IBM Java JVM Tuning For Maximum Performance

Iris Baron – IBM Java JIT Compiler Development

Session 17651
Wednesday, August 12, 2015: 10:00 AM - 11:00 AM
Dolphin, Americas Seminar

#SHAREorg

SHARE is an independent volunteer-run information technology association that provides **education**, professional **networking** and industry **influence.**

Objectives

- Why run Java on System z?
- Main performance features from zEC12 and Java7R1
  - zEDC
  - SMC-R
  - DAA
  - Large pages
- New performance features in IBM z13 and Java 8 and how they achieve superior performance
- Ramp-up performance
- Monitoring and diagnostic tools for Java

Java™ on System z®?

Naturally.

Two **pervasive technologies…**

There are **9 million** Java developers

80% of the world's corporate data resides on or originates on the **mainframe**

…combine for **powerful performance…**

15% increase in application performance

5x faster DB-response time

20% greater processing capacity

when DATEV eG ported business rules from a distributed server into CICS® Java

…that **everybody's talking about.**

> **z/OS is probably the most efficient place to run Java.**
>
> David Hodgson, techrepublic.com

> **You put the code where the data is, and you get to remove any network latency…**

> **Since the z9 was introduced, Java performance has exploded five times and it hasn't finished on that curve…**
>
> Scott Fagen, enterprisesystemsmedia.com

> **I've been impressed of late with the mainframe's Java support. It runs fast. It runs on the zAAPs. It runs all sorts of Java things without any recoding effort.**
>
> Scott Chapman, cmg.org

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 7.0

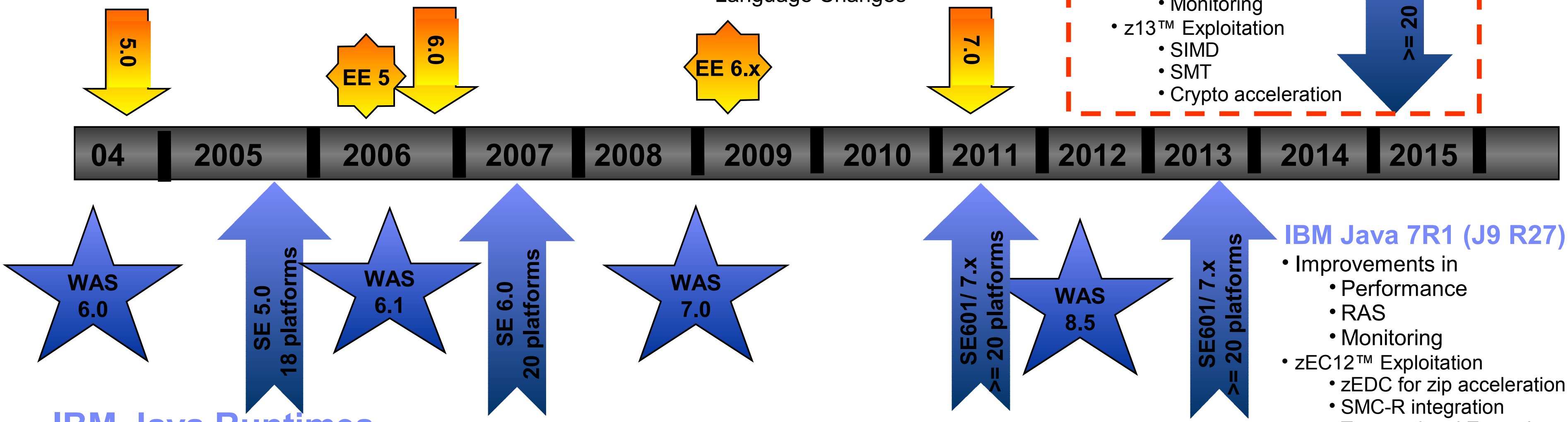
- Support for dynamic languages
- Improve ease of use for SWING
- New IO APIs (NIO2)
- Java persistence API
- JMX 2.x and WS connection for JMX agents
- Language Changes

Java 8.0

- Language improvements
- Closures for simplified fork/join

IBM Java 8 (J9 R28)

- Improvements in
  - Performance
  - RAS
  - Monitoring
- z13™ Exploitation
  - SIMD
  - SMT
  - Crypto acceleration



IBM Java 7R1 (J9 R27)

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

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 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 6.0.1/Java 7 (J9 R26)

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

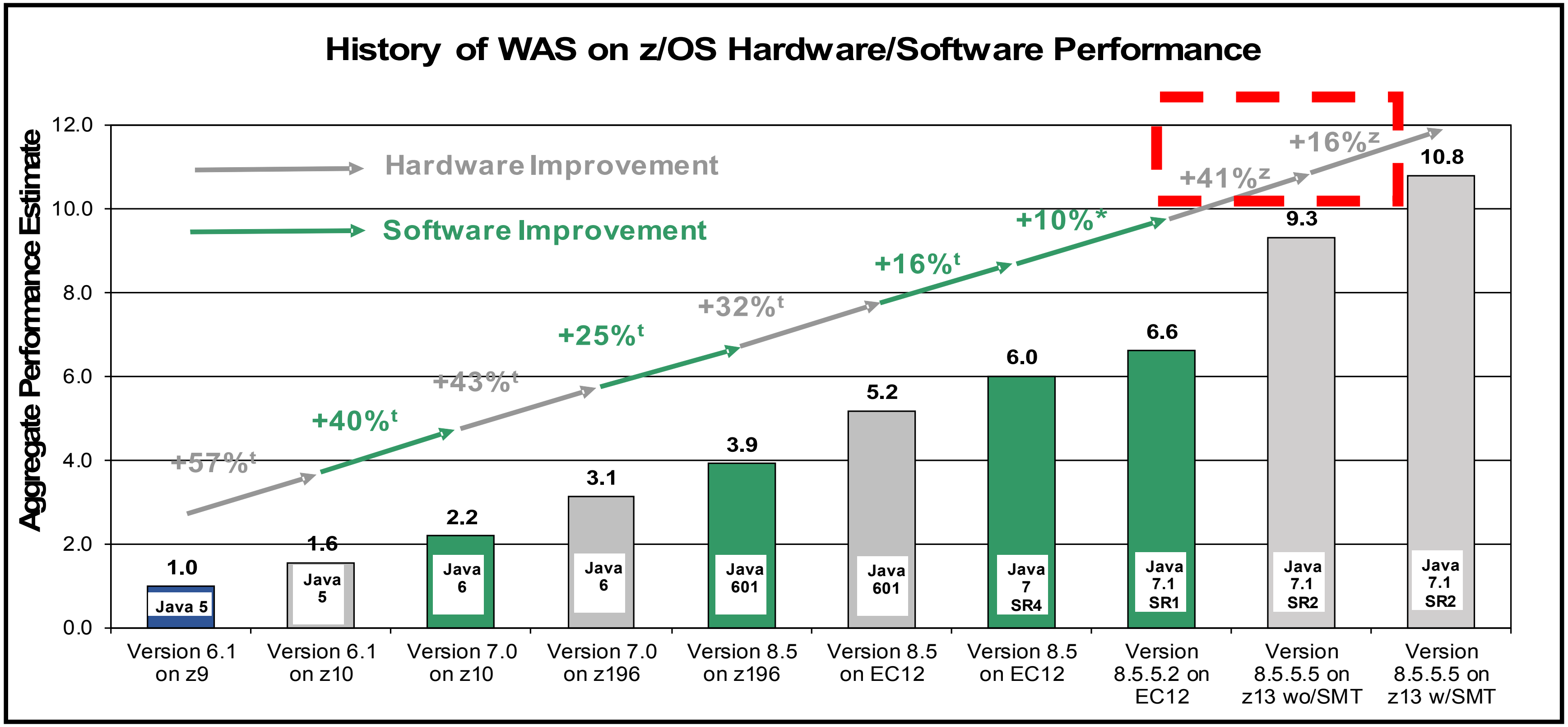
IBM Java 7 (J9 R26 SR3)

