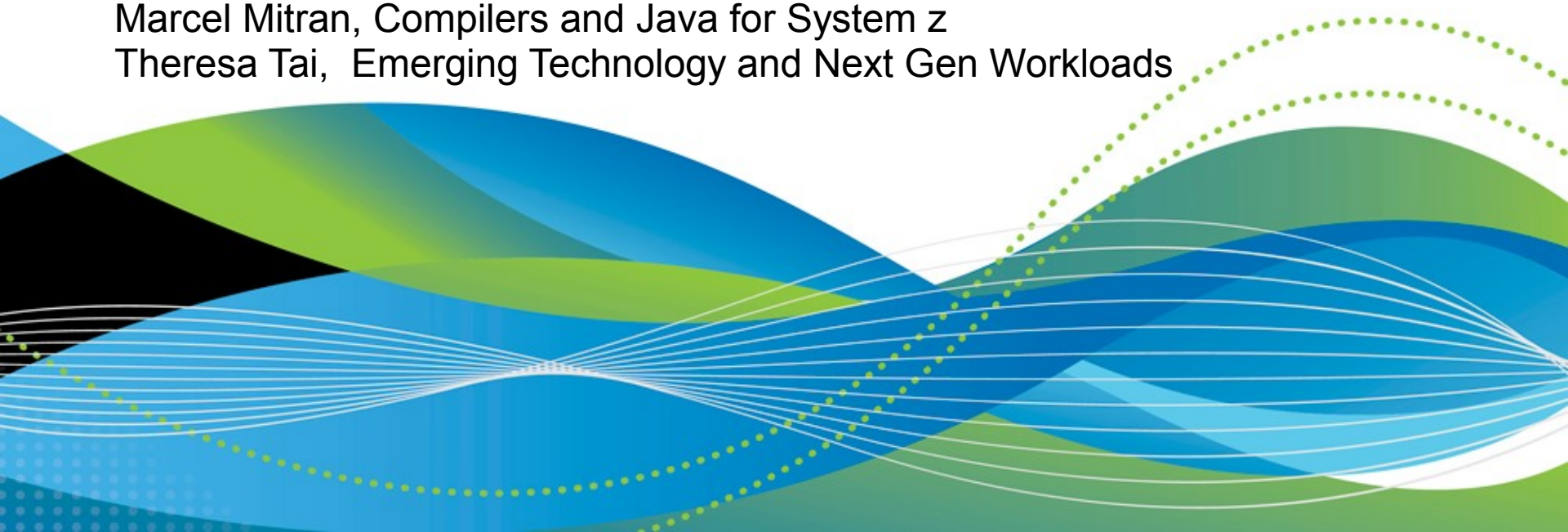
#SHAREorg

SHARE
Technology · Connections · Results

IBM Java 6.0.1 and 7.0 in zEnterprise and Technology Update

Session 10662

Ken Irwin, IBM, zOS Java L2 Service Support
Marcel Mitran, Compilers and Java for System z
Theresa Tai, Emerging Technology and Next Gen Workloads



Trademarks, Copyrights, Disclaimers

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Content

- IBM Java on Z (Marcel)
  - History, overview and roadmap
  - J9 R26 for Java601 and Java7
  - z196 exploitation and performance
- Java consumability and serviceability (Ken)

IBM Java Consumability and Serviceability

IBM J9 2.6 Technology Enhancements:
GC Policy Changes

Effective with Java R601

- Xgcpolicy: gencon is the default GC Policy
- Applies to both 31- and 64-bit JVMs

Benefits

- Long-lived objects are handled differently than short-lived objects
- Applications with many short-lived objects can see shorter pause times (better performance) while maintaining good throughput

Considerations

- Assess performance against current gcpolicy in use
- Consider returning to previous gcpolicy if needed

Other changes

- The subpool policy has been removed
- -Xgcpolicy:subpool is now an alias for -Xgcpolicy:optthruput

IBM J9 2.6 Technology Enhancements:
GenCon Policy Tuning

Gencon Defaults

- Default split is 25% newspace, 75% oldspace
- Both new and old space can grow and shrink dynamically

