12.–15.09.2010
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Mist gemessen?

Java Performance und Memory Analyse

Dr. Halil-Cem Gürsoy

adesso AG

Der Referent

- Insgesamt über 10 Jahre Beratung und Entwicklung rund um Java, davor Entwicklung im Forschungsumfeld
- Senior Software Engineer bei adesso AG, Dortmund
- Schwerpunkt EAI und SOA-Projekte im JEE Umfeld, aber auch klassische JEE-Projekte
- Autor und Referent auf Konferenzen

Agenda

- Über das Messen
- Performance
- Messungen bei Performance-Problemen
  - Manuell
  - JVisualVM
- Memory-Probleme
  - JVisualVM
  - MAT
  - IBM Heap Dump Analyzer

Agenda

- Über das Messen
- Performance
- Messungen bei Performance-Problemen
  - Manuell
  - JVisualVM
- Memory-Probleme
  - JVisualVM
  - MAT
  - IBM Heap Dump Analyzer

Was ist „Messen“?

Eine Messung ist das Ausführen von geplanten Tätigkeiten zu einer quantitativen Aussage über eine Messgröße durch Vergleich mit einer Einheit. Dabei ist die Messgröße jene physikalische Größe, der die Messung gilt. Die Bezeichnungen für die Messtechnik werden in der DIN-Norm DIN 1319 definiert.

Quelle: wikipedia.de

Wie wird gemessen?

- Was ist das Messproblem?
- In welcher Einheit wird gemessen?
- Messeinrichtung / Messgerät
- Kalibirierung
- Messablauf definieren
- Durchführung und Messergebnis
- Fehlerrechnung (Messunsicherheiten, Fehlergrenzen usw.)

Ein paar Anregungen…

- Ein beobachtetes System verändert sich durch die Messung
- Heisenbergsche Unschärferelation
  - Ort und Impuls nicht gleichzeitig exakt messbar

$$\sigma_x \sigma_p \geq \frac{\hbar}{2}$$

- „Quantenselbsmord“
  - Schrödingers Katze etwas weiterentwickelt
  - Der selbstmordende Wissenschaftlicher ist am Ende „nur ein bisschen Tot“
  - Also lebt Elvis doch… wenn auch nur ein bisschen!

Agenda

- Über das Messen
- Performance
- Messungen bei Performance-Problemen
  - Manuell
  - JVisualVM
- Memory-Probleme
  - JVisualVM
  - MAT
  - IBM Heap Dump Analyzer

"Performance"

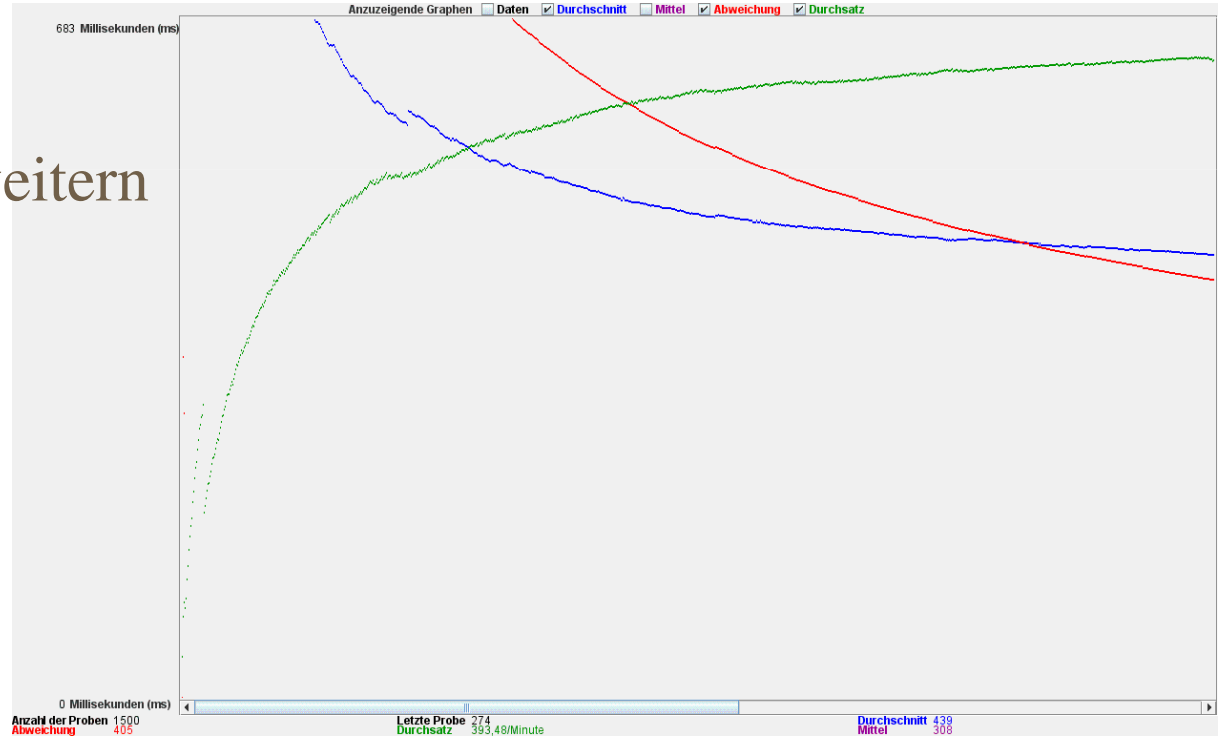
- WAS ist „Performance“?
- „Gefühlte Performance“
- Durchsatz – z.B.
  - Transaktionen / sec.
  - Anzahl verarbeitete Nachrichten,
  - Einhaltung von „Zeitfenstern“
  - Häufig nichtfunktionale Anforderungen im Projekt
- Viele Wege zum Messen…

Toolbasiertes Stressen & Messen

- Diverse freie Tools zum „Vermessen“ von Applikationen:
  - JMeter
  - Grinder
  - OpenSTA
  - Derivate von JUnit
- Messungen eher „Grobgranular“
- Aufruf „Seite XY“ oder „ServiceZ“ dauert zu lange

JMeter…

- …kann mehr als nur Web-Applikationen vermessen:
  - Datenbankzugriffe
  - JMS
  - Webservices
- Leicht zu erweitern



Agenda

- Über das Messen
- Performance
- Messungen bei Performance-Problemen
  - Manuell
  - JVisualVM
- Memory-Probleme
  - JVisualVM
  - MAT
  - IBM Heap Dump Analyzer

Ursachenforschung…

- Tatsächlich alles richtig gemessen… (?)
- Ergebnis: es gibt tatsächlich ein Performance-Problem
- Was ist die Ursache?

Wo geht die Zeit verloren?