- Improvements in
  - Performance
- zEC12™ Exploitation
  - Transactional Execution
  - Flash 1Meg pageable LPs
  - 2G large pages
  - Hints/traps

**Timelines and deliveries are subject to change.

WAS on z/OS – DayTrader

Aggregate HW, SDK and WAS Improvement: WAS 6.1 (IBM Java 5) on z9 to WAS 8.5 (IBM Java 7R1) on zEC12



10.8x aggregate hardware and software improvement comparing

WAS 6.1 IBM Java5 on z9 to WAS 8.5.5.2 IBM Java7R1 on z13 w/SMT

z zIIPs DayTrader 3
* DayTrader3
t DayTrader2

(Controlled measurement environment, results may vary)

zEC12 – More Hardware for Java

Continued aggressive investment in Java on Z

Significant set of new hardware features tailored and co-designed with Java

Hardware Transaction Memory (HTM)

Better concurrency for multi-threaded applications
eg. ~2X improvement to juc.ConcurrentLinkedQueue

Run-time Instrumentation (RI)

Innovation new h/w facility designed for managed runtimes
Enables new expanse of JRE optimizations

2GB page frames

Improved performance targeting 64-bit heaps

Pageable 1M large pages with Flash Express

Better versatility of managing memory

Shared-Memory-Communication

RDMA over Converged Ethernet

zEnterprise Data Compression accelerator

gzip accelerator

New software hints/directives/traps

Branch preload improves branch prediction
Reduce overhead of implicit bounds/null checks

New 5.5 GHz 6-Core Processor Chip

Large caches to optimize data serving

Second generation OOO design

*Up-to **60%** improvement in throughput amongst Java workloads measured with zEC12 and IBM Java 7*

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

Engineered Together—IBM Java and zEC12 Boost Workload Performance
http://www.ibmsystemsmag.com/mainframe/trends/whatsnew/java_compiler/

IBM SDK for z/OS, Java Tech. Edition, Version 7 Release 1 (IBM Java 7R1)

http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=AN&subtype=CA&htmlfid=897/ENUS213-498&appname=USN

- **Expand zEC12/zBC12 exploitation**
  - More TX, instruction scheduler, traps, branch preload
  - Runtime instrumentation exploitation
  - zEDC exploitation through java/util/zip
  - Integration of SMC-R
- **Improved native data binding - Data Access Accelerator**
  - Integrated with JZOS native record binding framework
- **Improved general performance/throughput**
  - Up-to 19% improvement to throughput (ODM)
  - Up-to 2.4x savings in CPU-time for record parsing batch application
- **Improved WLM capabilities**
- **Improved SAF and cryptography support**
- **Additional reliability, availability, and serviceability (RAS) enhancements**
- **Enhanced monitoring and diagnostics**

Store your Data - zEnterprise Data Compression and IBM Java 7R1

What is it?

- ✓ *zEDC Express is an IO adapter that does high performance industry standard compression*
- ✓ *Used by z/OS Operating System components, IBM Middleware and ISV products*
- ✓ *Applications can use zEDC via industry standard APIs (zlib and Java)*
- ✓ *Each zEDC Express sharable across 15 LPARs, up to 8 devices per CEC.*
- ✓ *Raw throughput up to **1 GB/s** per zEDC Express Hardware Adapter*

Every day over 2000 petabytes of data are created

- Between 2005 to 2020, the digital universe will grow by 300x, going from 130 to 40,000 exa-bytes**
- 80% of world's data was created in last two years alone

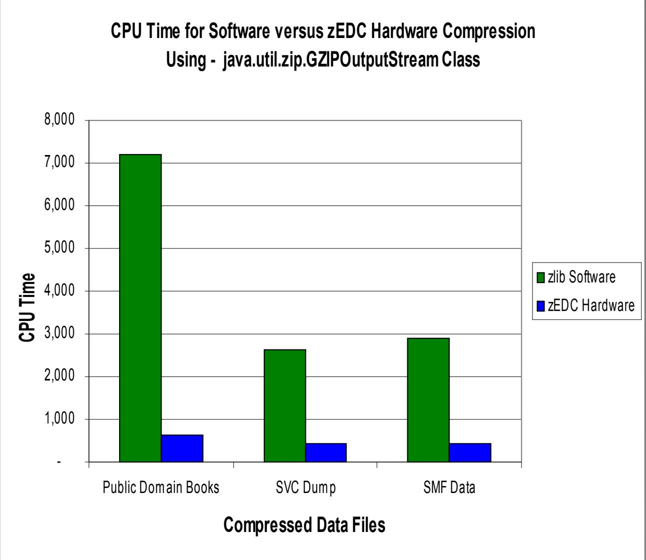
With **IBM Java 7R1** :