Old and New space size is definable

- -Xmns / -Xmnx
- -Xmos / -Xmox

Tuning Considerations

- In a fixed heap (ie -Xms = -Xmx), both areas are fixed in size
- Too small a nursery can cause
  too many local GCs
  premature tenuring >> more frequent Global GCs
- Too large a nursery can cause
  expensive local GCs
  smaller tenure space >> more frequent Global Gcs
- The tuning goal is to minimize the object survival rate
- If the rate is consistently too high, consider another policy
- Consider using the GCMV tool to help

IBM J9 2.6 Technology Enhancements:
RAS Enhancements

JVM Dump Support

- Environment Variables and ULIMITs included in javacores
- Native memory usage counters in javacore and from core dumps via DTFJ
- Multi-part TDUMPs on zOS 64

JVM Trace Support Improvements

- -Xtrace

  More internal tracepoints created
  New javastack and ceedump trigger points added

JVM Logging Improvements

- -Xlog

  New with Java 626, all JVM error messages (JVMxxxxnnnE) plus specific information messages JVMDUMP006I, JVMDUMP032I and JVMDUMP033I are written to the system log by default.
  The –Xlog option has been enhanced to provide controlled selection of Message routing to the system log.

IBM J9 2.6 Technology Enhancements:
RAS Enhancements: javacore contents

Environment Variables

--

_CXX_WORK_SPACE=(32000,(150,150))

_CXX_PMSGS=EDCPMSGE

MAIL=/usr/mail/CHAMBER

_CC_CNAME=CCNDRVR

PATH=/u/sovbld/bldsys:/usr/local/perl/bin:/u/java/bin:/bin:/usr/sbin:/u/chamb....

_C89_WORK_SPACE=(32000,(150,150))

_CXX_WORK_UNIT=SYSDA

_CXX_INCDIRS=/usr/include //DD:SYSLIB //'PP.ADLE370.ZOS180.SCEEH.NET.H'……

_C89_PNAME=EDCPRLK

TMPDIR=/tmp

SSH_CLIENT=9.20.183.84 1846 22

SHELL=/bin/sh

IBM J9 2.6 Technology Enhancements:
RAS Enhancements: javacore contents

ULIMITs

User Limits (in bytes except for NOFILE and NPROC)

---

type	soft limit	hard limit
RLIMIT_AS	unlimited	unlimited
RLIMIT_CORE	0	unlimited
RLIMIT_CPU	unlimited	unlimited
RLIMIT_DATA	unlimited	unlimited
RLIMIT_FSIZE	unlimited	unlimited
RLIMIT_LOCKS	unlimited	unlimited
RLIMIT_MEMLOCK	32768	32768
RLIMIT_NOFILE	1024	1024
RLIMIT_NPROC	16382	16382
RLIMIT_RSS	unlimited	unlimited
RLIMIT_STACK	10485760	unlimited
RLIMIT_MSGQUEUE	819200	819200

IBM J9 2.6 Technology Enhancements:
RAS Enhancements: javacore contents

- Native memory usage counters

```
NATIVEMEMINFO subcomponent dump routine
=======================================
JRE: 555,698,264 bytes / 1208 allocations
|
+--VM: 552,977,664 bytes / 856 allocations
||
|+--Classes: 1,949,664 bytes / 92 allocations
||
|+--Memory Manager (GC): 547,705,848 bytes / 146 allocations
|||
||+--Java Heap: 536,875,008 bytes / 1 allocation
|||
||+--Other: 10,830,840 bytes / 145 allocations
||
|+--Threads: 2,660,804 bytes / 104 allocations
|||
||+--Java Stack: 64,944 bytes / 9 allocations
|||
||+--Native Stack: 2,523,136 bytes / 11 allocations
|||
||+--Other: 72,724 bytes / 84 allocations
||
```

IBM J9 2.6 Technology Enhancements:
RAS Enhancements: tdump collection

- Multi-part TDUMPs
  - targeted for the 64bit JVM user
  - Existed in Java 6 also, so this should be a refresher

If you specify a template for the IEATDUMP file name, append the &DS token to enable multiple dumps. The &DS token is replaced by an ordered sequence number, and must be at the end of the file name. For example, X&DS generates file names in the form X001, X002, and X003.

