Java Monitoring and Diagnostic Tooling

Iris Baron
IBM Java JIT on System Z
ibaron@ca.ibm.com

Session ID: 16182
Java Road Map

Language Updates

Java 5.0
- New Language features:
  - Autoboxing
  - Enumerated types
  - Generics
  - Metadata

Java 6.0
- Performance Improvements
  - Client WebServices Support

Java 6.0
- Support for dynamic languages
- Improve ease of use for SWING
- New IO APIs (NIO2)
- Java persistence API
- JMX 2.x and WS connection for JMX agents
- Language Changes

Java 7.0
- Performance Improvements
- Client WebServices Support
- Support for dynamic languages
- Improve ease of use for SWING
- New IO APIs (NIO2)
- Java persistence API
- JMX 2.x and WS connection for JMX agents
- Language Changes

Java 8.0**
- Language improvements
- Closures for simplified fork/join

IBM Java Runtimes

IBM Java 5.0 (J9 R23)
- Improved performance
  - Generational Garbage Collector
  - Shared classes support
  - New J9 Virtual Machine
  - New Testarossa JIT technology
  - First Failure Data Capture
  - Full Speed Debug
  - Hot Code Replace
  - Common runtime technology
    - ME, SE, EE

IBM Java 5.0 (J9 R23)
- Improved performance
  - Generational Garbage Collector
  - Shared classes support
  - New J9 Virtual Machine
  - New Testarossa JIT technology
  - First Failure Data Capture
  - Full Speed Debug
  - Hot Code Replace
  - Common runtime technology
    - ME, SE, EE

IBM Java 6.0 (J9 R24)
- Improvements in
  - Performance
  - Serviceability tooling
  - Class Sharing
  - XML parser improvements
  - z10™ Exploitation
  - DFP exploitation for BigDecimal
  - Large Pages
  - New ISA features

IBM Java 7.0
- Improvements in
  - Performance
  - GC Technology
  - z196™ Exploitation
  - OOO Pipeline
  - 70+ New Instructions
  - JZOS/Security Enhancements

IBM Java 7.0
- Improvements in
  - Performance
  - GC Technology
  - z196™ Exploitation
  - OOO Pipeline
  - 70+ New Instructions
  - JZOS/Security Enhancements

IBM Java 7.0SR3
- Improvements in
  - Performance
  - zEC12™ Exploitation
  - Transactional Execution
  - Flash 1Meg pageable LPs
  - 2G large pages
  - Data Access Accelerator

IBM Java7R1
- Improvements in
  - Performance
  - RAS
  - Monitoring
  - zEC12™ Exploitation
  - zEDC for zip acceleration
  - SMC-R integration
  - Transactional Execution
  - Runtime instrumentation
  - Hints/traps

IBM Java7R1
- Improvements in
  - Performance
  - RAS
  - Monitoring
  - zEC12™ Exploitation
  - zEDC for zip acceleration
  - SMC-R integration
  - Transactional Execution
  - Runtime instrumentation
  - Hints/traps

IBM Java7.0SR3
- Improvements in
  - Performance
  - zEC12™ Exploitation
  - Transactional Execution
  - Flash 1Meg pageable LPs
  - 2G large pages
  - Data Access Accelerator

IBM Java7.0SR3
- Improvements in
  - Performance
  - zEC12™ Exploitation
  - Transactional Execution
  - Flash 1Meg pageable LPs
  - 2G large pages
  - Data Access Accelerator

**Timelines and deliveries are subject to change.
Linux on System z and Java7SR3 on zEC12

~12x aggregate hardware and software improvement comparing Java5SR4 on z9 to Java7SR3 on zEC12

- LP=Large Pages for Java heap
- CR=Java compressed references
- Java7SR3 using -Xaggressive + 1Meg large pages

(Controlled measurement environment, results may vary)
WAS on zLinux

Aggregate HW, SDK and WAS Improvement:
WAS 6.1 (Java 5) on z9 to WAS 8.5 (Java 7) on zEC12

