THE BUSY JAVA DEVELOPER'S GUIDE TO WEBSPHERE DEBUGGING & TROUBLESHOOTING

ROHIT KELAPURE
IBM ADVISORY SOFTWARE ENGINEER
HTTP://WWW.LINKEDIN.COM/IN/ROHITKELAPURE
HTTP://TWITTER.COM/RKELA
HTTP://WASDYNACACHE.BLOGSPOT.COM/
Agenda

- WebSphere Application Server component overview
- IBM Support Assistant
- JVM Troubleshooting Tools
- WebSphere Problem Determination Tools
- Scenario based problem resolution
- How customers get in trouble
- BadApp Demo
- Q&A
WAS Component Overview
IBM Support Assistant
Workbench to help with Problem Determination

Welcome to IBM Support Assistant

Find Information
Easily find the information you need including product specific information and search capabilities.

Analyze Problem
Diagnose and analyze problems through serviceability tools, collection of diagnostic artifacts, and guidance through problem determination.

Collect and Send Data
Collect problem determination files using automated data collection. Use these files for self-help problem determination, or use the Service Request function to send the files with a service request to IBM.

Latest News
- Update! Garbage Collection and Memory Visualizer 2.4 for ISA 4.1
- New! Guided Troubleshooter search extension
- New! HeapAnalyzer memory leak detector
- Updated! IBM Port Scanning Tool
- More News...
ISA V 4.1.1 Data Collection

Remote Collection using ISA lite

IBM Support Assistant Lite Usage

Step 1: Use the controls on the right to specify the collector type, the target directory for the generated collector and the collector filename for IBM Support Assistant Lite collector, then click the Export button.

Step 2: Transfer the exported IBM Support Assistant Lite collector to a remote system using ftp or any other available file transfer protocol.

Step 3: Unzip the exported IBM Support Assistant Lite collector, setup JAVA_HOME and run runISALite.bat (Windows) or runISALite.sh (UNIX/LINUX) script from the tool's ISALite directory on the remote system. Provide responses to any collection prompts.

More information about the configuration of JAVA_HOME

Step 4: IBM Support Assistant Lite Collector will collect the appropriate information and files into a collection archive file. You can then transfer the file back to IBM Support Workbench and import it into a case for analysis with various tools or send it to the third party, such as IBM Support, for analysis.
# ISA Tools

Single Repository, up-to-date, scripts with IGAA

## Tools Catalog

<table>
<thead>
<tr>
<th>Tool Name</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Tech Preview] Database Connection Pool Analyzer for IBM WebSphere Application Server</td>
<td>1.5.0.02</td>
</tr>
<tr>
<td>[Tech Preview] HeapAnalyzer</td>
<td>3.9.8.00</td>
</tr>
<tr>
<td>[Tech Preview] IBM Pattern Modeling and Analysis Tool for Java Garbage Collector (PMAT)</td>
<td>3.9.6.01</td>
</tr>
<tr>
<td>[Tech Preview] IBM Port Scanning Tool</td>
<td>1.1.0.00</td>
</tr>
<tr>
<td>[Tech Preview] IBM Thread and Monitor Dump Analyzer for Java (TMDA)</td>
<td>3.9.6.01</td>
</tr>
<tr>
<td>[Tech Preview] IBM Trace and Request Analyzer for WebSphere Application Server</td>
<td>2.1.0.03</td>
</tr>
<tr>
<td>[Tech Preview] IBM Web Server Plug-in Analyzer for WebSphere Application Server (WSPA)</td>
<td>3.5.0.02</td>
</tr>
<tr>
<td>[Tech Preview] Memory Dump Diagnostic for Java (MDD4J) version 3.0</td>
<td>3.0.1.beta-20091201202124</td>
</tr>
<tr>
<td>[Tech Preview] ThreadAnalyzer (Deprecated)</td>
<td>6.0.3.02</td>
</tr>
<tr>
<td>IBM Assist On-site</td>
<td>1.0.0.04</td>
</tr>
<tr>
<td><strong>IBM Monitoring and Diagnostic Tools for Java™ - Dump Analyzer</strong></td>
<td><strong>2.2.2.20090926232659</strong></td>
</tr>
<tr>
<td><strong>IBM Monitoring and Diagnostic Tools for Java™ - Garbage Collection and Memory Visualizer</strong></td>
<td><strong>2.4.0.20100127</strong></td>
</tr>
<tr>
<td><strong>IBM Monitoring and Diagnostic Tools for Java™ - Health Center v1.2 Beta</strong></td>
<td><strong>1.2.0.20100315</strong></td>
</tr>
<tr>
<td><strong>IBM Monitoring and Diagnostic Tools for Java™ - Memory Analyzer (Tech Preview)</strong></td>
<td><strong>0.5.2.200910011055</strong></td>
</tr>
<tr>
<td>Log Analyzer</td>
<td>4.5.0.200909240916</td>
</tr>
<tr>
<td>Memory Dump Diagnostic for Java (MDD4J)</td>
<td>2.0.0.20081219132011</td>
</tr>
<tr>
<td>Symptom Editor</td>
<td>4.5.0.200909231042</td>
</tr>
<tr>
<td>Visual Configuration Explorer (Tech Preview)</td>
<td>1.0.16.20090920832</td>
</tr>
</tbody>
</table>
### Tools for information generated by JVM

