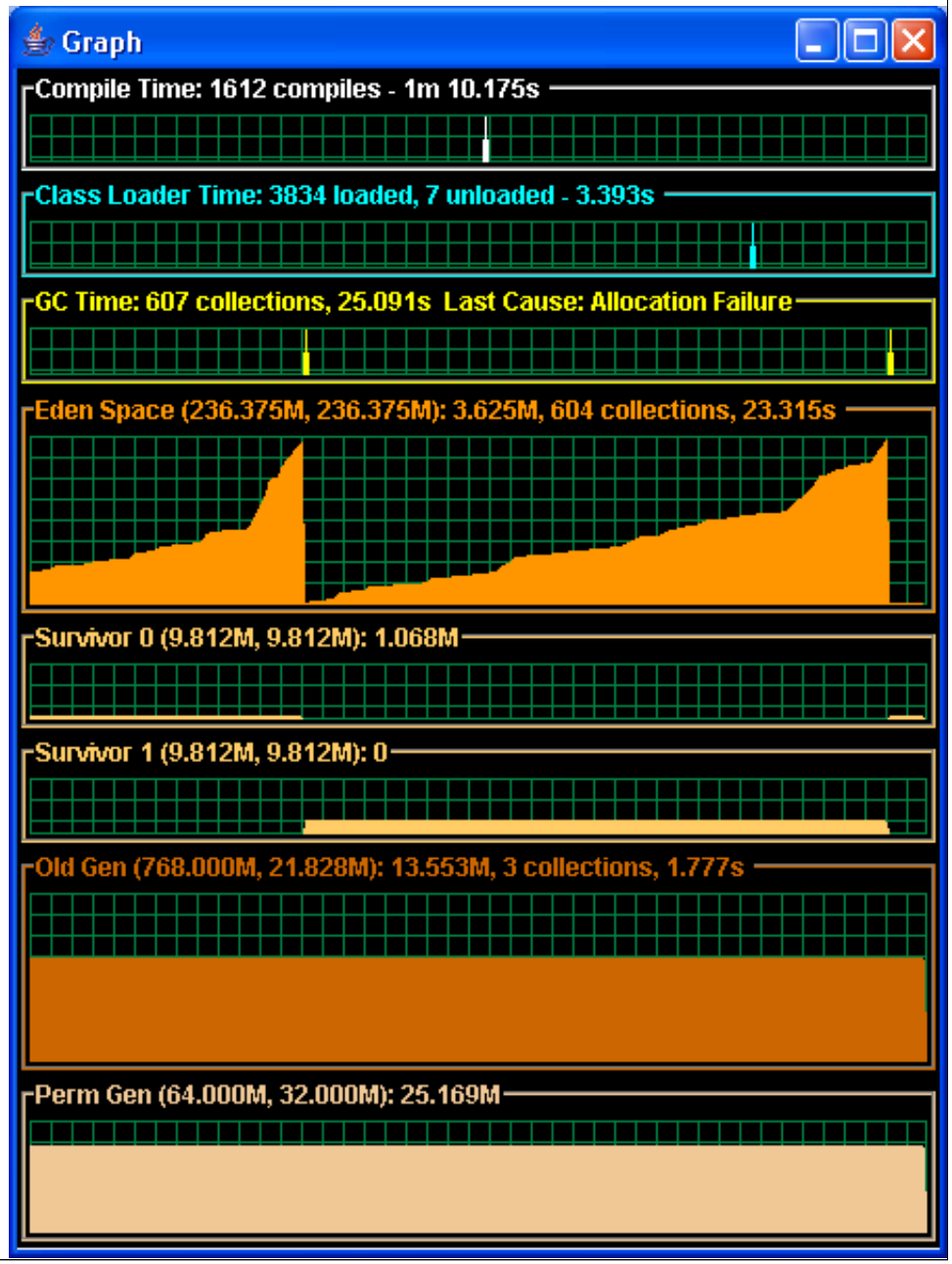
Status: profiling running (Analyze Memory)

Profiling results

Class Name - Allocated Objects	Bytes Allocated	Bytes Allocated	Objects Allocated
java.lang.**Object[]**		87 152 B (17%)	29 560 (23,8%)
java.io.**ObjectStreamClass$WeakClassKey**		80 384 B (15,7%)	23 876 (19,2%)
char[]		78 312 B (15,3%)	10 213 (8,2%)
java.util.**TreeMap$Entry**		65 984 B (12,9%)	19 533 (15,7%)
byte[]		41 456 B (8,1%)	1 975 (1,6%)
int[]		41 144 B (8%)	1 029 (0,8%)
java.lang.**String**		15 312 B (3%)	6 078 (4,9%)
java.util.**TreeMap**		11 280 B (2,2%)	2 193 (1,8%)

Visualgc

- visualgc <pid>
- Visual Garbage Collection Monitoring
  - a graphical tool for monitoring the HotSpot Garbage Collector, Compiler, and class loader. It can monitor both local and remote JVMs.