4x aggregate hardware and software improvement comparing WAS 6.1 Java5 on z9 to WAS 8.5 Java7 on zEC12

(Controlled measurement environment, results may vary)
IBM Operational Decision Manager

IBM Operational Decision Management zEC12 16-way

Throughput (Normalized to IBM Java 7 SR4)

IBM Java 7

IBM Java 7R1

(19% improvement to ODM with IBM Java 7R1 compared to IBM Java 7)

(Controlled measurement environment, results may vary)
Java Monitoring and Diagnostic Tooling

Agenda

• IBM Monitoring and Diagnostic Tools for Java
  – Why use the tools?
  – Where to get the tools?

• IBM Recommended Java Troubleshooting Tools
  – Health Center
  – Garbage Collector and Memory Visualizer
  – Memory Analyzer

• Summary
Java Monitoring and Diagnostic Tooling

Why use the IBM Tools?

- Tools simplify troubleshooting problems:
- IBM provides a free unified suite of tools to understand different aspects of Java applications
- Fully IBM supported
- Tools provide visualizations, analysis and recommendations

Fixing problems ...
... is much easier with the right tool for the job!

Possible problems:
- Application coding errors
- Environment variables
- Performance tuning
- Configuration problems
Java Monitoring and Diagnostic Tooling
Where to get the IBM Tools?

- IBM Support Assistant
  A free application available at:
  http://www.ibm.com/software/support/isa

- Eclipse Market Place
  Tools available to install directly into Eclipse
Java Monitoring and Diagnostic Tooling
What is IBM Support Assistant?

IBM Support Assistant (ISA) is a free application that:

- Provides the “toolbox” in which analysis and diagnostic tools reside
  Over one hundred “add-ons” available for various IBM products

- Provides Serviceability Tools across product families
  Simplifies software support

- Provides Search feature to query IBM and non-IBM knowledge banks

- Not a monitoring tool
Java Monitoring and Diagnostic Tooling

ISA Workbench – Diagnostic Tools

Cross-product Environment Troubleshooting
- Log Analyzer
- Visual Configuration Explorer
- Processor Time Analysis Tool for Linux

Java Troubleshooting
- Memory Analyzer
- Performance Analysis Tool
- Heap Analyzer
- Health Center
- Multicore Software Development Kit for Java
- Garbage Collection and Memory Visualizer
- IBM Thread and Monitor Dump Analyzer
- IBM Pattern Modeling and Analysis Tool

WebSphere Troubleshooting
- Web Server Plug-in Analyzer for WAS
- Database Connection Pool Analyzer for WAS
- WAS Analysis Module for Dump Analyzer
- IBM Trace and Request Analyzer for WAS
- IBM Web Services Validation Tool

Lotus Troubleshooting
- Lotus Notes Diagnostic
- Domino Configuration Tuner

IM / FileNet Troubleshooting
- FileNet OSAR Cable Tool

Remote assistance
Assist on-site
Java Monitoring and Diagnostic Tooling

What is IBM Support Assistant?

**ISA Workbench 4.1**

- Eclipse-based client
- Workbench is installed on each desktop (single user)
- Collect and organize diagnostic data (logs, traces, etc.)
- Find and use Problem Determination tools
- Search and browse support-related information about IBM products
Java Monitoring and Diagnostic Tooling
What is IBM Support Assistant?

ISA 5.0 Team Server

- Server-based model
- Install once - shared by many team members via browser
- Web 2.0 browser interface
- Remote execution of PD tools
- Off-load analysis processing
- Collaboration on PD
- Case Management
- Tool Management
- Single-user option available
Java Monitoring and Diagnostic Tooling

Agenda

• IBM Monitoring and Diagnostic Tools for Java
  – Why use the tools?
  – Where to get the tools?