<table>
<thead>
<tr>
<th>Problem</th>
<th>Artifact</th>
<th>Tools</th>
</tr>
</thead>
</table>
| Memory leaks                                 | Verbose Garbage collection log (native_stdout.log) | • Garbage Collection & Memory Visualizer  
• Pattern Modeling & Analysis tool            |
| Out of Memory errors                         |                                               |                                                                      |
| Application Unresponsive                      |                                               |                                                                      |
| High CPU, Crash, Hang, Performance bottleneck, Unexpected termination | Javadump, Javacore (javacore*.txt) | • Thread Monitor & Dump Analyzer (TMDA)                             |
| Lock Contention                              | Threads (Connection to running JVM)            | • Monitoring & Diagnostic Tools for Java – Health Center              |
| Low CPU at high load                         |                                               |                                                                      |
| Memory Leak                                  | Heapdump (*.phd, *.txt, *.hprof)               | • Memory Analyzer Tool  
• HeapAnalyzer  
• MDD4J                                           |
| Out of Memory errors                         |                                               |                                                                      |
| Native Memory Leak Anomalies                 | System or core dump (core.dmp, user.dmp), Files must be processed with jextract tool | • Dump Analyzer  
• Modules for Dump Analyzer                      |
| Unexpected Crash                             |                                               |                                                                      |
WAS Runtime Serviceability aids

- Troubleshooting panels in the admin console
- Performance Monitoring Infrastructure metrics
  - Tivoli Performance Viewer Monitoring
- Diagnostic Provider Mbeans
  - Dump Configuration, State and run self-test
- Application Response Measurement/Request Metrics
  - Follow transaction end-to-end and find bottlenecks
- Trace logs & First Failure Data Capture
- Runtime Performance Advisors
  - Memory leak detection, session size, ...
- Specialized tracing and Runtime checks
  - Session crossover, Connection leak, WsByteBuffer leak detection ...
- zOS runaway CPU thread protection
- Diagnostic Toolkit & Framework for Java
  - API for accessing information in a system dump in an OS agnostic way
## WebSphere Troubleshooting

<table>
<thead>
<tr>
<th>Problem</th>
<th>Tools, Runtime capabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread correlation, Trace log analysis</td>
<td>WAS Trace Analyzer</td>
</tr>
<tr>
<td>Delays in system response</td>
<td>WAS &amp; Plugin Trace &amp; Request Analyzer</td>
</tr>
<tr>
<td>Connection Leaks</td>
<td>• Database Connection Pool Analyzer</td>
</tr>
<tr>
<td></td>
<td>• Connection Leak Trace facility</td>
</tr>
<tr>
<td></td>
<td>• Enabled using trace string WAS.j2c=finest</td>
</tr>
<tr>
<td>Server hang</td>
<td>Thread Monitor - Hung Thread Detection Policy</td>
</tr>
<tr>
<td></td>
<td>• Every 3 minutes detects threads active for &gt; 10 min</td>
</tr>
<tr>
<td>Classloader issues</td>
<td>Classloader viewer in the WAS Admin console</td>
</tr>
<tr>
<td>Corrupt Install</td>
<td>Install Verification Utility (installver, ivt.bat/sh)</td>
</tr>
<tr>
<td>Runtime network status</td>
<td>Port Scanning tool</td>
</tr>
<tr>
<td>Comparing &amp; Validating Configuration files</td>
<td>• Visual Configuration Explorer</td>
</tr>
<tr>
<td></td>
<td>• WebServices validation tool for WSDL &amp; SOAP</td>
</tr>
<tr>
<td>HTTP Plugin &amp; Proxy (No response, Incorrect load balancing)</td>
<td>• WebServer Plugin Analyzer</td>
</tr>
<tr>
<td></td>
<td>• Dump ODC tree script</td>
</tr>
<tr>
<td>Issues with Security</td>
<td>LDAP Server test connection, SSL Certificate &amp; Key Management, Security Configuration Report</td>
</tr>
</tbody>
</table>
One more tool and I am going to scream
WebSphere Monitoring tools