DEMO

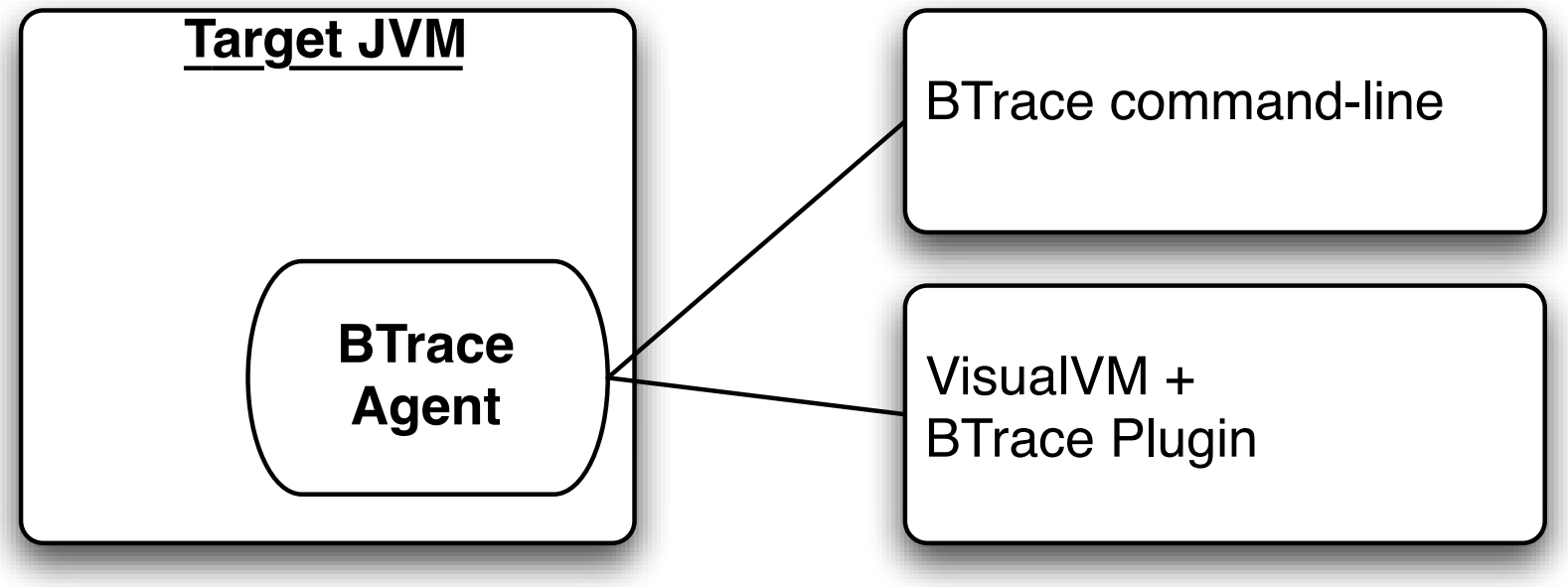
Agenda

- Java Memory Management
- Memory Management Tools
  - Command-line Tools
  - VisualVM
- Btrace
- Summary
- Resources

BTrace

- dynamically bytecode instrumenting (BCI)
  - read / not write
  - probe-based
  - observe running Java applications
- Integration with DTrace on Solaris
- http://btrace.dev.java.net
- visualvm plugin

BTrace Tools



BTrace Terminology

- Probe Point
  - “location” or “event” at which tracing statements are executed
- Trace Actions
  - statements which are executed whenever a probe fires
- Action Methods
  - static methods which define a trace

Probes and Actions

- Probe Targets
  - method entry / exit
  - line number
  - exceptions
    - return from method
    - exception throw (before)
  - synchronization entry / exit
- Actions
  - static methods in trace class

BTrace Restrictions

- no new objects
- no new arrays
- no exceptions
- no outer, inner, nested or local classes
- no synchronization blocks
- no loops
- no interfaces

Tracing

- Annotations
  - com.sun.btrace.annotations
  - @BTrace
    - denotes a btrace class
- Probe Points
  - @OnMethod
  - @OnTimer
  - @OnEvent
  - @OnExit