• IBM Recommended Java Troubleshooting Tools
  – Health Center
  – Garbage Collector and Memory Visualizer
  – Memory Analyzer

• Summary
Java Monitoring and Diagnostic Tooling
Health Center

Motivating questions:

• What is my JVM doing? Is everything ok?
• Why is my application running slowly?
• Why is it not scaling?
• Am I using the right options?
Java Monitoring and Diagnostic Tooling
Health Center

- Live monitoring tool with very low overhead (< 1%)
- Suitable for all Java applications running on IBM’s JVM
- Provides insight into your application behaviour with visualization
- Diagnoses potential problems with recommendations
- Powerful API allowing embedding of Health Center into other applications
Java Monitoring and Diagnostic Tooling
Health Center

Health Center provides visualization and monitoring in the following application areas:

• Method profiling
• Lock analysis
• Garbage Collection
• Threading
• Memory Usage
• System environment
• Java class loading
• Object Allocations
• File I/O
Java Monitoring and Diagnostic Tooling
Health Center - Installation

• The tool is provided in two parts:
  – An Agent that collects data from a running application
  – An Eclipse-based client that connects to the agent

• The Agent ships with the following vm’s:
  – Java 5sr9 and upwards
  – Java 6sr3 and upwards

• The latest version of the agent is always available from within the Health Center Client
  – Recommended to always update to the latest version of the agent
  – Agent package unzips over the jre directory of the JVM you are using
Java Monitoring and Diagnostic Tooling
Health Center - Enable for Monitoring

- Full instructions are provided within the help shipped with the Health Center Client but in most cases as simple as:

For Java 5 SR10 and later or Java 6 SR5 and later, including Java 7 (can be used in production)

```
java -Xhealthcenter HelloWorld
```

For Java 5 SR9 and earlier, or Java 6 SR4 and earlier (not recommended for use in a production environment)

```
java -agentlib:healthcenter -Xtrace:output=healthcenter.out HelloWorld
```
Java Monitoring and Diagnostic Tooling
Health Center – Advanced Options

• Headless mode for data collection without connecting the GUI
  – Useful for scenarios where firewall blocks connection
  – Configurable to limit disk space used
  – Timed collections
  – Interval based collections
  – Started with
    -Xhealthcenter:level=headless

• Late attach enabled

• Automated javacore creation
Java Monitoring and Diagnostic Tooling
Health Center – API

• The 2.2 release of Health Center contains a powerful API that allows Java developers to embed Health Center in their applications and harness its monitoring power to troubleshoot problems

• Example:

```java
// Create the connection object:
ConnectionProperties conn1 = new ConnectionProperties("localhost", 1973);

// Connect to the Health Center agent, using the previous connection settings:
HealthCenter hcObject = HealthCenterFactory.connect(conn1, true);

// Get garbage collection data and print:
GCData gcData = hcObject.getGCData();
System.out.println("GC Mode is " + gcData.getGCMode().toString());
```
Environment reporting
• Detects invalid Java options
• Detects options which may hurt performance or serviceability
• Useful for remote diagnosis of configuration-related problems

Garbage Collection visualization
• Visualizes heap usage and gc pause times over time
• Identifies memory leaks
• Suggests command-line and tuning parameters
• Same recommendation logic as GCMV
Java Monitoring and Diagnostic Tooling
Health Center

Class loading visualization
• Shows all loaded classes
• Shows load time
• Identifies shared classes
• Live class histogram information

CPU usage
• Visualizes overall system CPU use as well as application process use
Java Monitoring and Diagnostic Tooling

Health Center

I/O
- Monitor application file open/close events as they occur
- Lists currently open files

Native Memory
- Detect native memory leaks in application
- Determine if external forces are using more memory
- Memory counters showing which parts of the JVM are using the most native memory
Method Profiling
- Always-on profiling offers insight into application activity
- Identifies the hottest methods in an application
- Full call stacks to identify where methods are being called from and what methods they call
- No byte code instrumentation, no recompiling

Java Lock analysis
- Always-on lock monitoring
- Quickly allows the usage of all locks to be profiled
- Helps to identify points of contention in the application that are preventing scaling
Java Monitoring and Diagnostic Tooling
Health Center

Threads view
• List of current threads and states
• Deadlock detection and analysis
• Number of threads over time
• See contended monitors

Live control of application
• Trigger dumps
• Enable verbosegc collection
Java Monitoring and Diagnostic Tooling

Agenda

• IBM Monitoring and Diagnostic Tools for Java
  – Why use the tools?
  – Where to get the tools?

