Java Diagnosis for the z/OS System Programmer

Ken Irwin
IBM Corporation

August 5, 2010

Java has many diagnostic data sources

- Because problems may need multiple diagnostic data types for resolution, it is up to the Systems personnel to capture and collect the various types of output.

- We'll talk about some features in Java6 that can help you in this area. We'll also cover some diagnostic options available on z.

Diagnostic Data Types

- Dumps
  - Heapdumps : a dump of the java heap contents
    - phd format (Portable Heap Dumps)
    - txt format (Classic)
    - Common across all platforms
    - Triggered for OOM errors or programmatically
  - Javacores: a dump of the java virtual machine
    - Txt format
    - Triggered via exception or programmatically
    - Common across all platforms

Diagnostic Data Types (cont'd)

- Tdumps (Transaction Dumps)
  - Unique to z/OS
  - IPCS compatible format
  - Similar to a console dump of the Address Space
  - Includes typical zOS dump contents
- Snap trc files
  - Technically is a trace file containing a dump of internal jvm trace data
  - Must be formatted to display its contents
  - Common (mostly) content across all platforms

Trace Data Types

- Garbage Collection Trace
  - Common across all platforms
  - Enabled with -verbosegc (or -verbose:gc)
- Method Trace
  - Captures method entry and exit
  - Potential performance impacts
- Class-loader Trace
  - Captures class loader searching and application class loading data

How can I investigate this diagnostic data together ?

- ISA (IBM Support Assistant)
  - See John Hutchinson's Boston SHARE presentation titled
    ”Introduction to using IBM Support Assistant for WebSphere Application Server for z/OS”
- ISA provides the user with multiple tooling options for analyzing diagnostic data captured from the JVM (Java Virtual Machine)

What can I do on the z platform?

- While ISA does provide a great deal of diagnostic and analytic capability, it isn't z-based.
- Data must be transferred to ISA for analysis, most likely your workstation. Diagnostic data may be the javacore, the heapdump, verbosegc traces, etc.
- Consider using -Xdiagnosticscollector to simplify data collection

How can -Xdiagnosticscollector help me?

- The Diagnostics Collector runs just after the Java runtime produces diagnostic files
- It will search for:
  - System dumps
  - Java dumps
  - Heap dumps
  - Java trace dumps
  - Verbose GC logs matching the problem event timestamp

How can -Xdiagnosticscollector help me?

- Optionally, if a system dump (tdump) is found, it can execute **jextract** to post-process the dump and capture extra information required to analyze the system dump using **jdmpview**.

  more on **jextract and **jdmpview** later**

How can -Xdiagnosticscollector help me? (cont'd)

- Pros:
  - Provides a one-step collection action, with additional actions optional, to collect all diagnostic data possible.
  - Creates a single .zip file containing all the diagnostic data.
- Cons:
  - Requires processing time to perform the collection
  - Requires additional space to contain the resulting .zip

How can -Xdiagnosticscollector help me? (cont'd)

- PRO or CON? A .zip file will be created for every problem event
  - Example: multiple OutOfMemory errors occur but the application continues to run. Each OOM will create a .zip file

Diagnostics collector zip file format

- General zipfile name format is:

  java.<event>.<YYMMDD.hhmmss.pid>.zip
  where event describes what triggered the diagnostics collector.
- Example:
  java.outofmemoryerror.
  20100803.171454.83952567.zip

What analysis can I do with a TDump?

- Tdumps can be analyzed using IPCS, much like any z/OS address space dump.
- Commands that might help
  - ip verbx ledata 'nthreads(*),asid(xxxx)'
    where xxxx is the asid in Hex
    - Formats the C stacks (DSAs) for threads in the asid
  - ip verbx ledata 'asid(xxxx),tcb(tttttt),ceedump'
    - Formats the C stack for a single TCB in the ASID

What analysis can I do with a TDump?

- The system trace table may help with diagnosing a looping/hanging/long running application threads
  - ip systrace asid(x'XXXX') time(gmt)
  - ip systrace asid(x'XXXX') time(local)
- IPCS is beneficial if the applications use JNI or are primarily written in Native code, but not Java

So what do I need to use to look at Java data in a TDump?

- **jextract and jdmpview**
- jextract
  - Platform-specific utility to extract and package (compress) data from the dump generated by the operating system
  - jextract **MUST** be executed using the **EXACT** same java version as was in use by the JVM within the dump. Otherwise, jextract will fail with ”This version of jextract is incompatible with this dump.”

So what do I need to use to look at Java data in a TDump?

- **jextract** is executed from the OMVS command line. The dump passed to the command can either be in the HFS or it may be the actual zOS dump dataset name.
- The command is :
  jextract <dumpname>
- The output will be a zip file named <dumpname>.zip, containing a copy of the dump, the .xml , and the .dat file

So what do I need to use to look at Java data in a TDump?

- The .zip file created by **jextract** will be used as input to **jdmpview**
- The command is :
  jdmpview -zip <dumpname>.zip
- **GOOD NEWS!** jdumpview does **NOT** require using the same jvm version as jextract.
  - Why? Because the .xml file in the zip is used for accessing the data within the dump

What commands should I use?

- start with “help” to display the command list
- info proc
  - Displays threads, arguments, envars, shared modules
- info thread *
  - Displays information about Java and Native threads
- info thread <thread address>
  - Displays information about a specific thread

Suggested Reading:

- Java 6 Diagnostics Guide
- http://www.ibm.com/developerworks/java/jdk/diagnosis/60.html

- QUESTIONS??????????????