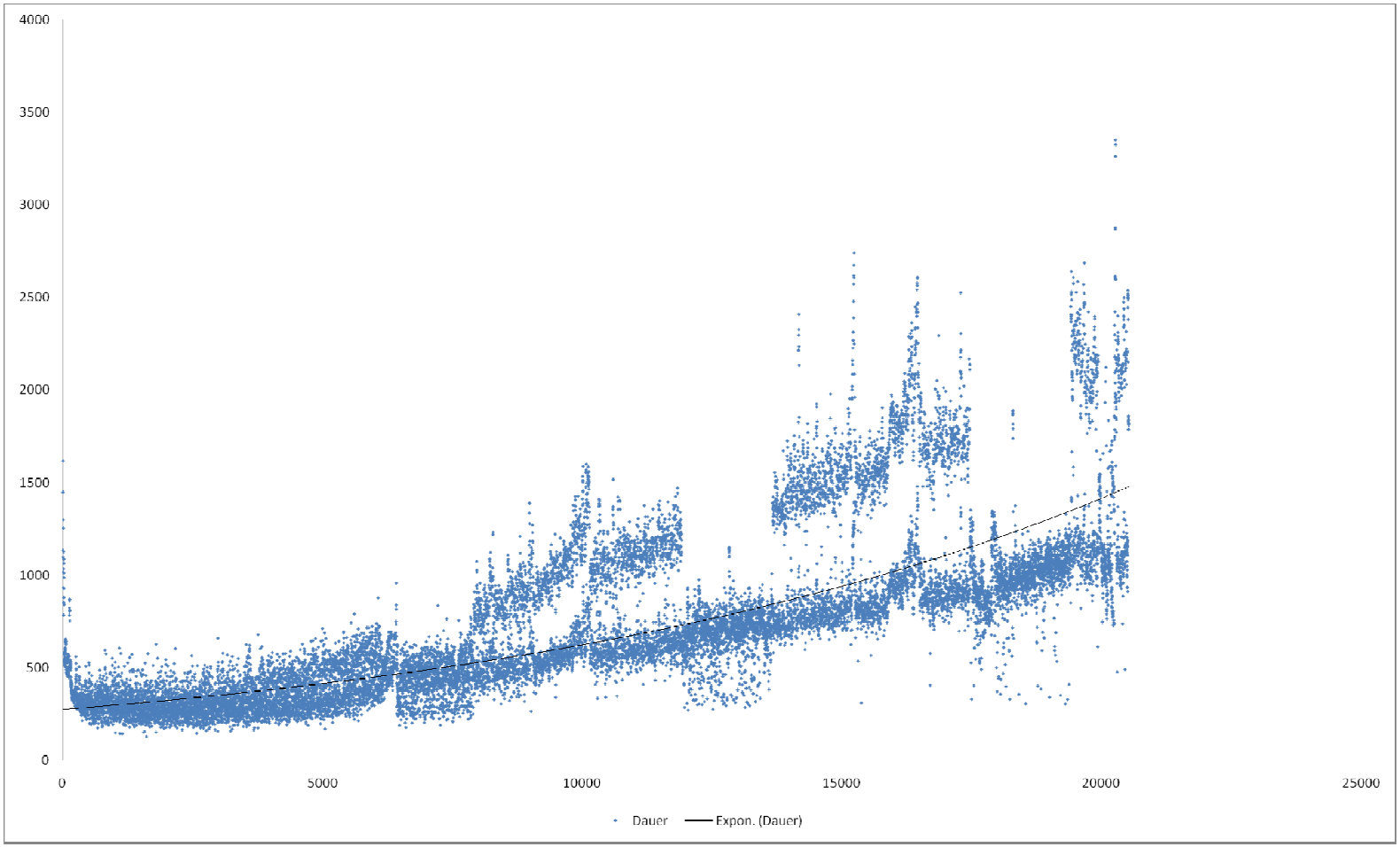
- Es kann viele Ursachen geben
- „Schlechter Code“
  - Blick in diverse Antipattern Blogs / Artikel / Bücher
  - Klassiker
    - *String*, *StringBuffer*
    - *compareTo()*
    - *resultSet.close()*
    - XML-Parser
    - JNDI Lookups
    - JDBC

Manuell Messen

- Auswertung von Logfiles
  - Zeiten zwischen Logausgaben
  - Gezielte Logausgaben mit Zeitangaben
  - AOP / Interceptoren
  - Aufwändige Analyse, schwer Daten zu aggregieren

```
@|03.09.2010 16:59:15.899|[[ACTIVE] ExecuteThread: '6' for queue:
'weblogic.kernel.Default (self-
tuning)']|DEBUG|CorrID[9d4e92197bc54b6f]|de.adesso.hgu.hcex.busines
s.interceptor.ProfilingInterceptor|profile|46|Methode 'public long
de.adesso.hgu.hcex.facade.SimpleBean.validate(de.adesso.hgu.hcex.ut
il.MDCWrapper,java.lang.String) throws
de.adesso.hgu.hcex.business.exception.ValidationException'
ausgefuehrt in 76ms
```

Eine Datenbank mit I/O-Problemen



Agenda

- Über das Messen
- Performance
- Messungen bei Performance-Problemen
  - Manuell
  - JVisualVM
- Memory-Probleme
  - JVisualVM
  - MAT
  - IBM Heap Dump Analyzer

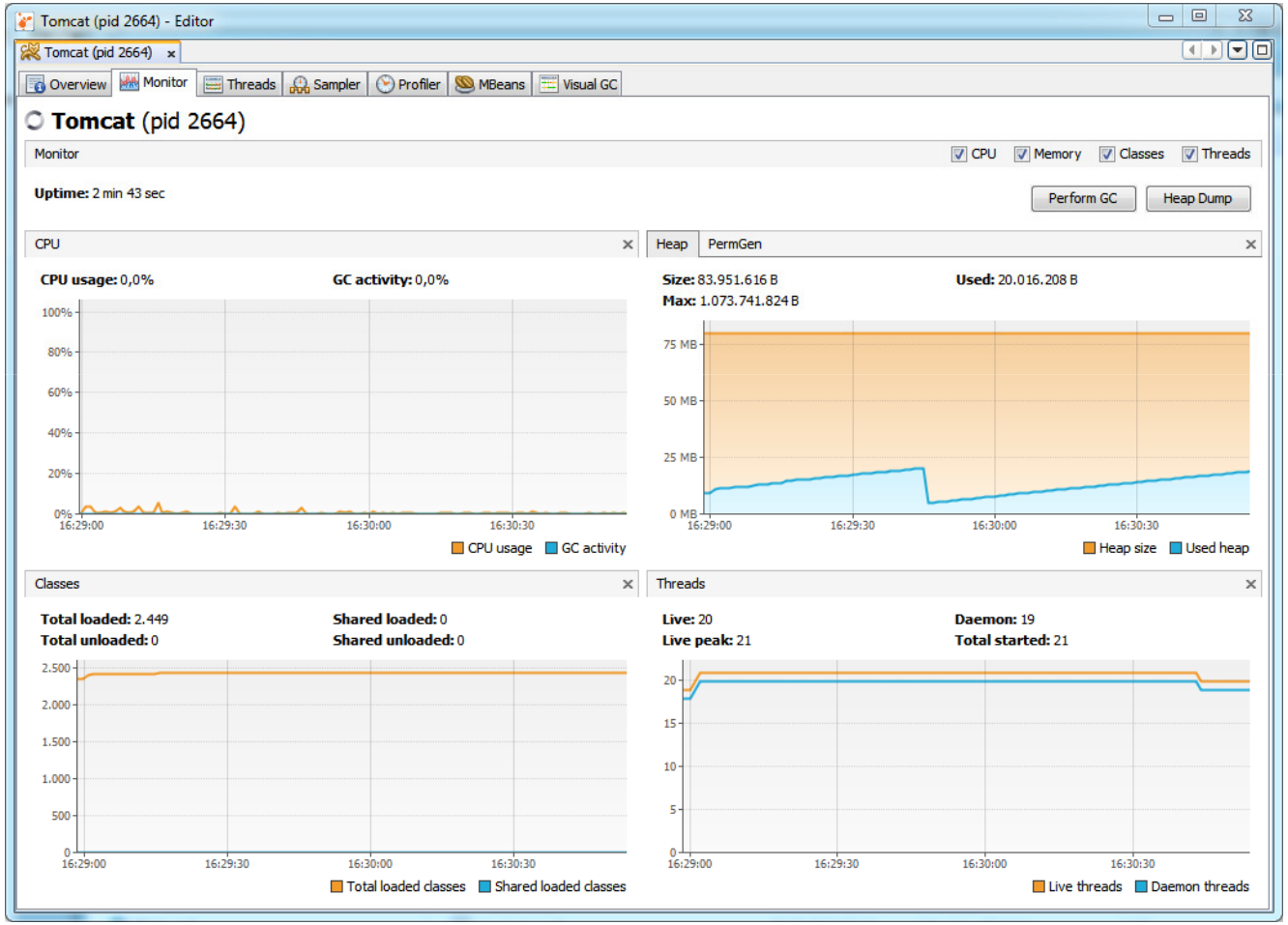
JVisualVM

- Es gibt viele Tools, z.B. JProbe u.a.
- JVM TI ist neue Schnittstelle innerhalb der JVM
- Geschenkt: JVisiualVM
- Teil der SUN JDK und unter
  https://visualvm.dev.java.net/
- Features
  - Konfiguration der Applikations-Runtime
  - Performance Monitor
  - Memory Monitor
  - Thread Monitor
  - Offline Analyse