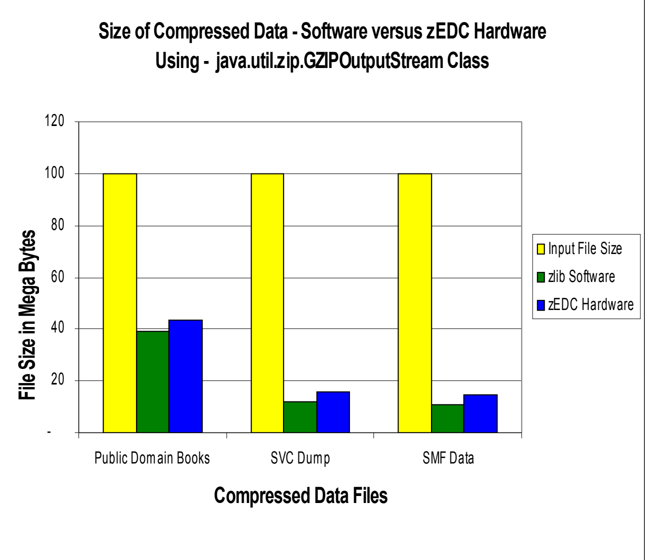
Java application to compress files using java.util.zip.GZIPOutputStream class

Up to 91% reduction in CPU time using zEDC hardware versus zlib software

Up to 74% reduction in Elapsed time (not shown)

Compression ratio up-to ~5x





Complete your session evaluations online at www.SHARE.org/Orlando-Eval

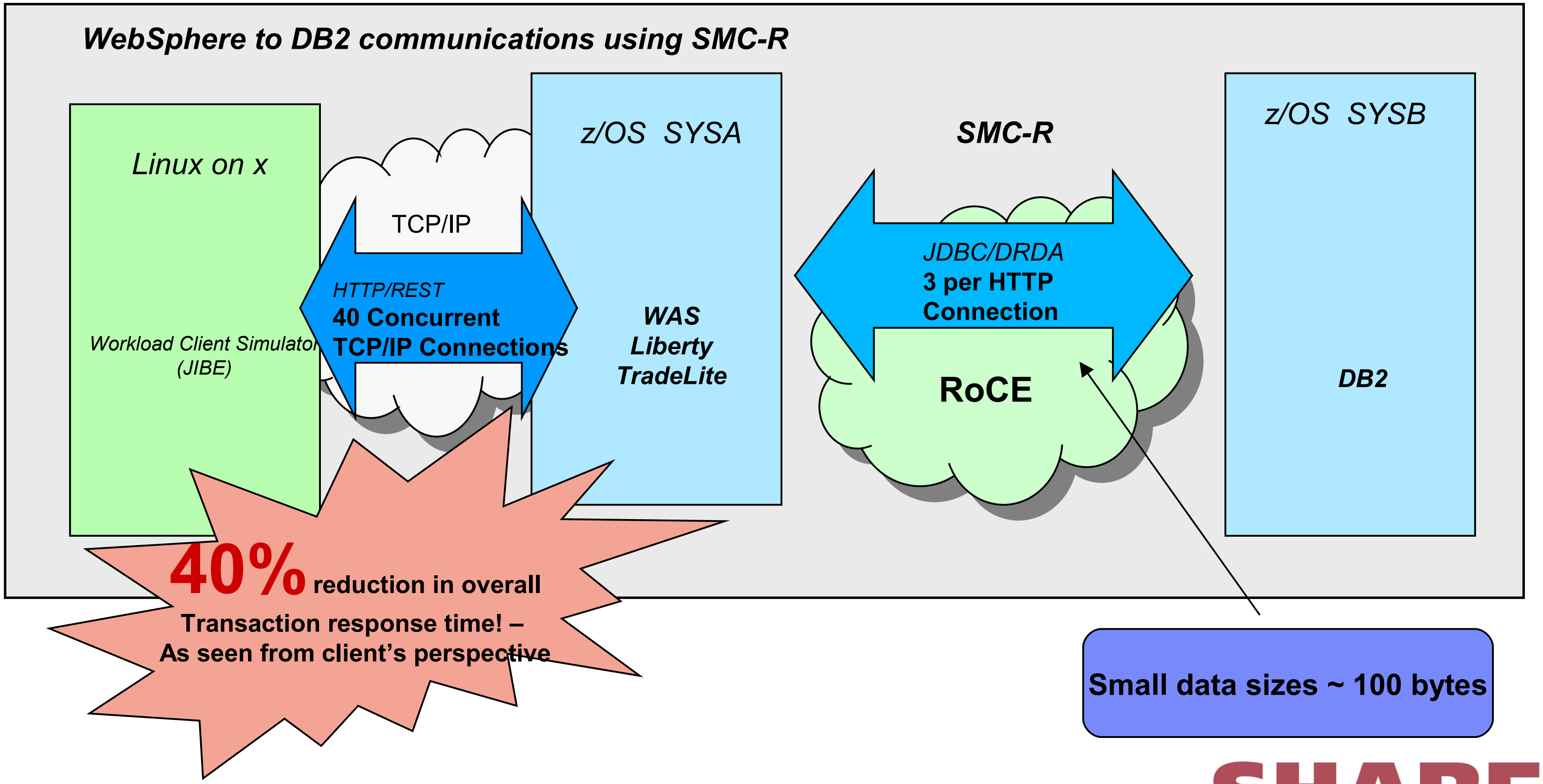
(Controlled measurement environment, results may vary)

** IDC: The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East

Move your Data - Shared Memory Communications (SMC-R)

RDMA Enables a host to read or write directly from/to a remote host's memory ***without*** **involving the remote host's CPU**

SMC-R automatically/transparently exploits RDMA/RoCE for sockets based TCP applications


WebSphere to DB2 communications using SMC-R
Linux on x
Workload Client Simulator (JIBE)
TCP/IP
HTTP/REST
40 Concurrent TCP/IP Connections
z/OS SYSA
WAS Liberty TradeLite
SMC-R
JDBC/DRDA
3 per HTTP Connection
RoCE
z/OS SYSB
DB2

40% reduction in overall Transaction response time! – As seen from client's perspective

Small data sizes ~ 100 bytes

(Controlled measurement environment, results may vary)

Transform your Data - Data Access Accelerator in IBM Java 7R1

A Java library for bare-bones data conversion and arithmetic

Operates directly on byte arrays

No Java object tree created

Orchestrated with JIT for deep platform opt.