• IBM Recommended Java Troubleshooting Tools
  – Health Center
  – Garbage Collector and Memory Visualizer (GCMV)
  – Memory Analyzer

• Summary
Java Monitoring and Diagnostic Tooling
Garbage Collector and Memory Visualizer (GCMV)

Motivating questions:
- How is the GC behaving? Can I do better?
- How much time is GC taking?
- How much free memory does my JVM have?

Overview
- Analyze Java verbose GC logs, providing insight into application behaviour
- Uses `ps -p $PID -o pid,vsz,rss` output to plot native footprint
- Visualize a wide range of GC data and Java heap statistics over time
- Provides the ability to detect memory leaks and optimized GC
- Recommendations use heuristics to guide you towards GC performance tuning
Java Monitoring and Diagnostic Tooling
Garbage Collector and Memory Visualizer (GCMV)

- Views of GCMV
Java Monitoring and Diagnostic Tooling
Garbage Collector and Memory Visualizer (GCMV)

Graphical Display of Data
• Allows graphing of all available data: pause times, heap size etc
• Allows zoom, cropping and change of axes value and units
• Allows comparison of multiple files

Tuning recommendation
• The garbage collector seems to be compacting excessively. On average 45% of each pause was spent compacting the heap. Compaction occurred on 40% of collections. Possible causes of excessive compaction include the heap size being too small or the application allocating objects that are larger than any contiguous block of free space on the heap.
• The garbage collector is performing system (forced) GCs. 5 out of 145 collections (3.44%) were triggered by System.gc() calls. The use of System.gc() is generally not recommended since they can cause long pauses and do not allow the garbage collection algorithms to optimise themselves. Consider inspecting your code for occurrences of System.gc().
• The mean occupancy in the nursery is 7%. This is low, so the gencon policy is probably an optimal policy for this workload.
• The mean occupancy in the tenured area is 14%. This is low, so you have some room to shrink the heap if required.

Summary

<table>
<thead>
<tr>
<th>Allocation failure count</th>
<th>140</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concurrent collection count</td>
<td>0</td>
</tr>
<tr>
<td>Forced collection count</td>
<td>5</td>
</tr>
<tr>
<td>GC Mode</td>
<td>gencon</td>
</tr>
<tr>
<td>Global collections - Mean garbage collection pause (ms)</td>
<td>165</td>
</tr>
<tr>
<td>Global collections - Mean interval between collections (minutes)</td>
<td>0.13</td>
</tr>
<tr>
<td>Global collections - Number of collections</td>
<td>5</td>
</tr>
<tr>
<td>Global collections - Total amount tenured (MB)</td>
<td>93.1</td>
</tr>
<tr>
<td>Largest memory request (bytes)</td>
<td>127784</td>
</tr>
<tr>
<td>Minor collections - Mean garbage collection pause (ms)</td>
<td>46.2</td>
</tr>
<tr>
<td>Minor collections - Mean interval between collections (ms)</td>
<td>71.93</td>
</tr>
<tr>
<td>Minor collections - Number of collections</td>
<td>140</td>
</tr>
<tr>
<td>Minor collections - Total amount tenured (MB)</td>
<td>668</td>
</tr>
<tr>
<td>Minor collections - Total amount uncollected (MB)</td>
<td>36.0</td>
</tr>
<tr>
<td>Proportion of time spent in garbage collection pauses (%)</td>
<td>0.76</td>
</tr>
<tr>
<td>Proportion of time spent unpaused (%)</td>
<td>99.24</td>
</tr>
<tr>
<td>Rate of garbage collection (MB/minutes)</td>
<td>874</td>
</tr>
</tbody>
</table>