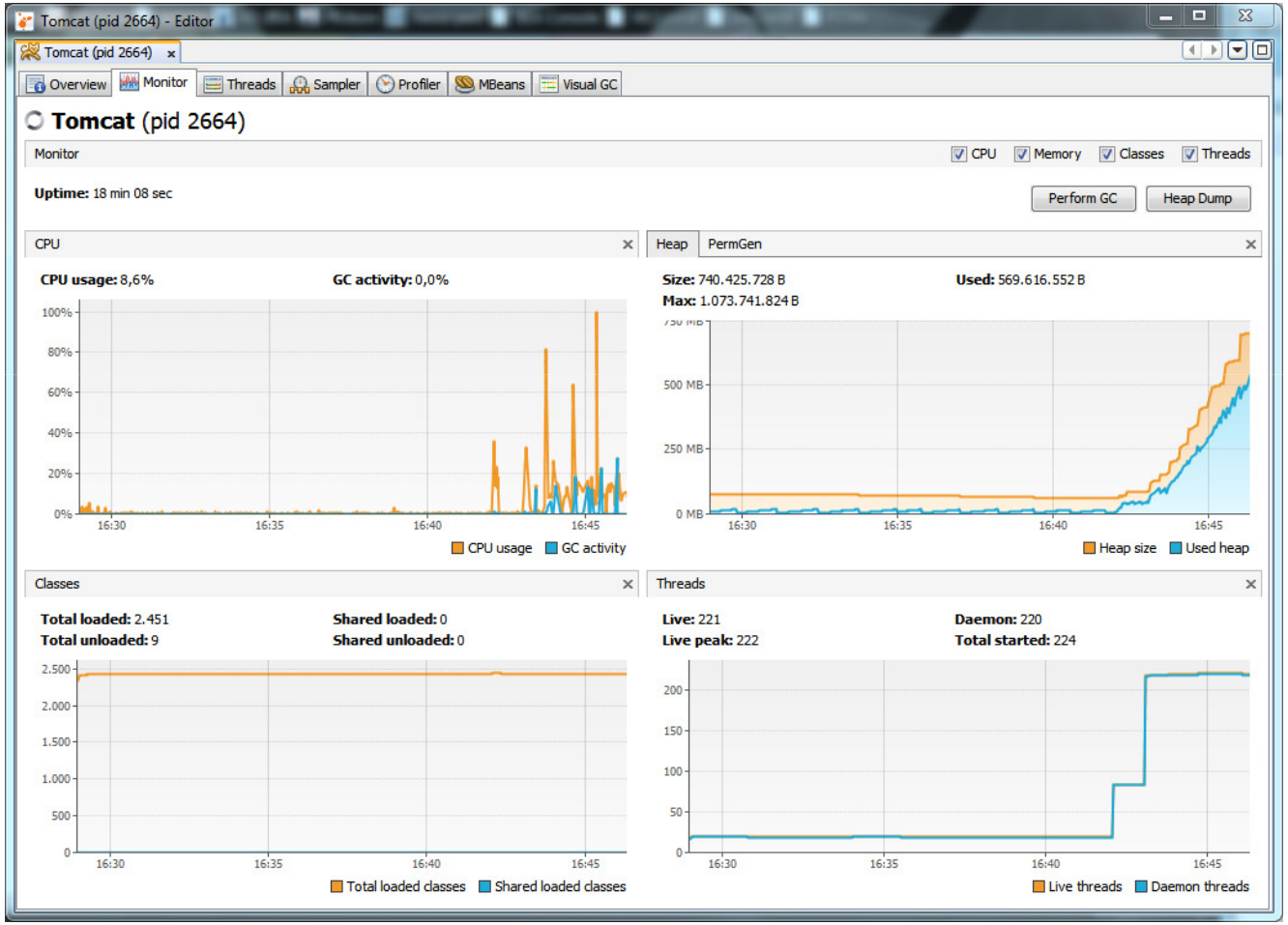
JVisualVM – Verbinden

- Lokale, laufende JVM‘s unter dem gleichen User werden automatisch gefunden
- Remote-JVM‘s
  - Starten von jstatd auf dem Remote-System. Muss unter dem gleichen User wie die JVM laufen
  - Starten der Applikation mit den Systemparametern
    - *com.sun.management.jmxremote.port*
    - *com.sun.management.jmxremote.ssl*
    - *com.sun.management.jmxremote.authenticate*