- Sun origin JVM Tools - visualVM & jConsole
- SIBus – Explorer & Destination Handler
- Dynacache - Cache Monitor & Extended Cache Monitor
- HTTPSession - Session tracker debug Servlet*
- WebServices Request/Response tracking
  - TCPMonitor & Fiddler
- dumpNameSpace tool
- osgiConsole.sh/bat
- Coregroup Visualization and analysis tool*
- Sun JVM ONLY Tools- jps, jstatd, jstack, jmap, jinfo, jhat
- LDAP tools- ldapsearch, apache LDAP browser
- JVM Diagnostics Collector
Most common JVM Problem Scenarios

**Functional Problems**
- Unexpected Exceptions, Compatibility

**OOM Errors**
- Java Heap, Native Heap, Classloaders

**Hangs**
- Synchronized resources, GC Pause times

**Crash**
- JVM errors, JIT errors, JNI errors

**High CPU**
- Spin loops
Have you ever pondered …

- Why does my app. run slow every time I do <…>?
- Why does my app. have erratic response times?
- Why am I getting Out of Memory Errors?
- What is my applications memory footprint?
- Which parts of my app. are CPU intensive?
- How did my JVM vanish without a trace?
- Why is my application unresponsive?
- $%@ is wrong with my application?
- What monitoring do I put in place for my app.?
Why does my app run slow when I do xxx?

- Understand impact of activity on WAS components
  - Look at the thread & method profiles in the IBM Java Health Center

- Set IBM JVM method & dump trace - pinpoint performance problems.
  - Shows entry & exit times of any Java method
    - Method to trace to file for all methods in tests.mytest.package
      - `-Xtrace:maximal=mt,output=trace.out,methods={tests/mytest/*.*}`
  - Allows taking javadump, heapdump, etc when a method is hit
    - Dump javacore when method testInnerMethod in an inner class TestInnerClass of a class TestClass is called
      - `-Xtrace:trigger=method{com/ibm/TestClass$TestInnerClass.testInnerMethod,javadump,,,5}`

- `-Xtrace` and `-Xdump` can be triggered on a huge range of events … with filters and ranges
  - gpf, user, abort, fullgc, slow, allocation, thrstop, throw …
  - Stack traces, tool launching
Why does my application have erratic response times?

- Verbose gc should be enabled by default
  - (<2%) impact on performance

- GCMV & PMAT: Visualize gc output & look @ recos.
  - In use space after GC
    - Positive gradient indicates memory leak
    - Increased load (use for capacity plan) OR memory leak (take HDs for PD.)

- Chose the right GC policy
  - Xgcpolicy:optthroughput - optimized for “batch” type applications, consistent allocation profile
  - Xgcpolicy:optavgpause - tight responsiveness criteria, allocations of large objects
  - Xgcpolicy:gencon - high rates of object “burn”, large # of transitional objects
  - Xgcpolicy:subpools - 12, 16 core SMP systems with allocation contention (AIX only)

- GC overhead > 10% \(\rightarrow\) wrong policy or more tuning

- Enable compressed References enabled for 64 bit JVM?
Why am I getting Out Of Memory Errors?

- **Java 1.4.* -look for heap fragmentation** (rare for Java 5/6)
  - Set -Xp and -Xk for pinned and dosed objects

- **JVM Heap sized incorrectly**
  - IBM JDK Team does NOT recommend Xms == Xmx
  - GC adapts heap size to keep occupancy [40, 70]%

- **Determine heap occupancy of the app. under load**
  - Xmx = 43% larger than max. occupancy of app.
    - For 700MB occupancy, 1000MB Max. heap is reqd. (700 + 43% of 700)

- **Analyze heapdumps & system dumps with dump tools**
  - Lack of Java heap or Native heap

- **Finding which methods allocated large objects**
  - Prints stacktrace for all objects above 1K
    - -Xdump:stack:events=allocation,filter=#1k

- **Enable Java Heap and Native heap monitoring**
  - JMX and PMI metrics output by WAS JVM

- **Classloader exhaustion in pre Java 5 SR3**
What is my applications memory footprint?

- HPROF — profiler shipped with JDK — uses JVMTI
  - Analysis of memory usage -Xrunhprof:heap=all
- Performance Inspector tools - JPROF Java Profiling Agent
  - Capture state of the Java Heap later processed by HDUMP
- Use MAT to investigate heapdumps & system dumps
  - Find large clumps, Inspect those objects, What retains them?
    - Why is this object not being garbage collected
      - List Objects > incoming refs, Path to GC roots, Immediate dominators
    - Limit analysis to a single application in a JEE environment
      - Dominator tree grouped by Class Loader
    - Set of objects that can be reclaimed if we could delete X
      - Retained Size Graphs
    - Traditional memory hogs like Session, Dynacache
      - Use Object Query Language (OQL)
Which parts of my application are CPU intensive?