Avoids expensive Java object instantiation

Library is platform and JVM-neutral

Marshalling and Un-marshalling

Transform primitive type (short, int, long, float, double) ⇔ byte array

Support both big/little endian byte arrays

Packed Decimal (PD) Operations

Arithmetic:	+, -, *, /, % on 2 PD operands
Relation:	>,<,>=,<=,==,!= on 2 PD operands
Error checking:	checks if PD operand is well-formed
Other:	shifting, and moving ops on PD operand

Decimal Data Type Conversions

Decimal ⇔ Primitive:	Convert Packed Decimal(PD), External Decimal(ED), Unicode Decimal(UD) ⇔ primitive types (int, long)
Decimal ⇔ Decimal:	Convert between dec. types (PD, ED, UD)
Decimal ⇔Java:	Convert dec. types (PD, ED, UD) ⇔ BigDecimal, BigInteger

Current Approach:

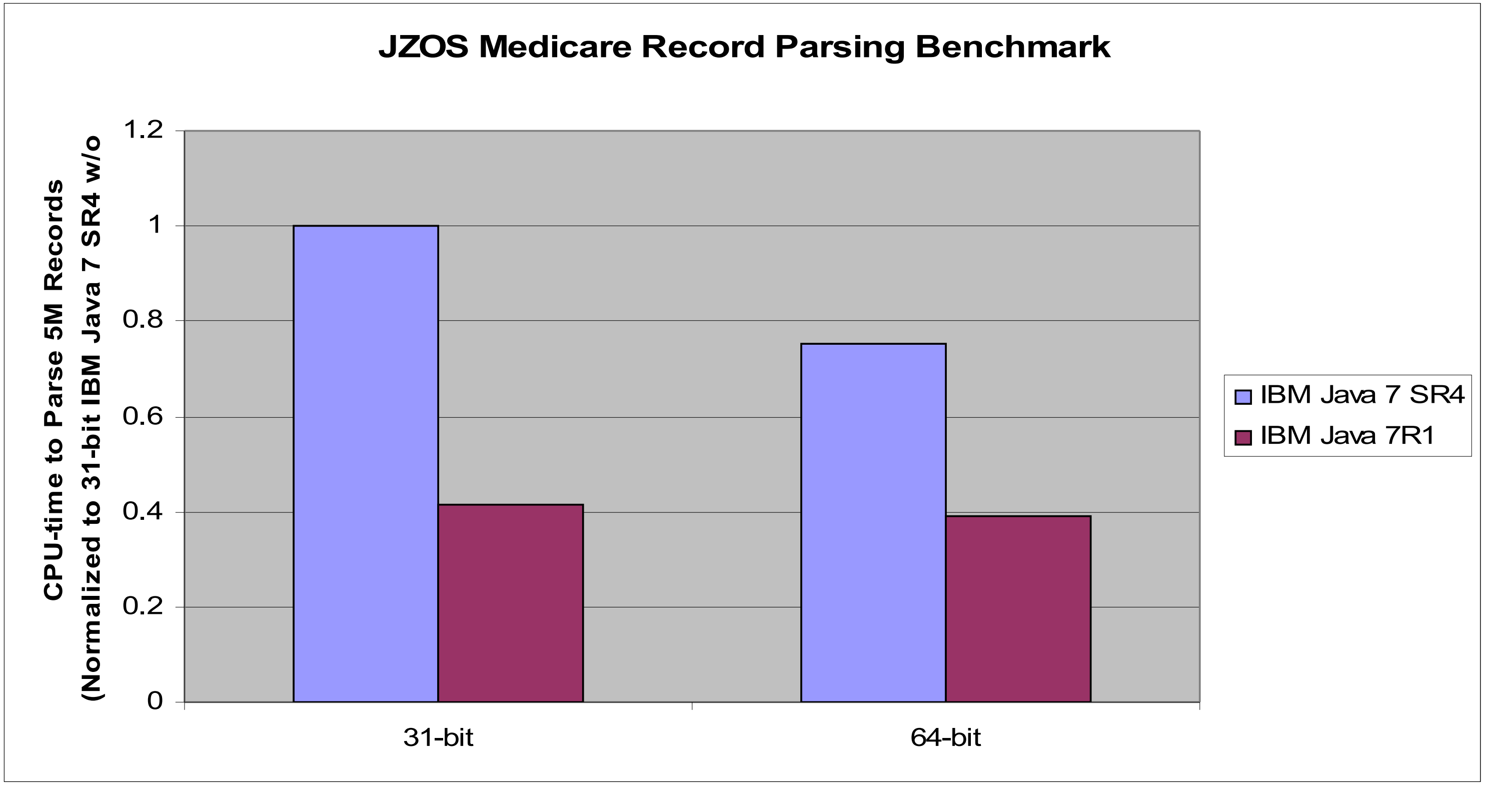
```
byte[] addPacked(array a[], array b[]) {
      BigDecimal a_bd = convertPackedToBd(a[]);
      BigDecimal b_bd = convertPackedToBd(b[]);
      a_bd.add(b_bd);
return  (convertBDtoPacked(a_bd));
}
```

Proposed Solution:

```
byte[] addPacked(array a[], array b[]) {
      DAA.addPacked(a[], b[]);
return  (a[]);
}
```