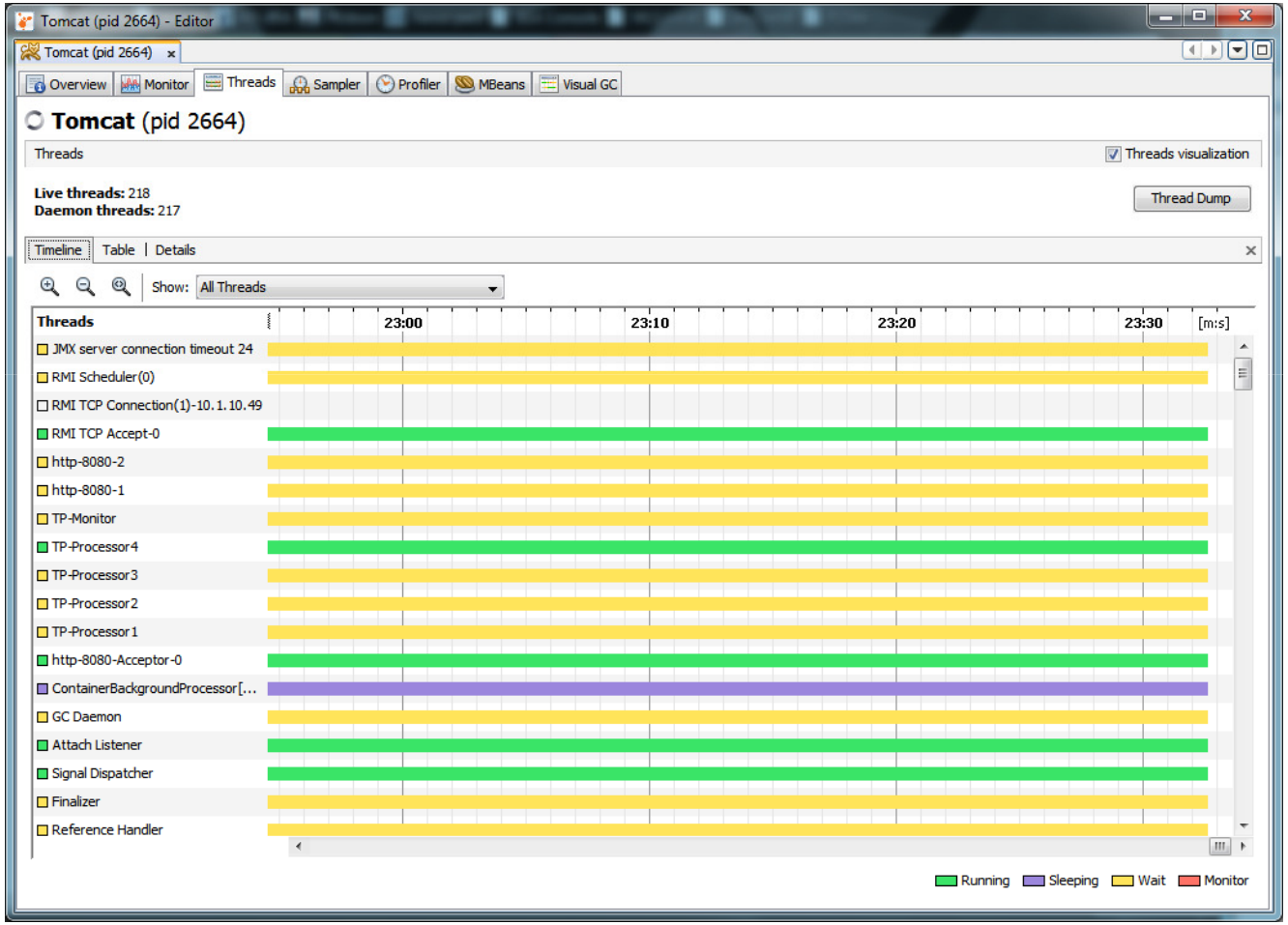
Monitoren der Basisdaten



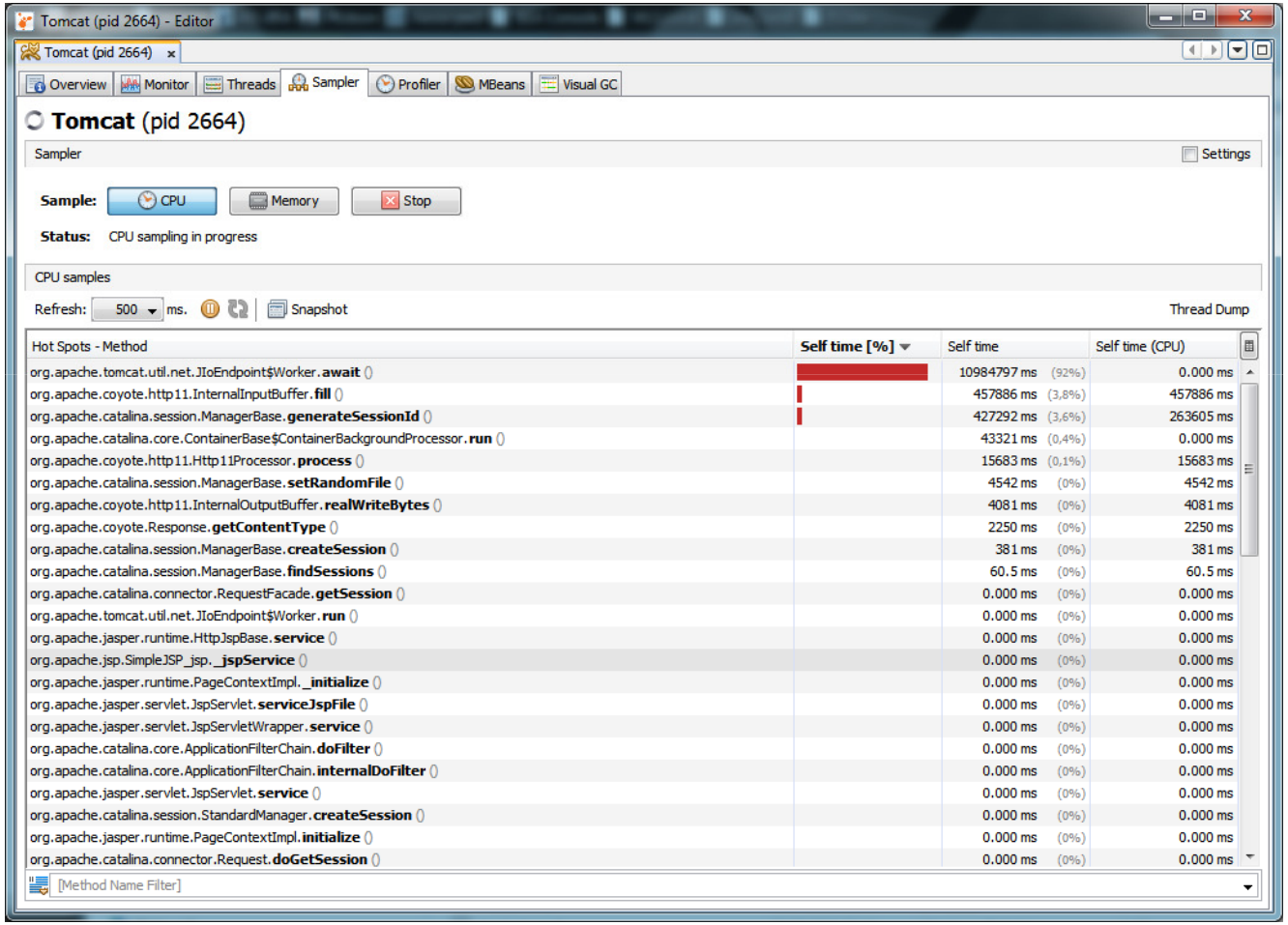
Tomcat unter Last



Threads


Tomcat (pid 2664) - Editor
Tomcat (pid 2664)
Overview
Monitor
Threads
Sampler
Profiler
MBeans
Visual GC
Tomcat (pid 2664)
Threads
Threads visualization
Live threads: 218
Daemon threads: 217
Thread Dump
Timeline
Table | Details
Show: All Threads
Threads
23:00
23:10
23:20
23:30
[m:s]
JMX server connection timeout 24
RMI Scheduler(0)
RMI TCP Connection(1)-10.1.10.49
RMI TCP Accept-0
http-8080-2
http-8080-1
TP-Monitor
TP-Processor4
TP-Processor3
TP-Processor2
TP-Processor1
http-8080-Acceptor-0
ContainerBackgroundProcessor[...
GC Daemon
Attach Listener
Signal Dispatcher
Finalizer
Reference Handler
Running
Sleeping
Wait
Monitor

Sampler

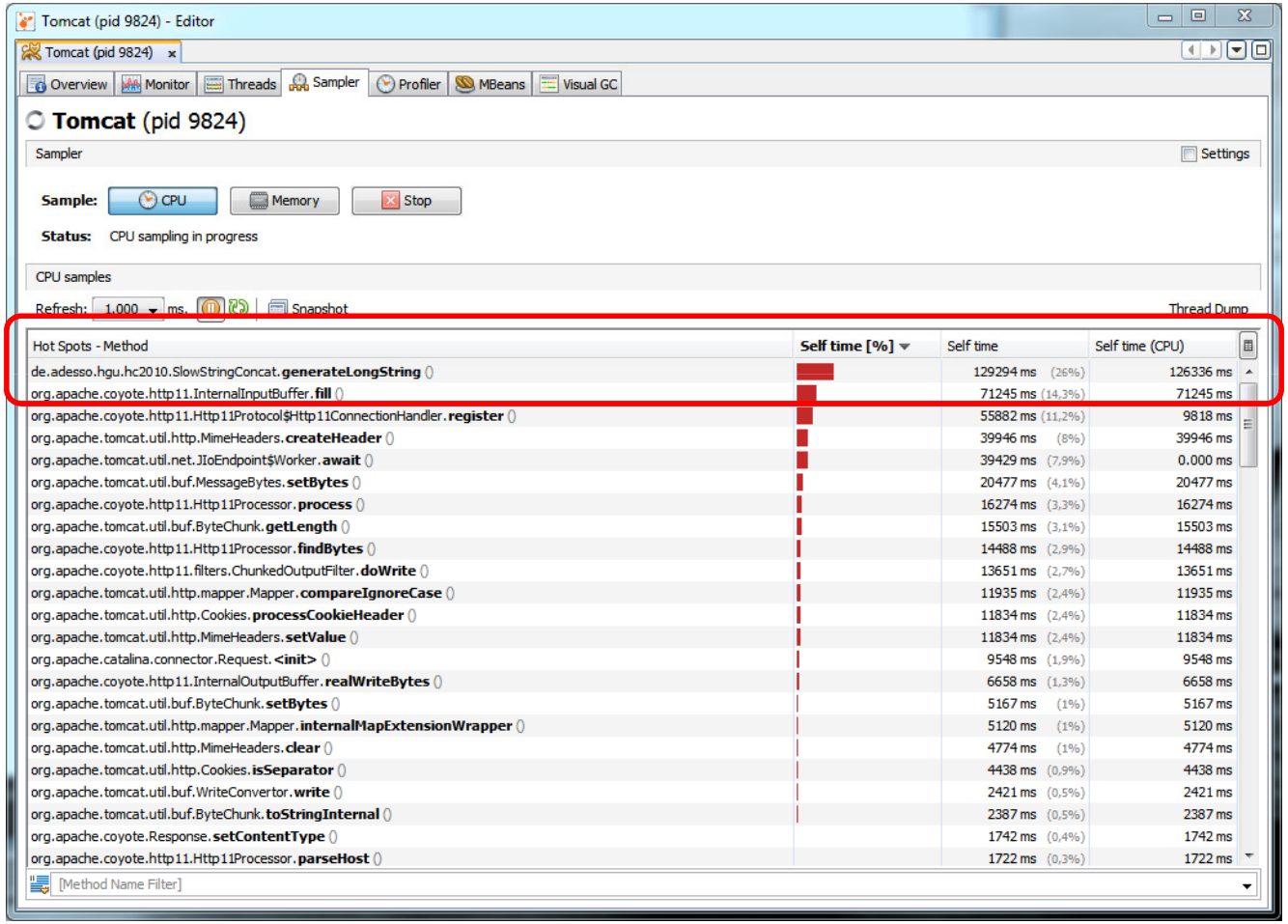

Tomcat (pid 2664) - Editor
Tomcat (pid 2664)
Overview
Monitor
Threads
Sampler
Profiler
MBeans
Visual GC
Tomcat (pid 2664)
Sampler
Settings
Sample:
CPU
Memory
Stop
Status:
CPU sampling in progress
CPU samples
Refresh:
500
ms.
Snapshot
Thread Dump
Hot Spots - Method
Self time [%]
Self time
Self time (CPU)
org.apache.tomcat.util.net.JIoEndpoint$Worker.await () 10984797 ms (92%) 0.000 ms
org.apache.coyote.http11.InternalInputBuffer.fill () 457886 ms (3,8%) 457886 ms
org.apache.catalina.session.ManagerBase.generateSessionId () 427292 ms (3,6%) 263605 ms
org.apache.catalina.core.ContainerBase$ContainerBackgroundProcessor.run () 43321 ms (0,4%) 0.000 ms
org.apache.coyote.http11.Http11Processor.process () 15683 ms (0,1%) 15683 ms
org.apache.catalina.session.ManagerBase.setRandomFile () 4542 ms (0%) 4542 ms
org.apache.coyote.http11.InternalOutputBuffer.realWriteBytes () 4081 ms (0%) 4081 ms
org.apache.coyote.Response.getContentType () 2250 ms (0%) 2250 ms
org.apache.catalina.session.ManagerBase.createSession () 381 ms (0%) 381 ms
org.apache.catalina.session.ManagerBase.findSessions () 60.5 ms (0%) 60.5 ms
org.apache.catalina.connector.RequestFacade.getSession () 0.000 ms (0%) 0.000 ms
org.apache.tomcat.util.net.JIoEndpoint$Worker.run () 0.000 ms (0%) 0.000 ms
org.apache.jasper.runtime.HttpJspBase.service () 0.000 ms (0%) 0.000 ms
org.apache.jsp.SimpleJSP_jsp._jspService () 0.000 ms (0%) 0.000 ms
org.apache.jasper.runtime.PageContextImpl._initialize () 0.000 ms (0%) 0.000 ms
org.apache.jasper.servlet.JspServlet.serviceJspFile () 0.000 ms (0%) 0.000 ms
org.apache.jasper.servlet.JspServletWrapper.service () 0.000 ms (0%) 0.000 ms
org.apache.catalina.core.ApplicationFilterChain.doFilter () 0.000 ms (0%) 0.000 ms
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter () 0.000 ms (0%) 0.000 ms
org.apache.jasper.servlet.JspServlet.service () 0.000 ms (0%) 0.000 ms
org.apache.catalina.session.StandardManager.createSession () 0.000 ms (0%) 0.000 ms
org.apache.jasper.runtime.PageContextImpl.initialize () 0.000 ms (0%) 0.000 ms
org.apache.catalina.connector.Request.doGetSession () 0.000 ms (0%) 0.000 ms
[Method Name Filter]