- **HPROF - CPU spends most of its time**
  - `-Xrunhprof:cpu=samples, -Xrunhprof:cpu=time`

- **JPROF - method level execution times, who calls whom, etc.**
  - Generate WAS startup script & set the JVM argument
  - "-agentlib:jprof=rtarcf,callflow,logpath=./jprof" "-Xjit:disableInlining"
  - Output visualized using VPA

- **Poor mans profiler**
  - - Periodic javacores
  - - Thread analysis - TMDA

- **IBM – Health Center**
  - Low overhead
  - Gets data from JIT
How did my JVM vanish without trace?

- **JVM Process Crash Usual Suspects**
  - Bad JNI calls, Segmentation violations, Call Stack Overflow
  - Native memory leaks - Object allocation fails with sufficient space in the JVM heap
  - Unexpected OS exceptions (out of disk space, file handles), JIT failures

- **Monitor the OS process size**

- **Runtime check of JVM memory allocations -Xcheck:memory**

- **Native memory usage - Create a core or svc dump on an OOM**
  - -Xdump:system:events=systhrow,filter=java/lang/OutOfMemoryError

- **JNI code static analysis -Xcheck:jni (errors, warnings, advice)**

- **GCMV provides scripts and graphing for native memory**
  - Windows “perfmon“, Linux “ps” & AIX “svmon”

- **IBM Dump Analyzer & TMDA to find the last stack of native code executing on the thread during the crash**
Why is my application unresponsive?

- **Usual Suspects (25-50% overhead of an app.)**
  - Thread Contention & sync. issues Deadlocks and Spin loops

- **Take three javadumps spaced 1 minute apart**
  - Native stacks in javadumps in Java 6, SR7

- **Monitor hold time accounting & Contention statistics**
  - Java Lock Monitor, IBM Health Center (always on profiling)

- **Monitor hung thread messages**
  - WSVR0605W (Thread hung), WSVR0606W (Thread completed), WSVR0607W (threshold adjusted)
  - Cause javacore on thread hang - Set “com.ibm.websphere.threadmonitor.dump” custom property to “true”
  - Spawning new threads is an anti-pattern since these threads are outside the hang detection monitoring

- **Is traffic flowing into the application server from the Plugin or the Proxy?**
  - Increasing OS level TCP/IP connection backlog at the webserver/proxy

- **IBM Health Center starts up on port 1972 for WAS and needs a generic JVM argument “-agentlib:healthcenter”**
  - Overhead < 3%
What is wrong with my application?

General problem determination steps

- Finding the Bottleneck
  - CPU, Memory, I/O, Network
  - Systems Resource Usage
    - `top`, `perfmon`, `svmon`, `tasklist`, `ps`, <add your favorite utility here …>

- JVM Heap Sizing
  - Setting Correct GC Policy, Heap occupancy
  - JIT Compilation & Performance

- Application Profiling
  - Inbuilt `hprof`, Health Center
  - Free - Performance Inspector tools (JPROF, TPROF, JLM, Jinsight …)
  - Commercial- RAD, Jprofiler, Yourkit …

- OS specific tuning
  - Setting TCP `TIME_WAIT`
What do I monitor?

### CPU
- Total % CPU
- User %CPU
- Wait %CPU
- System %CPU
- Avg. CPU RunQueue
- Context Switching
- CPU per processor
- CPU usage per process

### Memory
- Total Memory
- Real Memory Free
- Total Swap Space
- Swap Space Used
- Page In
- Page Out
- Pages/sec
- Memory usage per process
- Process % used
- FScache % used
- Scan Rate
- Page faults

### Disk
- Disk Service Time Per Disk
- Avg. Disk Queue Length
- Disk transfer per second
- %Busy
- Disk reads
- Disk writes

### OS
- JDBC Connection Pool
- JVM Runtime
- ServletSession Manager
- SystemData
- ThreadPool
- Web Applications

### Metrics
- Average response/service time
- Number of requests (transactions)
- Number of live HTTP sessions
- Web server thread pools, the WAS Web Container and Enterprise JavaBeans (EJB) thread pools, Database connection pool size
- Java virtual memory (JVM)
- PMI
- CPU, I/O, System paging
11 Malpractices

- No arch. plan
- No migration plan
- No change records
- No Capacity plan
- No Production traffic profile
- Changes put directly in Prod.
- No load & Stress testing
- Communication breakdown
- No education
- Application Error
- Test environment != Production
Demo, Q&A
I would like to thank …

Please google & read their blogs & articles

- **Daniel Julin** (IBM Lab)
- **John Pape** (SWAT)
- **Kevin Grigorenko** (SWAT)
- **Art Jolin** (ISSW)
- **Chris Bailey** (JDK)
- **Katie Barett** (Serviceability team)
- **Russell Wright** (ISA)
- **Thomas Alcott** (Tech. Sales)