Data Access Accelerator

JZOS Medicare Record Benchmark and IBM Java 7R1

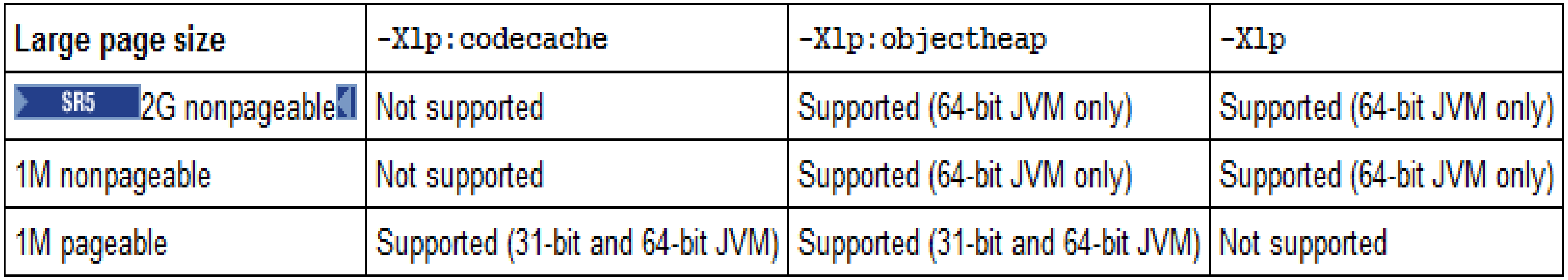


- 31-bit IBM Java 7R1 with DAA versus IBM Java 7 CPU Time improved by by 2.4x
- 64-bit IBM Java 7R1 with DAA versus IBM Java 7 CPU Time improved by by 1.9x

http://www.ibm.com/developerworks/java/zos/javadoc/jzos/index.html?com/ibm/jzos/sample/fields/MedicareRecord.html

(Controlled measurement environment, results may vary)

IBM Java - Large Pages



Large page size	-Xlp:codecache	-Xlp:objectheap	-Xlp
SR5 2G nonpageable	Not supported	Supported (64-bit JVM only)	Supported (64-bit JVM only)
1M nonpageable	Not supported	Supported (64-bit JVM only)	Supported (64-bit JVM only)
1M pageable	Supported (31-bit and 64-bit JVM)	Supported (31-bit and 64-bit JVM)	Not supported

- **z/OS 31 and 64 bit Java 7 SR3**
  1M Pageable large pages for JIT code cache and Java heap:
  - -Xlp:codecache:pagesize=1m,pageable
  - -Xlp:objectheap:pagesize=1m,pageable
  - No RACF Facility Class required
  - No z/OS IEASYSxx LFAREA required
  - Requires **zEC12 with FLASH Express**® feature (#0402) plus z/OS 1.13 offerings
  - **New default in Java 7 SR4**
  - Controlled by PAGESCM=**ALL** | NONE in the IEASYSxx parmlib

- **z/OS 64 bit Java 7 SR5**
  2G nonpageable large pages for Java heap:
  - -Xlp:objectheap:pagesize=2G,nonpageable
  - Requires zEC12 and z/OS 1.13 offerings
  - LFAREA in IEASYSxx parmlib member controls 2G nonpageable large pages
  - Must be authorized to the IARRSM.LRGPAGES resource in the RACF (or an equivalent security product) FACILITY class with read authority

z/OS Java CompressedRefs and Large Page Usage

-verbose:sizes

```
    -Xlp:objectheap:pagesize=1M,nonpageable
                available large page sizes:
                4K pageable
                1M pageable
                1M nonpageable
                2G nonpageable
```

large page size for Java heap

```
     -Xlp:codecache:pagesize=1M,pageable
                available large page sizes for JIT code cache:
                4K pageable
                1M pageable
```

large page size for JIT code cache

-verbose:gc

```
    <attribute name="compressedRefsDisplacement" value="0x0" />
     <attribute name="compressedRefsShift" value="0x0" />
     <attribute name="pageSize" value="0x100000" />
     <attribute name="pageType" value="nonpageable" />
     <attribute name="requestedPageSize" value="0x100000" />
     <attribute name="requestedPageType" value="nonpageable" />
```

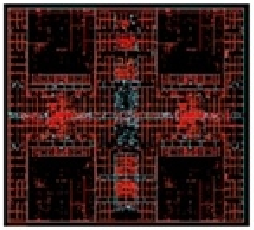
for heap size 2048M or smaller

Example: java -Xlp -Xlp:codecache:pagesize=1M,pageable -Xcompressedrefs -Xmx2048M

http://www-01.ibm.com/support/knowledgecenter/SSYKE2_7.0.0/com.ibm.java.zos.70.doc/user/alloc_large_page.html?cp=SSEQTP_8.5.5%2F7-5-5-4&lang=en

z Systems Processor Roadmap

z10
2/2008



Workload Consolidation and Integration Engine for CPU Intensive Workloads

Decimal FP

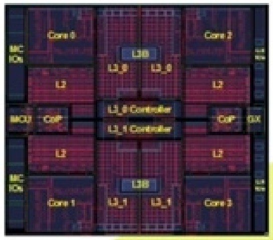
Infiniband

64-CP Image

Large Pages

Shared Memory

z196
9/2010



Top Tier Single Thread Performance,System Capacity

Accelerator Integration

Out of Order Execution

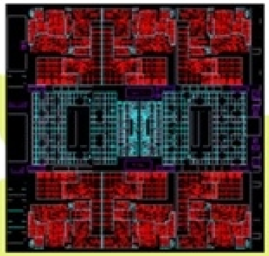
Water Cooling

PCIe I/O Fabric

RAIM

Enhanced Energy Management

zEC12
8/2012



Leadership Single Thread, Enhanced Throughput

Improved out-of-order

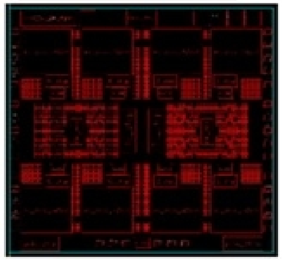
Transactional Memory

Dynamic Optimization

2 GB page support

Step Function in System Capacity

z13
1/2015



Leadership System Capacity and Performance

Modularity & Scalability

Dynamic SMT

Supports two instruction threads

SIMD

PCIe attached accelerators (XML)

Business Analytics Optimized

IBM z13 – Taking Java Performance to the Next Level

Continued aggressive investment in Java on Z

Significant set of new hardware features tailored and co-designed with Java

Simultaneous Multi-Threading (SMT)

- *2x hardware threads/core for improved throughput*
- *Available on zIIPs and IFLs*

Single Instruction Multiple Data (SIMD)

- *Vector processing unit*
- *Accelerates loops and string operations*

Cryptographic Function (CPACF)