- If you specify a template without the &DS token, .X&DS is appended automatically to the end of your template by the JVM. If your template is too long to append .X&DS, a message is issued. The message advises that the template pattern is too long and that a default pattern will be used.

- If you do not specify a template, the default template is used. The default template is: %uid.JVM.%job.D%y%m%d.T%H%M%S.X&DS

***Dump files MUST be merged before use with IPCS or JDmpview

IBM J9 2.6 Technology Enhancements:
RAS Enhancements: -Xlog

- Using –Xlog to control system log messages

-Xlog[:help]|[:<options>]
Sub-options are:
error log all error (JVMxxxnnnE) messages (default)
warn log all warning (JVMxxxnnnW) messages
info log all information (JVMxxxnnnI) messages
config log all configuration messages (there are none yet!)
vital log VM-selected messages, eg for location of dump files (default)
all log all messages
none turn off logging

Examples
To log error and warning messages
-Xlog:error,warn.
To turn off logging
-Xlog:none.

IBM J9 2.6 Technology Enhancements:
Verbosegc Updates

Verbosegc Format has Changed

- Still XML, but built on a different model

- Old format was transactional and nested
  - *Worked well for optthruput*
  - *Not so well for optavgpause and gencon*
- New format is event driven and flatter
  - *Provides a better fit for concurrent GC work*

Added BONUS! Verbose GC output now contains a listing of arguments passed to the JVM

- Helpful for diagnosing OOMs, long GC cycles, etc
- Helps when a matching javacore isn't available

Verbosegc: Old Format

- Pre R601 trace entry:

```
<con event="kickoff" timestamp="Mar 22 10:38:49 2011">
<kickoff reason="Kickoff threshold reached" />
</con>
<af type="nursery" id="16" timestamp="Mar 22 10:38:49 2011" intervalms="21.127">
<gc type="scavenger" id="17" totalid="19" intervalms="21.165">
</gc>
<time totalms="6.952" />
</af>
<con event="collection" id="1" timestamp="Mar 22 10:38:49 2011" intervalms="1526.710">
<gc type="global" id="2" totalid="19" intervalms="660.545">
<timesms mark="1.410" sweep="0.437" compact="0.000" total="4.002" />
</gc>
<time totalms="4.080" />
</con>
```

Verbosegc: New Format

- R601 trace entry:

```
<concurrent-kickoff id="235" timestamp="2011-03-22T10:40:51.701">
<kickoff reason="threshold reached" targetBytes="933651" thresholdFreeBytes="127933" />
</concurrent-kickoff>
<af-start id="240" totalBytesRequested="65544" timestamp="2011-03-22T10:40:51.705" intervalms="23.097" />
<gc-start id="242" type="scavenge" contextid="241" timestamp="2011-03-22T10:40:51.705">
<gc-op id="244" type="scavenge" timems="3.990" contextid="241" timestamp="2011-03-22T10:40:51.709">
<gc-end id="246" type="scavenge" contextid="241" durationms="4.161" timestamp="2011-03-22T10:40:51.709">
<af-end id="250" timestamp="2011-03-22T10:40:51.709" />
<concurrent-collection-start id="253" timestamp="2011-03-22T10:40:51.719" intervalms="1337.041" />
<gc-start id="254" type="global" contextid="237" timestamp="2011-03-22T10:40:51.719">
<gc-op id="258" type="mark" timems="1.433" contextid="237" timestamp="2011-03-22T10:40:51.722">
<gc-op id="259" type="sweep" timems="0.394" contextid="237" timestamp="2011-03-22T10:40:51.722" />
<gc-end id="260" type="global" contextid="237" durationms="3.277" timestamp="2011-03-22T10:40:51.723">
<concurrent-collection-end id="263" timestamp="2011-03-22T10:40:51.723" />
```

Inter-language Communication: Signal Handling with -XCEEHDLR (31-bit z/OS only)