Analysis and Recommendations
• Provides tuning recommendations based on data and flags errors.
• Analysis can be limited using cropping.
• Values and units used in analysis can be changed by changing axes values and units
Java Monitoring and Diagnostic Tooling

Agenda

• IBM Monitoring and Diagnostic Tools for Java
  – Why use the tools?
  – Where to get the tools?

• IBM Recommended Java Troubleshooting Tools
  – Health Center
  – Garbage Collector and Memory Visualizer
  – Memory Analyzer

• Summary
Java Monitoring and Diagnostic Tooling
Memory Analyzer

Motivating questions:
- Why did I run out of Java memory?
- What’s in my Java heap? How can I explore it and get new insights?

Overview
- Tool for analyzing heap dumps and identifying memory leaks from JVMs
- Works with IBM system dumps, heapdumps and Sun HPROF binary dumps
- Provides memory leak detection, footprint analysis:
  - Objects by Class, Dominator Tree Analysis, Path to GC Roots, Dominator Tree by Class Loader
  - Shows areas of memory wastage:
    - Collections, duplicate strings, substring/char arrays, constant value primitives
- Displays Stack trace with object references
- Provides SQL like object query language (OQL)
- Provides extension points to write analysis plugins
**Java Monitoring and Diagnostic Tooling**

**Memory Analyzer**

**Overview:**
- Overview of the heapdump including size and total number of objects.
- Provides links to continued analysis

**Path to GC Roots:**
- Provides the reference chain that prevents an object being garbage collected

**Dominator Tree grouped by Class Loader:**
- Lists the biggest objects using a “keep alive tree” Grouping by Class
- Loader limits the analysis to a single application in a JEE environment
Java Monitoring and Diagnostic Tooling

Agenda

• IBM Monitoring and Diagnostic Tools for Java
  – Why use the tools?
  – Where to get the tools?

• IBM Recommended Java Troubleshooting Tools
  – Health Center
  – Garbage Collector and Memory Visualizer
  – Memory Analyzer

• Summary
## Java Monitoring and Diagnostic Tooling

### Problem Scenarios and Tools

<table>
<thead>
<tr>
<th></th>
<th>GCMV</th>
<th>Health Center</th>
<th>Memory Analyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Performance</strong></td>
<td>• Garbage Collection performance only</td>
<td>• Method Profiling</td>
<td>• Garbage analysis</td>
</tr>
<tr>
<td></td>
<td>• Garbage Collection only</td>
<td>• Lock Analysis</td>
<td>• Collection efficiency</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>• Garbage Collection memory monitoring</td>
<td>• Garbage Collection memory monitoring</td>
<td>• Java heap memory analysis</td>
</tr>
<tr>
<td></td>
<td>• Native (process) memory monitoring</td>
<td>• Native (process) memory monitoring</td>
<td></td>
</tr>
<tr>
<td><strong>Runtime</strong></td>
<td>• Process settings</td>
<td>• Class Loading</td>
<td>• Thread execution analysis</td>
</tr>
<tr>
<td></td>
<td>• Class Loading</td>
<td></td>
<td>• Application state reports</td>
</tr>
</tbody>
</table>

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval
Thank You

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval
Where to find more information

- Documentation
- zOS SDK
- System z Linux SDK
- Java Tuning documentation
  - http://www.ibm.com/developerworks/views/java/libraryview.jsp?search_by=java+technology+ibm+style:
- IBM Support Assistant
- IBM Monitoring and Diagnostic Tools for Java™
  - http://pic.dhe.ibm.com/infocenter/isa/v4r1m0/index.jsp
- Health Center API articles
  - Monitor a Java application with the Health Center API parts 1 and 2