Tomcat…

- Simple Methode um CPU und Speicher zu verbrauchen:

```
public static String generateLongString() {
   String result = new String();
   for (int i = 0; i < 10000; i++) {
       result = result + "dummy" + i + "<br>";
   }
   return result;
}
```

Sampler-Ergebnis



Agenda

- Über das Messen
- Performance
- Messungen bei Performance-Problemen
  - Manuell
  - JVisualVM
- Memory-Probleme
  - JVisualVM
  - MAT
  - IBM Heap Dump Analyzer

Memory Probleme

- Häufig treten Memory und Performance-Probleme gemeinsam auf
- Memory-Probleme werden in der Entwicklung noch häufiger ignoriert als Performance-Probleme
  - Testumgebungen laufen nicht lang genug
  - zu wenig Testdaten
- Analyse in einer produktiven Umgebung schwierig
- Häufig beobachtete Dauer-Zwischenlösung
  - Application Server Reboots (täglich, stündlich…)

Heap Dumps

- Wichtiges Werkezeug um aktuelle Speichersitaution einzufangen und zu analysieren
- Heap Dumps erzeugen mit…
  - -XX:-HeapDumpOnOutOfMemoryError und dazugehörige Optionen
    - erzeugt Heap Dumps bei einer OOME
  - Jmap , JConsole, JVisualVM – Teile der SUN JDK
  - Über JMX
  - Tastenkombinationen

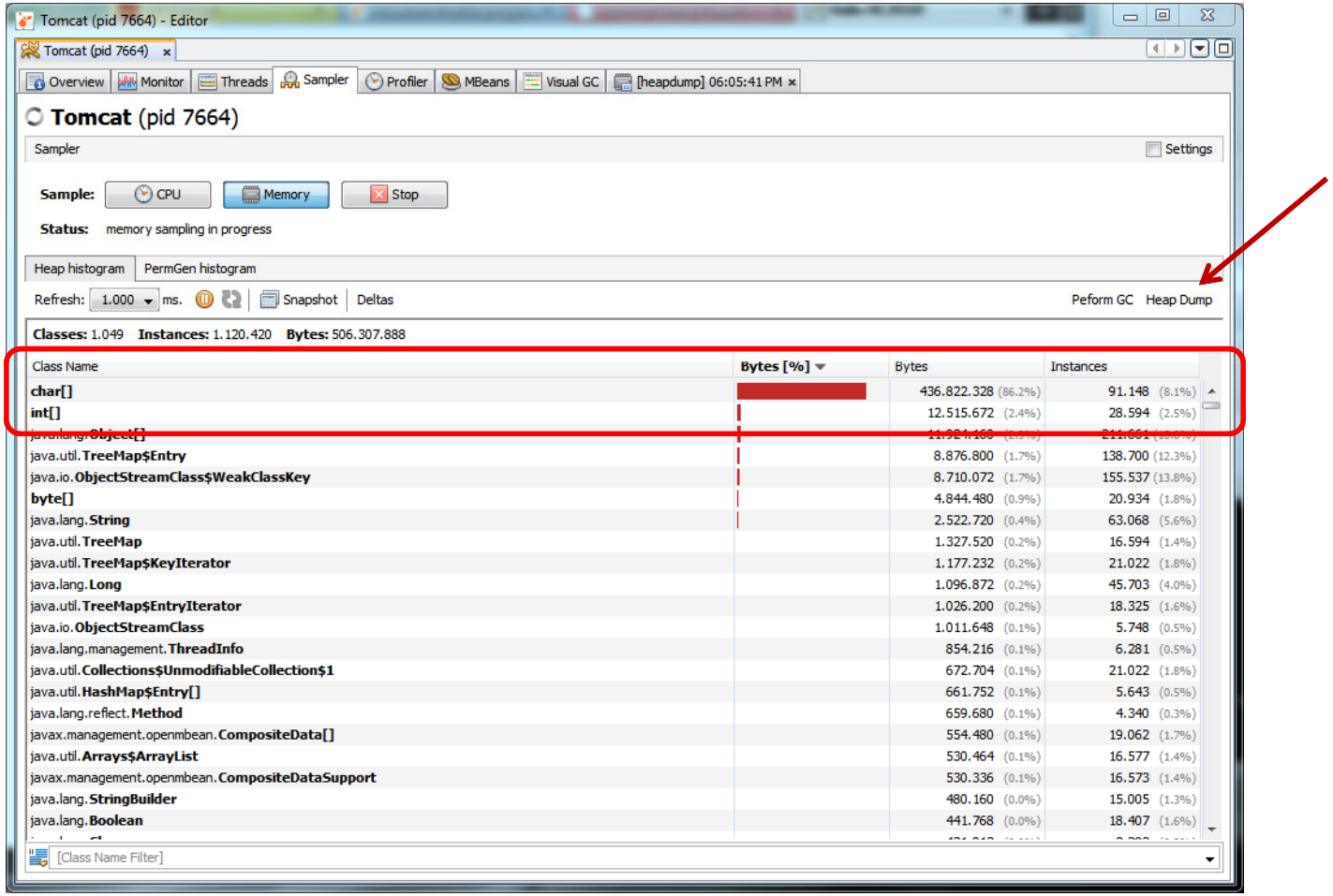
GC und Performance

- GC kostet CPU-Zeit
- Sichtbar: „die Welt steht“ („Stop the World“)
- Es gibt verschiedene GC‘s
  - leider immer noch eine Erwähnung wert!
  - Default-Einstellungen abhängig von System, CPU & Speicher
- Abhängig vom Einsatzszenario
- Durchsatzoptimiert = „Stop the World“
- Ziel: Minimierte Pausen
  - kostet aber Durchsatz!
  - auch in G1 (wenn auch kurz)