Simple Example: Looking for Object Size

```
package com.sun.btrace.samples;

import com.sun.btrace.annotations.*;
import static com.sun.btrace.BTraceUtils.*;

@BTrace public class Sizeof {
    @OnMethod(
        clazz="javax.swing.JComponent",
        method="<init>"
    )
    public static void onnew(Object obj) {
        println(concat("object of: ", name(classOf(obj))));
        println(concat("size: ", str(sizeof(obj))));
    }
}
```

Simple Example: Looking for Object Size

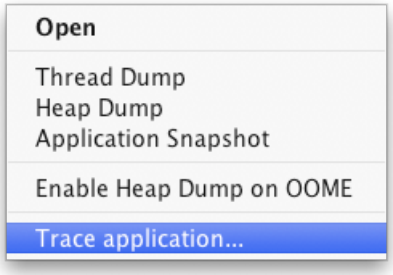
```
package com.sun.btrace.samples;

import com.sun.btrace.annotations.*;
import static com.sun.btrace.BTraceUtils.*;

@BTrace public class Sizeof {
    @OnMethod(
        clazz="javax.swing.JComponent",
        method="<init>"
    )
    public static void onnew(Object obj) {
        println(concat("object of: ", name(classOf(obj))));
        println(concat("size: ", str(sizeof(obj))));
    }
}
```

BTrace Run Options

- BTrace Command-Line
  - btrace
    - runs btrace tool (and compiles btrace script if necessary)
  - btracec
    - btrace script compiler
  - btracer
    - convenience script to start java project with tracing enabled at application startup
- VisualVM + BTrace Plugin



Testing the Probe

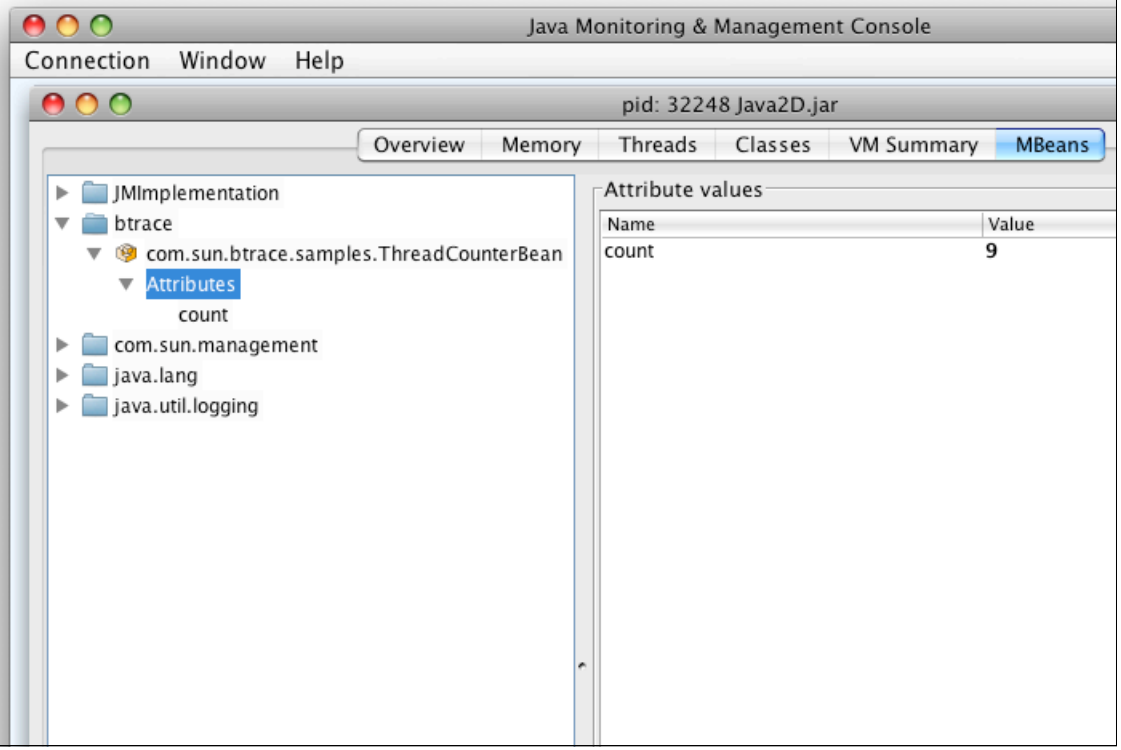
- Start Target Application
  - java -jar java2demo.jar
- Get the PID
  - jps
- Inject Probe
  - btrace <pid> Sizeof.java