- Improved performance of crypto co-processors

New Instructions

- Packed Decimal ⇔ Decimal Floating Point
- Load Immediate on Condition
- Load Logical and Zero Rightmost Byte

New 5.0 GHz 8-Core Processor Chip

480Mb L4 cache to optimize for data serving

Up to **50%** improvement in throughput for generic applications

Up to **2X** improvement in throughput per core for security enabled applications

- z13 toleration for Linux on z:
  - Java 7.1 SR2
  - Java 7 SR8
  - Java 6.1 SR8 FP2
  - Java6 SR16 FP2
- z13 toleration for z/OS is transparent

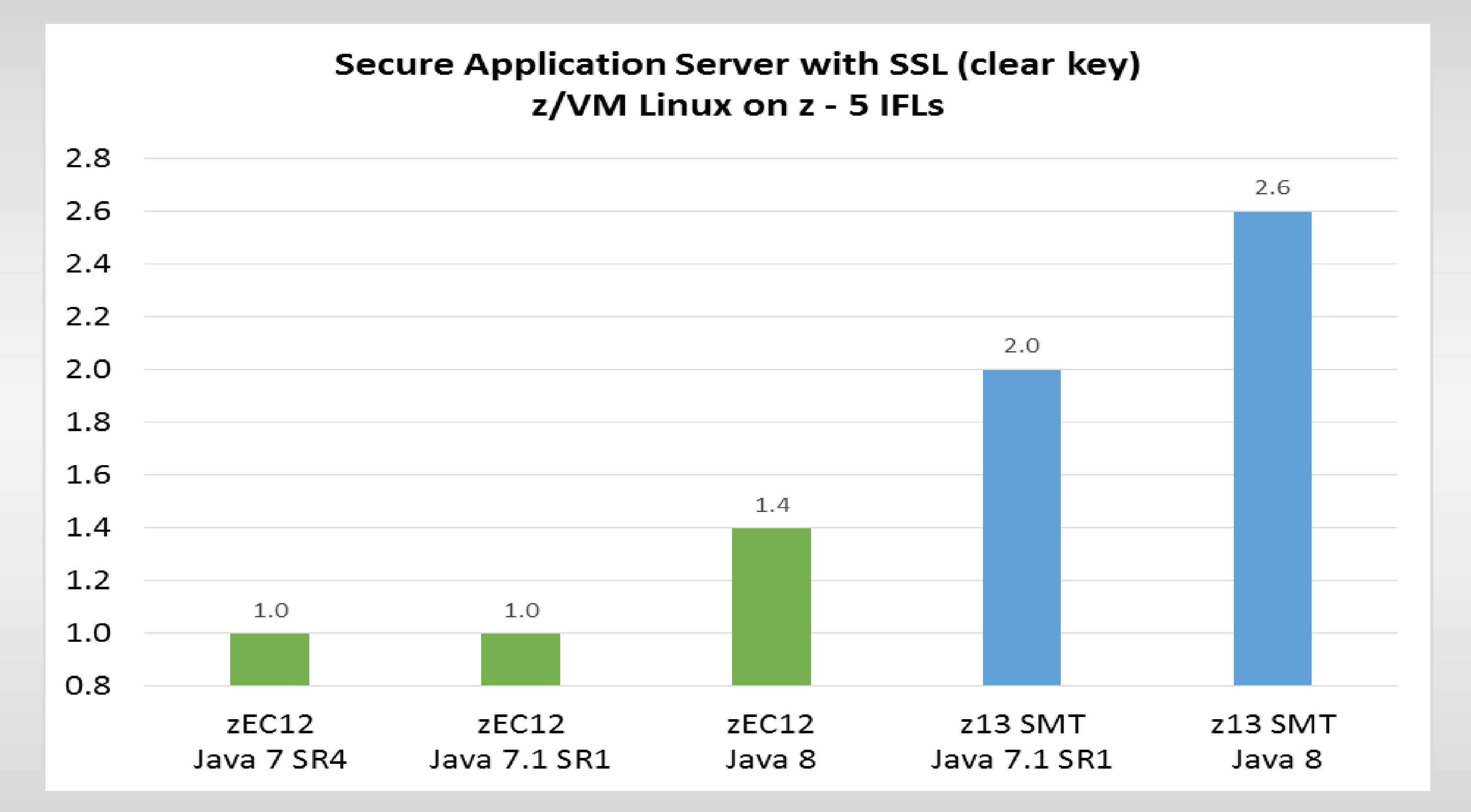
IBM SDK Java Tech. Edition, Version 8 (IBM Java 8)

- **New Java8 Language Features**
  - Lambdas, virtual extension methods
- **IBM z13 exploitation**
  - Vector exploitation and other new instructions
  - Instruction scheduling
- **General throughput improvements**
  - Up-to 17% better application throughput
  - Significant improvements to ORB
- **Improved crypto performance for IBMJCE**
  - Block ciphering, secure hashing and public key
    - Up-to 4x improvement to Public Key using ECC
    - CPACF instructions: AES, 3DES, SHA1, SHA2, etc
- **Significantly improved application ramp-up**
  - Up-to 50% less CPU to ramp-up to steady-state
  - Improved perf of ahead-of-time compiled code
- **Improved Monitoring**
  - JMX beans for precise CPU-time monitoring
- **Enhancements to JZOS Toolkit for Java batch**

SMT and SIMD Availability

	z/OS	z/VM	Linux on z - native
SMT	✓z/OS 2.1 with PTFs on zIIPs	✓on IFLs (Linux on z) ✓z/VM V6.3 and up	– Future RHEL7.1 and SLES12 update *Plan 3Q2015
SIMD	✓z/OS 2.1 with PTFs	– Not yet supported	– Future RHEL7.1 and SLES12 update *Plan 3Q2015