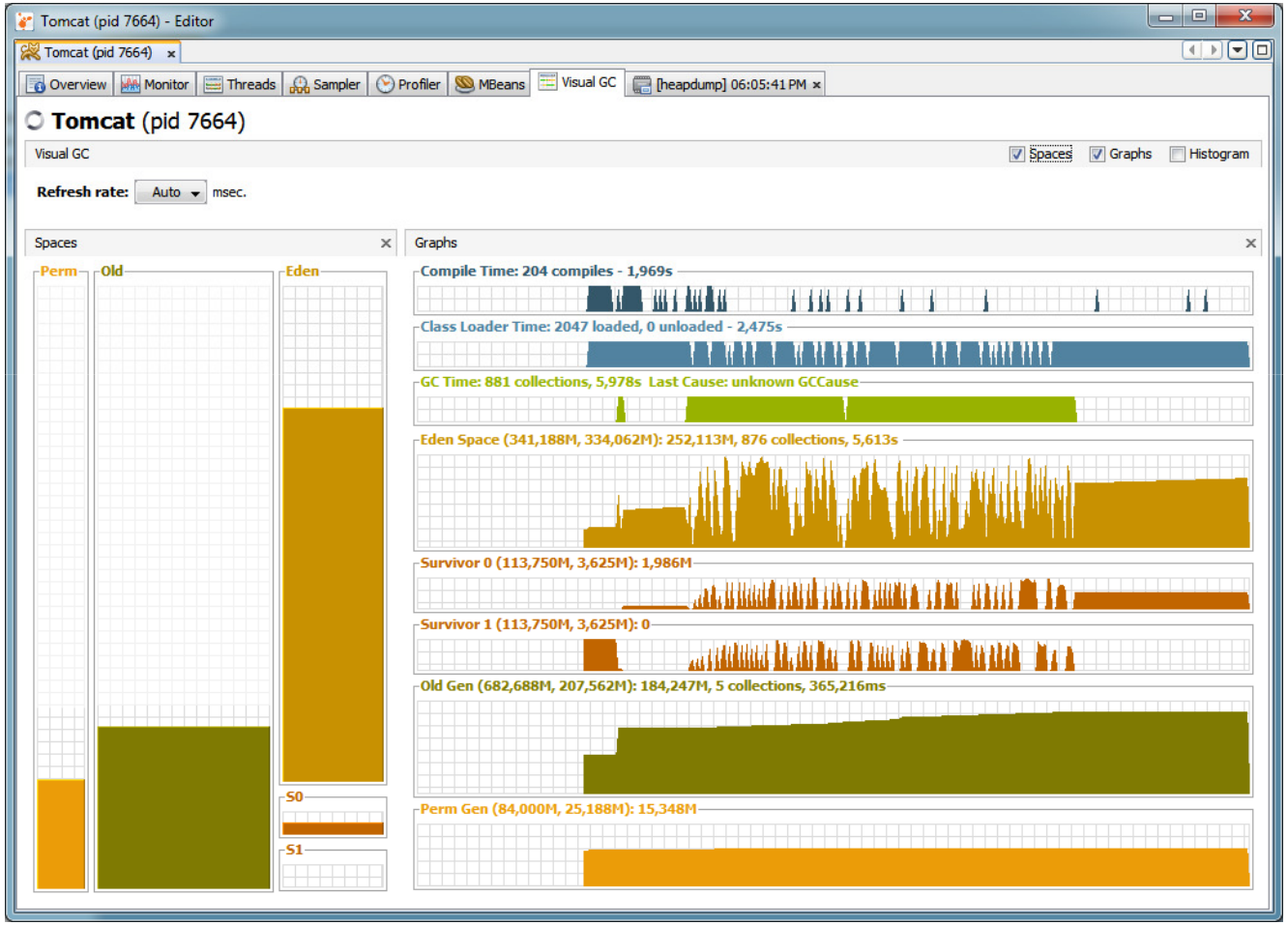
Agenda

- Über das Messen
- Performance
- Messungen bei Performance-Problemen
  - Manuell
  - JVisualVM
- Memory-Probleme
  - JVisualVM
  - MAT
  - IBM Heap Dump Analyzer

Sampler… String in Session



GC in JVisualVM



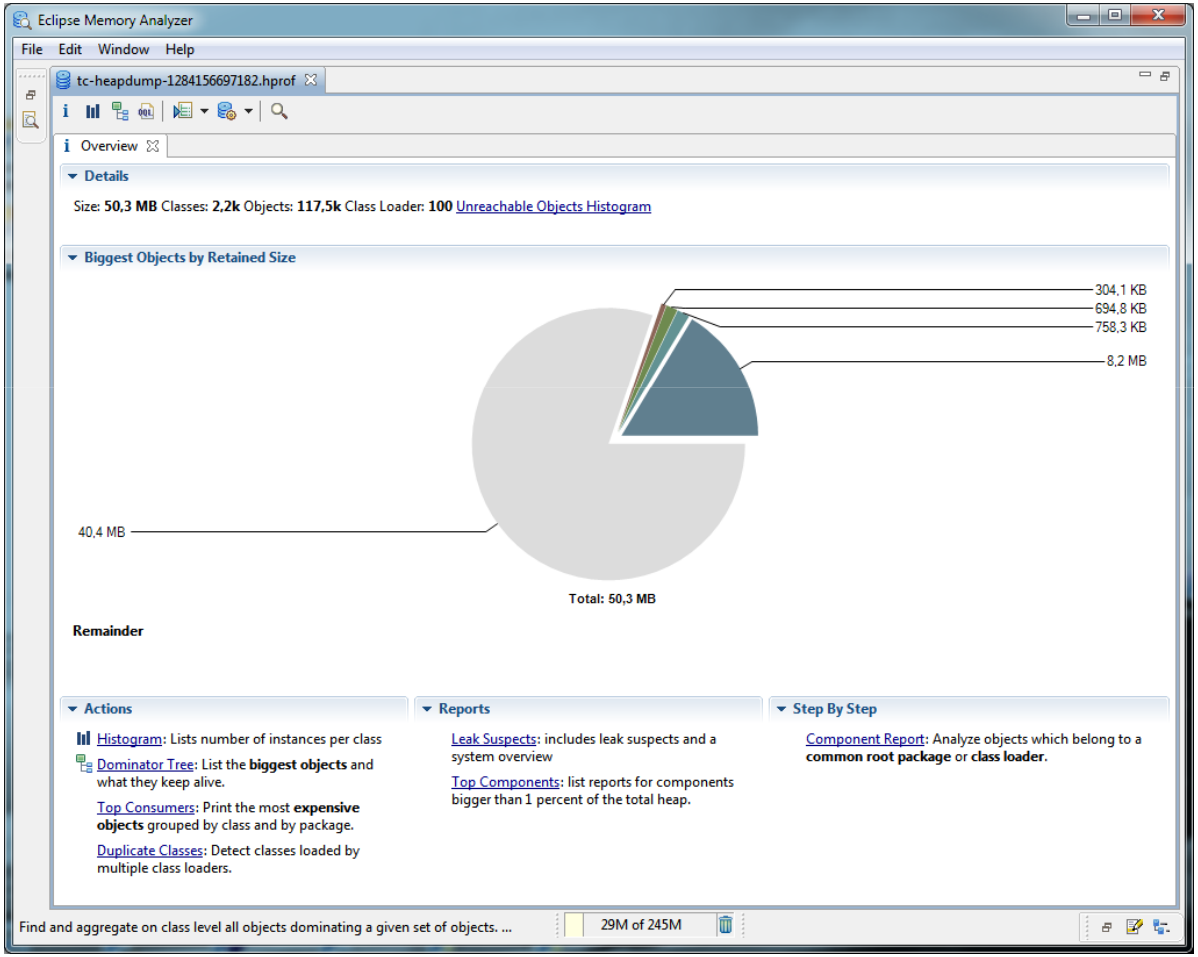
Agenda

- Über das Messen
- Performance
- Messungen bei Performance-Problemen
  - Manuell
  - JVisualVM
- Memory-Probleme
  - JVisualVM
  - MAT
  - IBM Heap Dump Analyzer

Eclipse Memory Analyzer (MAT)

- Standalone (RCP) und als Plugin
- Bei großen Heap Dumps RCP mit hohen Memory-Settings verwenden, sonst schnell OOME !
- Kann Heap Dumps von SUN & IBM JDK‘s auswerten
- Ähnlich JVisualVM anfordern von Heap Dumps
- Unterstützt OQL (Object Query Language)