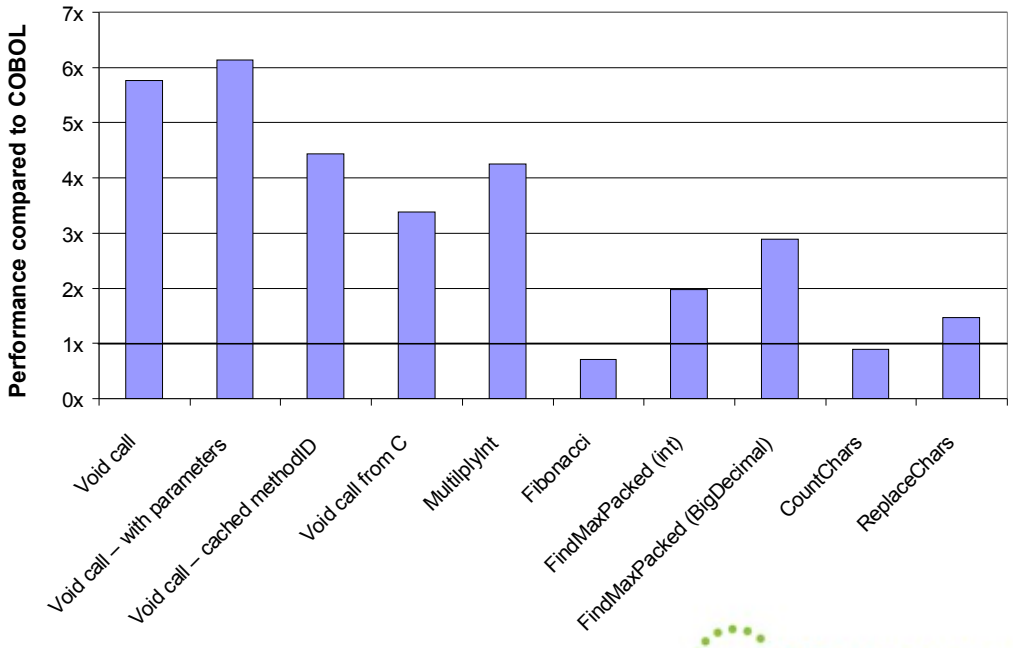
- New feature in Java 6.0.1 and 7.0
  - Requested by Java/COBOL interoperability batch mode environment
- Switches JVM from POSIX to LE Signal Handling for
  - SIGBUS
  - SIGFPE
  - SIGILL
  - SIGSEGV
  - SIGTRAP
- A condition triggered while executing a JNI component causes the JVM to convert the Language Environment condition into a Java ConditionException
  - Allows Java application to see/catch LE conditions
  - com.ibm.le.conditionhandling.ConditionException exception is thrown

Inter-language Communication
Java/COBOL Inter-Operability Performance

- COBOL → Java compared to COBOL → COBOL
  - Java void() method shows 6x more overhead
  - Caching methodID reduces overhead to 4.4x
  - Max operation on a set of packed decimals:
    - 2x slowdown when Java transformed decimals into ints
    - 2.9x slowdown when Java transformed decimals into BigDecimals
  - Fibonacci (42 adds in a loop) shows Java performs 40% better than COBOL

- +96% of the program is eligible for zAAP offload

- Best practices:
  - Do as much work in Java as possible
  - Have as few COBOL ↔ Java transitions as possible



Inter-language Communication: JNI Best Practices

- Common performance pitfalls
  1. Not caching method IDs, field IDs, and classes
     - Avoid redundent calls to `FindClass(), GetFieldID(), GetMethodId(), and GetStaticMethodID()`
  2. Triggering array copies
     - Assume arrays are buffered, hence be precise about which elements you really need to avoid needless copying
  3. Reaching back instead of passing parameters
     - When possible, flatten object fields into parameters of call. Avoid using JNI services to get to object fields
  4. Choosing the wrong boundary between native and Java code
     - Assume Native ⇔ Java call overhead 10x slower than Native ⇔ Native or Java ⇔ Java
  5. Using many local references without informing the JVM

See http://www.ibm.com/developerworks/java/library/j-jni/

Other changes of interest

- Service has requested a TDUMP be captured by default on OutOfMemory exceptions.
  *Enabled effective with Java601 SR1 and Java7

Jmpview no longer requires TDUMPs to be pre-processed by jextract.

- Eliminating the requirement for a matching jvm build level to that captured within the dump makes diagnosis simpler.
- Support has been backported to more current versions of Java6 and Java5, but requires using an R601 build level to execute jdmpview. Support exists beginning with:
  - Java6 SR9
  - Java5 SR12

More changes of interest....