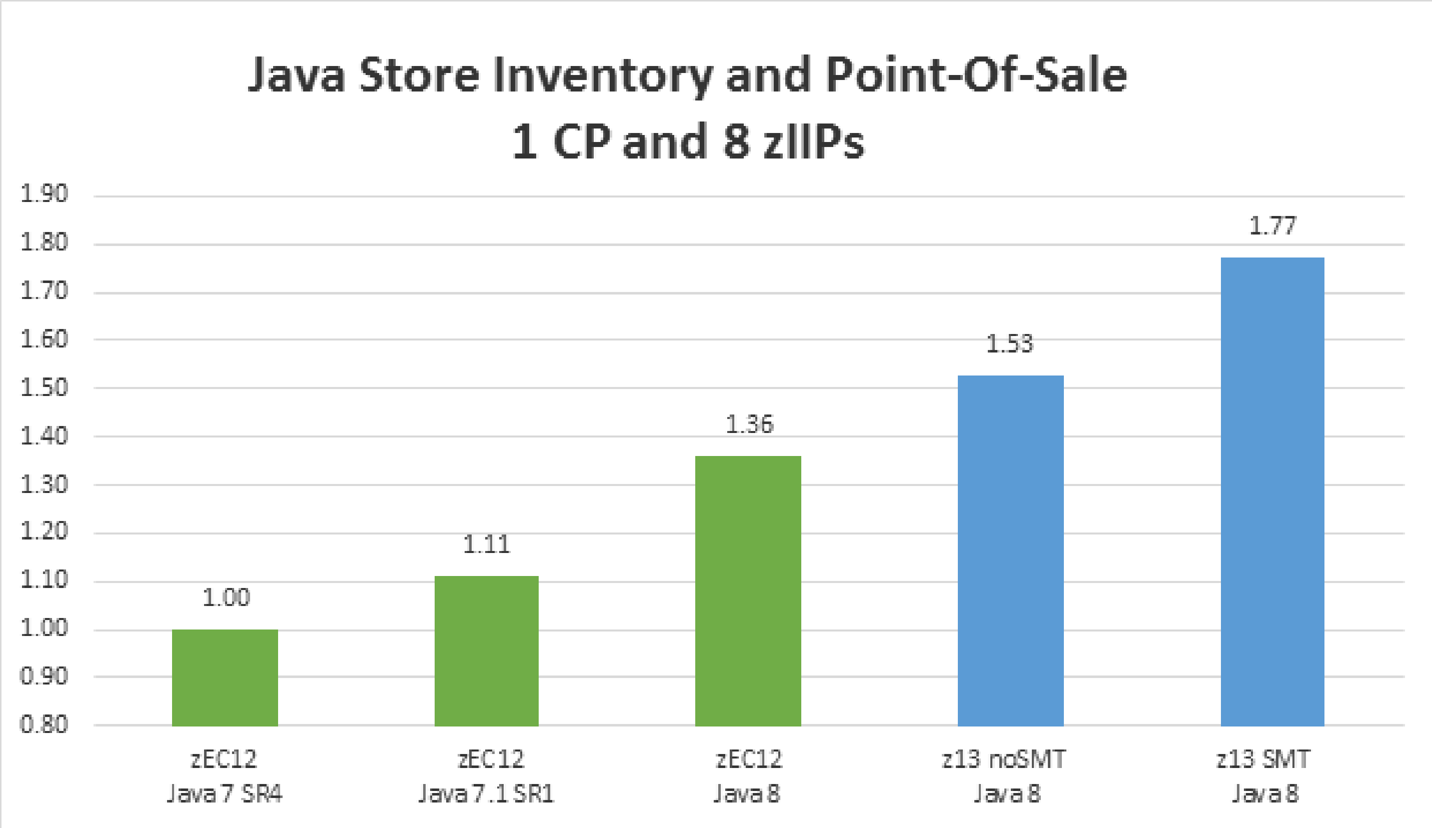
z/VM IFLs WAS8.5.5.5 Liberty - SSL-Enabled DayTrader 3.0



z/VM Linux on z - 2.6x improvement in throughput with IBM Java 8 and IBM z13

(Controlled measurement environment, results may vary)