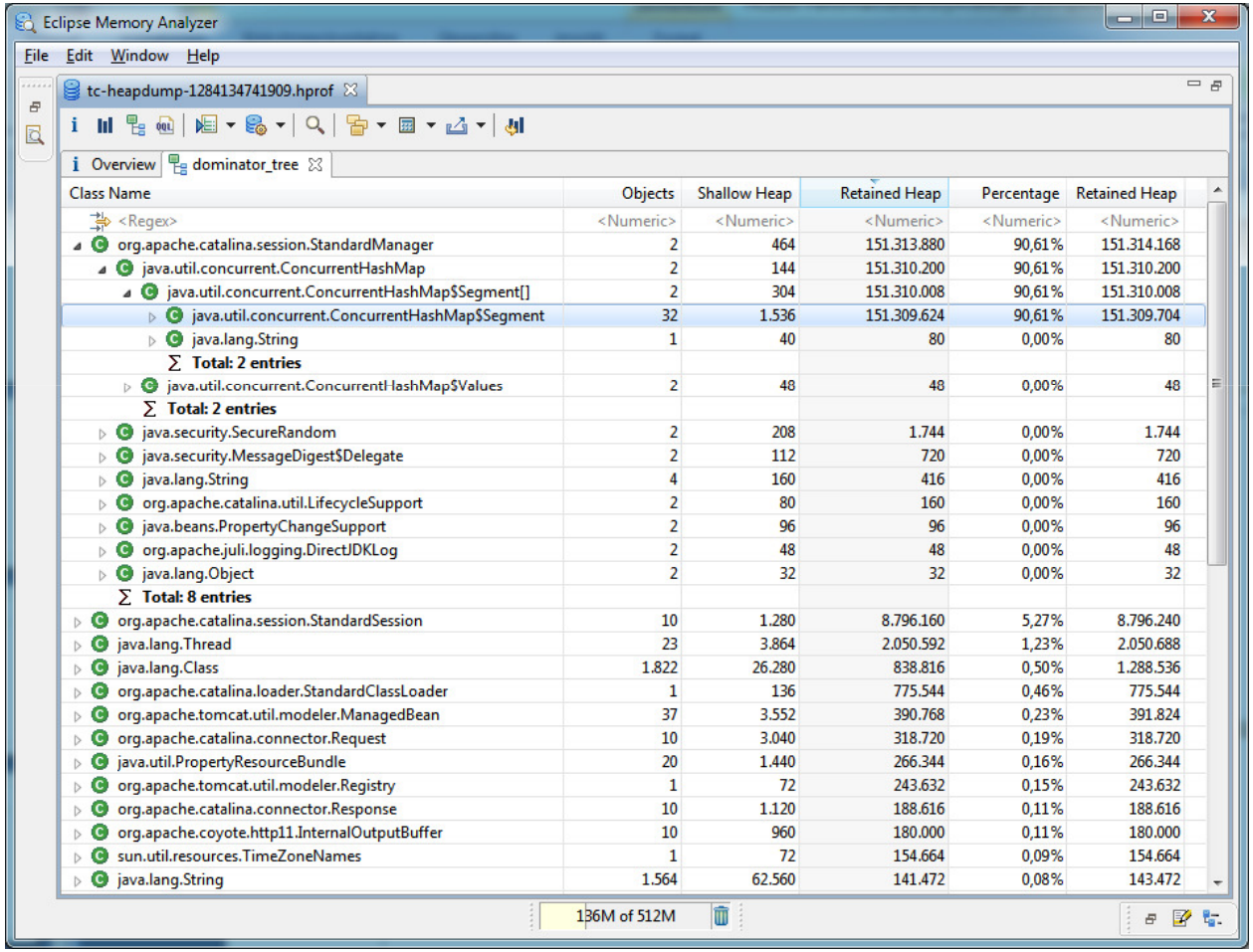
MAT – Overview



MAT

- Leak Suspects Report
  - Analysiert Heap Dump nach möglichen Memory Leaks
- Top Components
  - Alle Komponenten die > 1% des Heaps belegen
- Component Report
  - Identifiziert z.B. vervielfachte Strings, leere Collections usw.
- Histogramm / Dominator Tree
  - Speicherverbrauch herunter gebrochen auf einzelne Instanzen

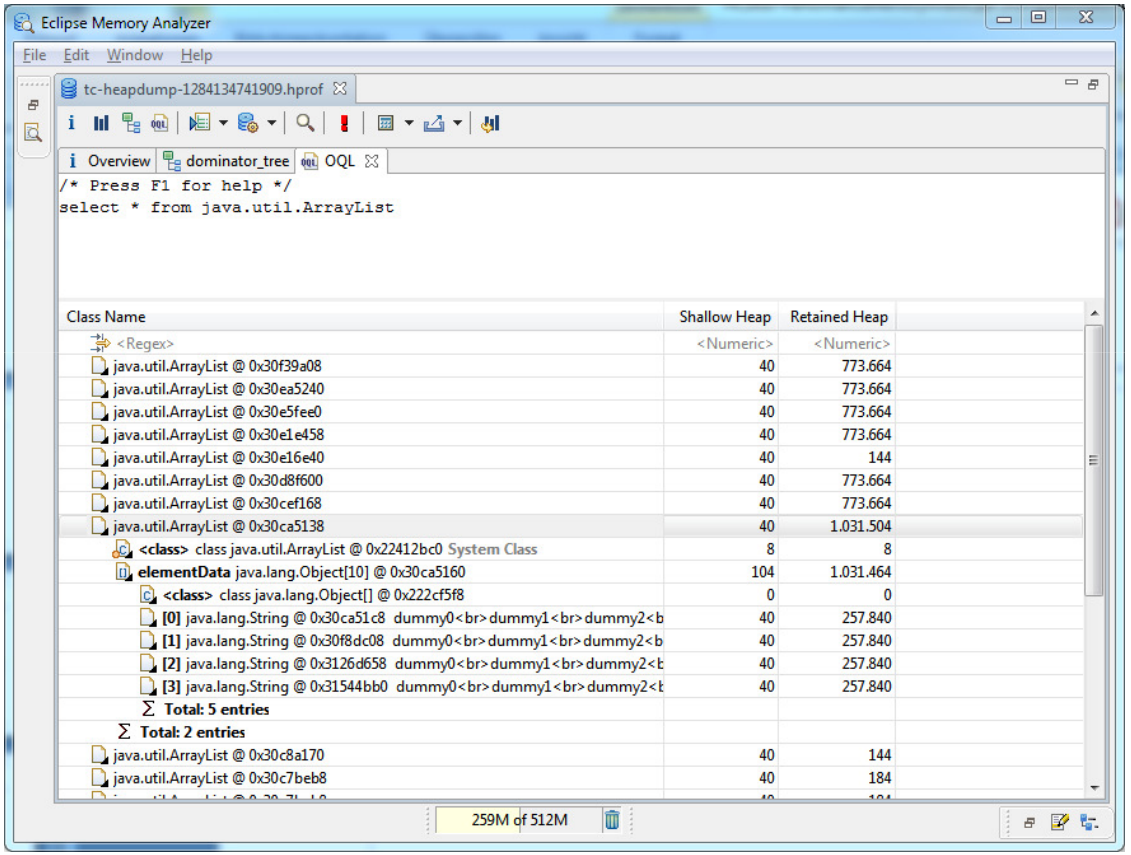
MAT – Domintor Tree



Class Name	Objects	Shallow Heap	Retained Heap	Percentage	Retained Heap
<Regex>	<Numeric>	<Numeric>	<Numeric>	<Numeric>	<Numeric>
org.apache.catalina.session.StandardManager	2	464	151.313.880	90,61%	151.314.168
java.util.concurrent.ConcurrentHashMap	2	144	151.310.200	90,61%	151.310.200
java.util.concurrent.ConcurrentHashMap$Segment[]	2	304	151.310.008	90,61%	151.310.008
java.util.concurrent.ConcurrentHashMap$Segment	32	1.536	151.309.624	90,61%	151.309.704
java.lang.String	1	40	80	0,00%	80
Total: 2 entries					
java.util.concurrent.ConcurrentHashMap$Values	2	48	48	0,00%	48
Total: 2 entries					
java.security.SecureRandom	2	208	1.744	0,00%	1.744
java.security.MessageDigest$Delegate	2	112	720	0,00%	720
java.lang.String	4	160	416	0,00%	416
org.apache.catalina.util.LifecycleSupport	2	80	160	0,00%	160
java.beans.PropertyChangeSupport	2	96	96	0,00%	96
org.apache.juli.logging.DirectJDKLog	2	48	48	0,00%	48
java.lang.Object	2	32	32	0,00%	32
Total: 8 entries					
org.apache.catalina.session.StandardSession	10	1.280	8.796.160	5,27%	8.796.240
java.lang.Thread	23	3.864	2.050.592	1,23%	2.050.688
java.lang.Class	1.822	26.280	838.816	0,50%	1.288.536
org.apache.catalina.loader.StandardClassLoader	1	136	775.544	0,46%	775.544
org.apache.tomcat.util.modeler.ManagedBean	37	3.552	390.768	0,23%	391.824
org.apache.catalina.connector.Request	10	3.040	318.720	0,19%	318.720
java.util.PropertyResourceBundle	20	1.440	266.344	0,16%	266.344
org.apache.tomcat.util.modeler.Registry	1	72	243.632	0,15%	243.632
org.apache.catalina.connector.Response	10	1.120	188.616	0,11%	188.616
org.apache.coyote.http11.InternalOutputBuffer	10	960	180.000	0,11%	180.000
sun.util.resources.TimeZoneNames	1	72	154.664	0,09%	154.664
java.lang.String	1.564	62.560	141.472	0,08%	143.472

MAT – OQL

- Mit Hilfe von OQL ist gezielte Suche nach Speicherverbrauch von bestimmten Objekten möglich
- Syntax sehr einfach… analog SQL



MAT – Viele Funktionen

- MAT bietet sehr viel Funktionalität
  - Suche nach GC Roots
  - Referenz-Zusammenfassungen
  - Ggfs. wichtige Hinweise warum ein Objekt nicht weggeräumt wird!
- Für schnelle Analysen sind die Reports geeignet
- Tiefergehende Analysen benötigen einiges an Einarbeitung