- Effective with Java601 SR1, Jmpview provides a batch capability (versus interactive only) for command processing! To read about the changes, use this link to the Java 6 Supplement, and search on 'jdmpview'
- http://public.dhe.ibm.com/common/ssi/ecm/en/zsl03118usen/ZSL03118USEN.PDF

  Examples:
  - To execute a command , append it to the end of the jdmpview command line: to execute 'info class' command, use the following:
    jdmpview -core mycore.dmp info class
  - The wildcard * must be replaced with keyword ALL
  - Example: info thread * would be
    jdmpview -core mycore.dmp info thread ALL

Documentation

Download a copy of the

“ IBM SDK Java Technology Edition Version 6 Supplement” for more details:

http://public.dhe.ibm.com/common/ssi/ecm/en/zsl03118usen/ZSL03118USEN.PDF

R601 31-bit:

http://www-03.ibm.com/systems/z/os/zos/tools/java/products/sdk601_31.html

R601 64-bit:

http://www-03.ibm.com/systems/z/os/zos/tools/java/products/sdk601_64.html

Documentation

For Java 7 Documentation, follow this link:

http://www.ibm.com/developerworks/java/jdk/
docs/java7/zos/index.html

Documentation available is:

- IBM User Guide for Java v7 on z/OS
- IBM JVM Messages
- Security documentation for z/OS


Questions

New and existing supported Java products – z/OS

- IBM 31-bit SDK for z/OS, Java Technology Edition, Version 7.0
  - Web available on October 4, 2011 at Java SE 6 level
  - Product 5655-R31, supported on z/OS V1.10 and above
- IBM 64-bit SDK for z/OS, Java Technology Edition, Version 7.0
  - Web available on October 4, 2011 at Java SE 6 level
  - Product 5655-R32, supported on z/OS V1.10 and above
- Earlier Deliveries
  - IBM 31-bit SDK for z/OS, Java Technology Edition, Version 6.0.1
    - Web available on March 15, 2011 at Java SE 6 level
    - Product 5655-R31, supported on z/OS V1.10 and above
  - IBM 64-bit SDK for z/OS, Java Technology Edition, Version 6.0.1
    - Web available on March 15, 2011 at Java SE 6 level
    - Product 5655-R32, supported on z/OS V1.10 and above
  - IBM 31-bit SDK for z/OS, Java Technology Edition, Version 6.0.0
    - Web available on December 14, 2007 at Java SE 6 level
    - Product 5655-R31
  - IBM 64-bit SDK for z/OS, Java Technology Edition, Version 6.0.0
    - Web available on December 14, 2007 at Java SE 6 level
    - Product 5655-R32
  - IBM 31-bit SDK for z/OS, Java 2 Technology Edition, Version 5.0
    - Web available on November 30, 2005
    - Product 5655-N98
  - IBM 64-bit SDK for z/OS, Java 2 Technology Edition, Version 5.0
    - Web available on November 30, 2005
    - Product 5655-N99
- All products are delivered via the z/OS Java website in non-SMP/E format and via ShopIBM in SMP/E format
- All products are independently orderable and serviceable and follow the z/OS RFA rules for Withdrawal from Marketing and End of Service

Important references

- z/OS Java web site
  - http://www.ibm.com/systems/z/os/zos/tools/java/
- IBM SDK Java Technology Edition Version 7 Information Center
  - http://publib.boulder.ibm.com/infocenter/java7sdk/v7r0/index.jsp
- IBM SDK Java Technology Edition Version 6 Supplement
  - http://public.dhe.ibm.com/common/ssi/ecm/en/zsl03118usen/ZSL03118USEN.PDF
- JZOS Batch Launcher and Toolkit Installation and User's Guide (SA38-0696-00)
  - For JZOS function included in IBM Java SE 7 SDKs for z/OS
  - http://publibz.boulder.ibm.com/epubs/pdf/ajvc0110.pdf
- JZOS Batch Launcher and Toolkit Installation and User's Guide (SA23-2245-03)
  - For JZOS function included in IBM Java SE 6 and SE 5 SDKs for z/OS
  - http://publibfi.boulder.ibm.com/epubs/pdf/ajvc0103.pdf