BTrace + JMX Script

```
@BTrace public class ThreadCounterBean {

    // @Property makes the count field to be exposed
    // as an attribute of this MBean.
    @Property
    private static long count;

    @OnMethod(
        clazz="java.lang.Thread",
        method="start"
    )
    public static void onnewThread(Thread t) {
        count++;
    }
```



BTrace + jstat

```
@BTrace public class ThreadCounter {

    // create a jvmstat counter using @Export
    @Export private static long count;

    @OnMethod(
        clazz="java.lang.Thread",
        method="start"
    )
    public static void onnewThread(Thread t) {
        count++;
    }

    @OnTimer(2000)
    public static void ontimer() {
        println(perfLong("btrace.com.sun.btrace.samples.ThreadCounter.count"));
    }
}
```

Terminal — bash — 150×27

```
ibm-kondapaneni:~ kensipe$ jps
30996
32271 jar
32307 Jps
32277 Main
ibm-kondapaneni:~ kensipe$ jstat -J-Djstat.showUnsupported=true -name btrace.com.sun.btrace.samples.ThreadCounter.count 32271
btrace.com.sun.btrace.samples.ThreadCounter.count
1
ibm-kondapaneni:~ kensipe$
```

ThreadLocal

```
@BTrace public class OnThrow {
    // store current exception in a thread local
    // variable (@TLS annotation). Note that we can't
    // store it in a global variable!
    @TLS static Throwable currentException;

    @OnMethod(
        clazz="java.lang.Throwable",
        method="<init>"
    )
    public static void onthrow(Throwable self) {
        currentException = self;
    }

    // when any constructor of java.lang.Throwable returns
    // print the currentException's stack trace.
    @OnMethod(
        clazz="java.lang.Throwable",
        method="<init>",
        location=@Location(Kind.RETURN)
    )
    public static void onthrowreturn() {
        if (currentException != null) {
            jstack(currentException);
            println("=====================");
            currentException = null;
        }
    }
```

BTrace Events

```
@OnEvent
public static void onEvent() {

    // Top 10 queries only
    BTraceUtils.truncateAggregation(histogram, 10);

    println("-----------------------------------------");
    printAggregation("Count", count);
    printAggregation("Min", min);
    printAggregation("Max", max);
    printAggregation("Average", average);
    printAggregation("Sum", sum);
    printAggregation("Histogram", histogram);
    printAggregation("Global Count", globalCount);
    println("-----------------------------------------");
}
```

```
DEBUG: received com.sun.btrace.comm.MessageCommand@2e1474fc
0
^CPlease enter your option:
        1. exit
        2. send an event
        3. send a named event
DEBUG: received com.sun.btrace.comm.MessageCommand@72cc5002
```

Checking Synchronization Entry / Exits

```
@BTrace public class AllSync {
    @OnMethod(
        clazz="/javax\\.swing\\..*/",
        method="/.*/",
        location=@Location(value=Kind.SYNC_ENTRY, where=Where.AFTER)
    )
    public static void onSyncEntry(Object obj) {
        println(strcat("after synchronized entry: ", identityStr(obj)));
    }

    @OnMethod(
        clazz="/javax\\.swing\\..*/",
        method="/.*/",
        location=@Location(Kind.SYNC_EXIT)
    )
    public static void onSyncExit(Object obj) {
        println(strcat("before synchronized exit: ", identityStr(obj)));
    }
}
```

Tracing Opportunities

- Thread Monitors
- Socket / Web Services
- Object Creation and Size
- File Access

DEMO

Known Solutions with BTrace

- Terracotta
  - concurrency issue
- Hibernate / Atlassian Issue
  - thread / session management issue
  - http://jira.atlassian.com/browse/CONF-12201

Summary

- jmap, jhat, jconsole, jstat
  - tools already in the JDK bin directory
- VisualVM
  - swiss army knife for JVM process monitoring and tracing
- BTrace
  - dynamic tracing tool

Resources

- Performance and Troubleshooting
  - http://java.sun.com/performance/reference/whitepapers/6_performance.html
  - http://java.sun.com/javase/6/webnotes/trouble/TSG-VM/html/docinfo.html
- VisualVM
  - https://visualvm.dev.java.net/
- BTrace
  - https://btrace.dev.java.net/

JavaOne℠

Thank You

Ken Sipe
http://kensipe.blogspot.com

twitter: @kensipe