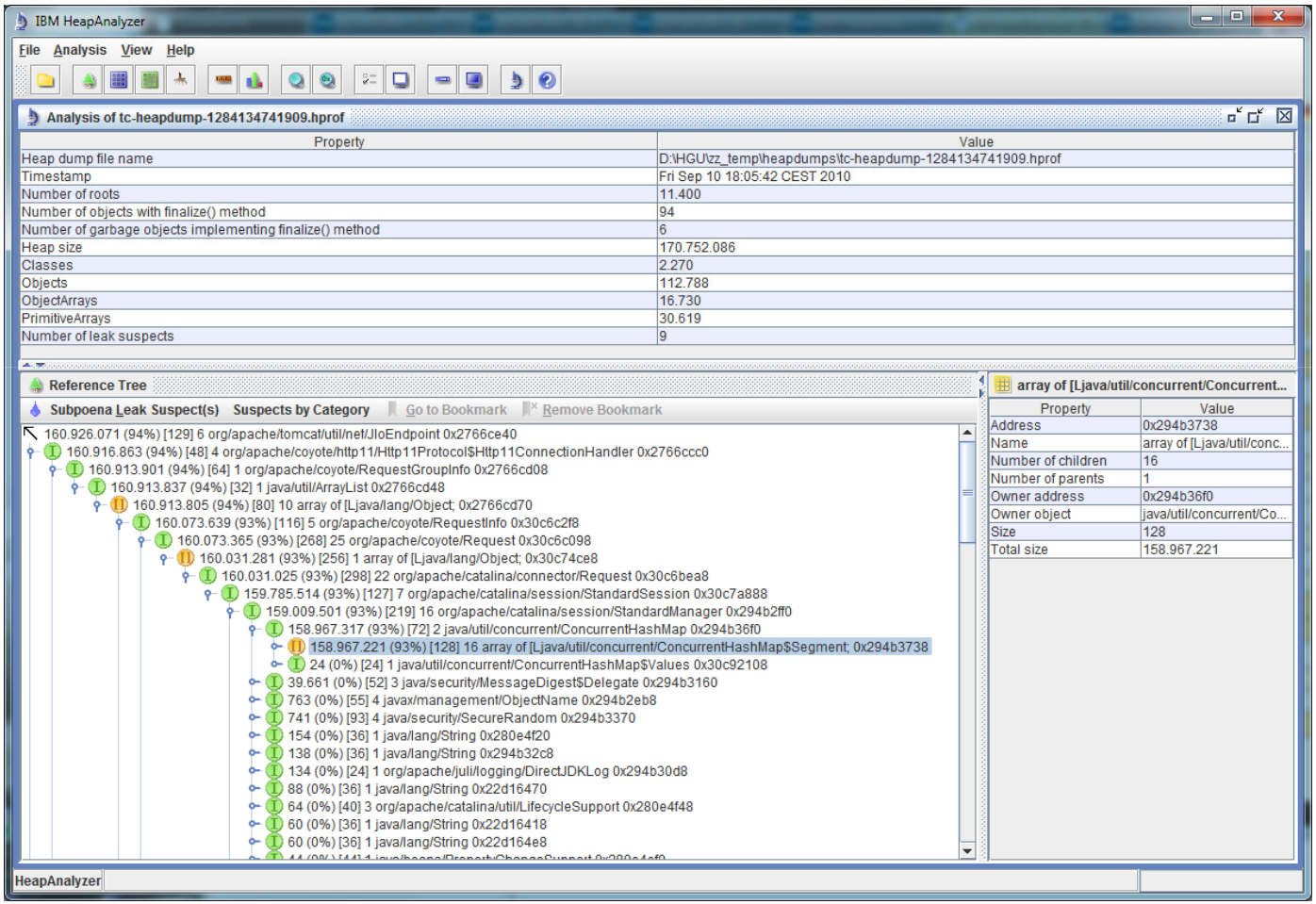
Agenda

- Über das Messen
- Performance
- Messungen bei Performance-Problemen
  - Manuell
  - JVisualVM
- Memory-Probleme
  - JVisualVM
  - MAT
  - IBM Heap Dump Analyzer

IBM Heap Dump Anaylzer

- "IBM Thread and Monitor Dump Analyzer for Java"
  - http://www.alphaworks.ibm.com/tech/jca
- Mein "Favorit" für schnelle Analysen
- Unterstützt (inzwischen) SUN & IBM JDK Heap Dumps im hprof-Format
- Bietet suche nach Leaks
- Übersichtlich, aber nicht so detailverliebt wie MAT = Pragmatisch

IBM HDA


IBM HeapAnalyzer
File Analysis View Help
Analysis of tc-heapdump-1284134741909.hprof
Property
Value
Heap dump file name
D:\HGU\zz_temp\heapdumps\tc-heapdump-1284134741909.hprof
Timestamp
Fri Sep 10 18:05:42 CEST 2010
Number of roots
11.400
Number of objects with finalize() method
94
Number of garbage objects implementing finalize() method
6
Heap size
170.752.086
Classes
2.270
Objects
112.788
ObjectArrays
16.730
PrimitiveArrays
30.619
Number of leak suspects
9
Reference Tree
Subpoena Leak Suspect(s)
Suspects by Category
Go to Bookmark
Remove Bookmark
160.926.071 (94%) [129] 6 org/apache/tomcat/util/net/JIoEndpoint 0x2766ce40
160.916.863 (94%) [48] 4 org/apache/coyote/http11/Http11Protocol$Http11ConnectionHandler 0x2766ccc0
160.913.901 (94%) [64] 1 org/apache/coyote/RequestGroupInfo 0x2766cd08
160.913.837 (94%) [32] 1 java/util/ArrayList 0x2766cd48
160.913.805 (94%) [80] 10 array of [Ljava/lang/Object; 0x2766cd70
160.073.639 (93%) [116] 5 org/apache/coyote/RequestInfo 0x30c6c2f8
160.073.365 (93%) [268] 25 org/apache/coyote/Request 0x30c6c098
160.031.281 (93%) [256] 1 array of [Ljava/lang/Object; 0x30c74ce8
160.031.025 (93%) [298] 22 org/apache/catalina/connector/Request 0x30c6bea8
159.785.514 (93%) [127] 7 org/apache/catalina/session/StandardSession 0x30c7a888
159.009.501 (93%) [219] 16 org/apache/catalina/session/StandardManager 0x294b2ff0
158.967.317 (93%) [72] 2 java/util/concurrent/ConcurrentHashMap 0x294b36f0
158.967.221 (93%) [128] 16 array of [Ljava/util/concurrent/ConcurrentHashMap$Segment; 0x294b3738
24 (0%) [24] 1 java/util/concurrent/ConcurrentHashMap$Values 0x30c92108
39.661 (0%) [52] 3 java/security/MessageDigest$Delegate 0x294b3160
763 (0%) [55] 4 javax/management/ObjectName 0x294b2eb8
741 (0%) [93] 4 java/security/SecureRandom 0x294b3370
154 (0%) [36] 1 java/lang/String 0x280e4f20
138 (0%) [36] 1 java/lang/String 0x294b32c8
134 (0%) [24] 1 org/apache/juli/logging/DirectJDKLog 0x294b30d8
88 (0%) [36] 1 java/lang/String 0x22d16470
64 (0%) [40] 3 org/apache/catalina/util/LifecycleSupport 0x280e4f48
60 (0%) [36] 1 java/lang/String 0x22d16418
60 (0%) [36] 1 java/lang/String 0x22d164e8
array of [Ljava/util/concurrent/Concurrent...
Property
Value
Address
0x294b3738
Name
array of [Ljava/util/conc...
Number of children
16
Number of parents
1
Owner address
0x294b36f0
Owner object
java/util/concurrent/Co...
Size
128
Total size
158.967.221
HeapAnalyzer

Weitere Tools (teilweise veraltet)

- IBM Diagnostic Tool for Java Garbage Collector
  - Für IBM JDK
  - http://www.alphaworks.ibm.com/tech/gcdiag
- jhat – Heap Dump Browser, Teil der SUN JDK
- Jconsole – Teil der SUN JDK
  - Funktionalität durch JVisualVM abgedeckt
- HPROF
  - http://java.sun.com/developer/technicalArticles/Programming/HPROF.html
  - Sehr rudimentäres Profiling-Tool, Analyse mit *perfanal*
  - Option *-Xrunhprof*

12.–15.09.2010
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Vielen Dank!

Dr. Halil-Cem Gürsoy

adesso AG

halil-cem.guersoy [at] adesso.de & @hgutwit

adesso AG

adesso

- Sponsor Herbstcampus 2010
- Mitarbeiter
  - Über 750 Mitarbeiter in der adesso Group
- Umsatz
  - Umsatzerwartung 2010: > 79 Mio. Euro
- Auszeichnungen
  - Platz 18 der deutschen Top-25 Beratungs- und Systemintegrationsunternehmen (Lünendonk-Liste 2010)
  - Top Job (2008)
  - Deutschlands beste Arbeitgeber (2005 und 2010)
- Standorte in Dortmund, Köln, Aachen, Stuttgart, Hamburg, Berlin, Frankfurt, München sowie in der Schweiz und Österreich