Java Store, Inventory and Point-of-Sale App with IBM Java 8 and IBM z13

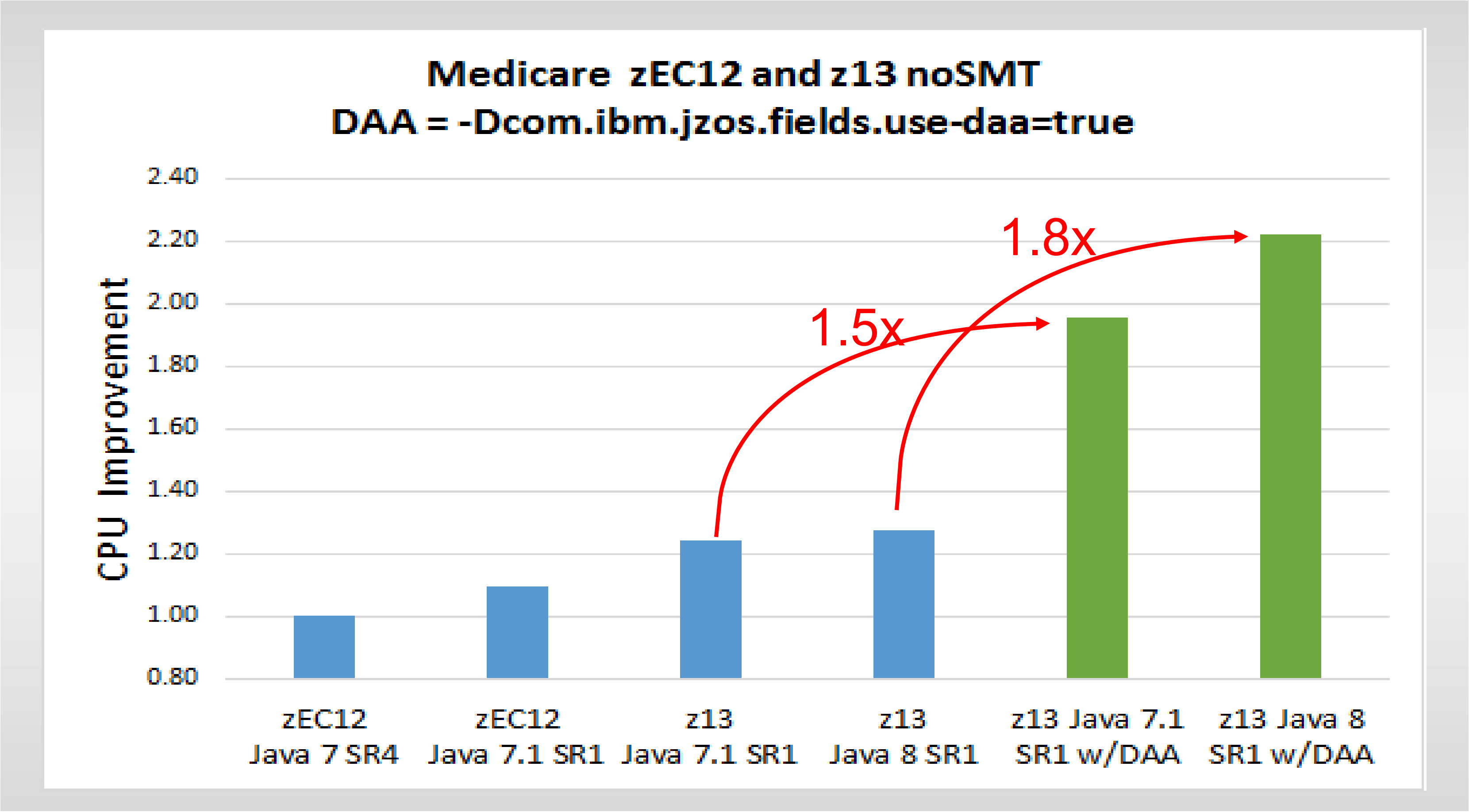


1.77x improvement in throughput with IBM Java 8 and IBM z13

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

(Controlled measurement environment, results may vary)

z/OS Data Access Accelerator



- **Aggregate 2.2x improvement from DAA with IBM Java8 and z13**
- **83% improvement from DAA on Java8 (vs 55% with Java7.1 SR1) on z13**

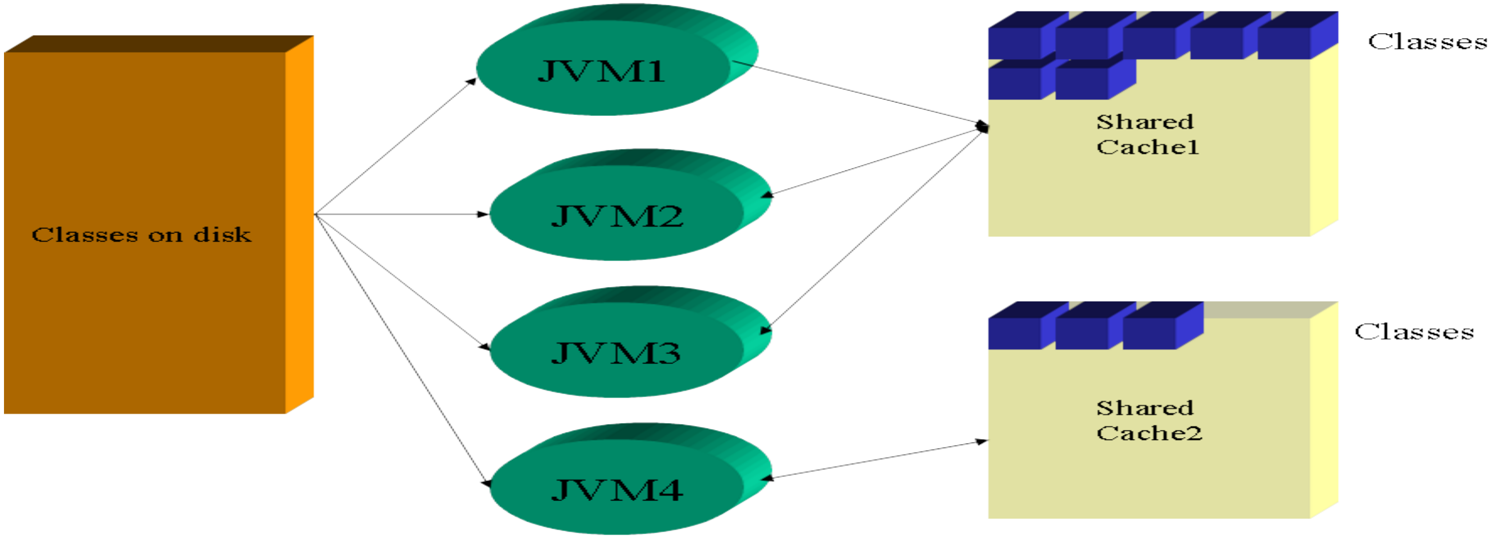
(Controlled measurement environment, results may vary)

Ramp-up Performance:
Dynamic Runtime

- Java Runtime Environment **dynamically** loads Java classes at runtime
- The Java Virtual Machine (JVM) can:
  - Interpret Java methods’ bytecodes
  - Compile Java methods’ bytecodes into assembly instructions
- Compilations
  - Pros:
    - Compiles only the methods that matter
    - Profiles your application characteristics for better optimizations
    - Optimize for your exact hardware
  - Cons:
    - Compilation ➔ runtime overhead

Ramp-up Performance:
Shared Classes

- Store classes into a cache that shared by multiple JVMs
- Read-only portions of the class
- Memory footprint reduction
- Startup time improvements (class initialization)
- Cache memory page protection (read-only caches)



Ramp-up Performance:
Ahead of Time Compilations

- Compiled code generated “ahead-of-time” (AOT)
- Subsequent JVMs can simply load this AOT code
- Startup time improvements
- CPU utilization reduction
- Persisted into the same shared cache

Ramp-up Performance:
Why not AOT everything?

- Dynamic class loading is a fundamental feature of Java
  - JVM must support dynamic class loading to pass Java certification
  - Classes may not even be stored on disk: Built on-the-fly from raw bytes, e.g. Java serialization and reflection services
  - Classes can be transformed on loading to insert new code or adjust existing code

- Dynamic class loading means constraints on compilation change
  - Re-use of a compilation requires verification that same conditions exists in new instance

Ramp-up Performance:
AOT/JIT Best Practices

- JIT modes:

Default	balanced throughput, startup, rampup (server-side)
-Xquickstart	faster startup, reduced throughput (client-side)
-Xtune:virtualized	reduced compilation overhead, reduced throughput

- Many diagnostic knobs, **not** for performance tuning
  - Impose a new compilation count
  - Impose optimization level
  - Limit compilation to a specific set of methods

- Tuning the shared classes cache size (-Xscmx)

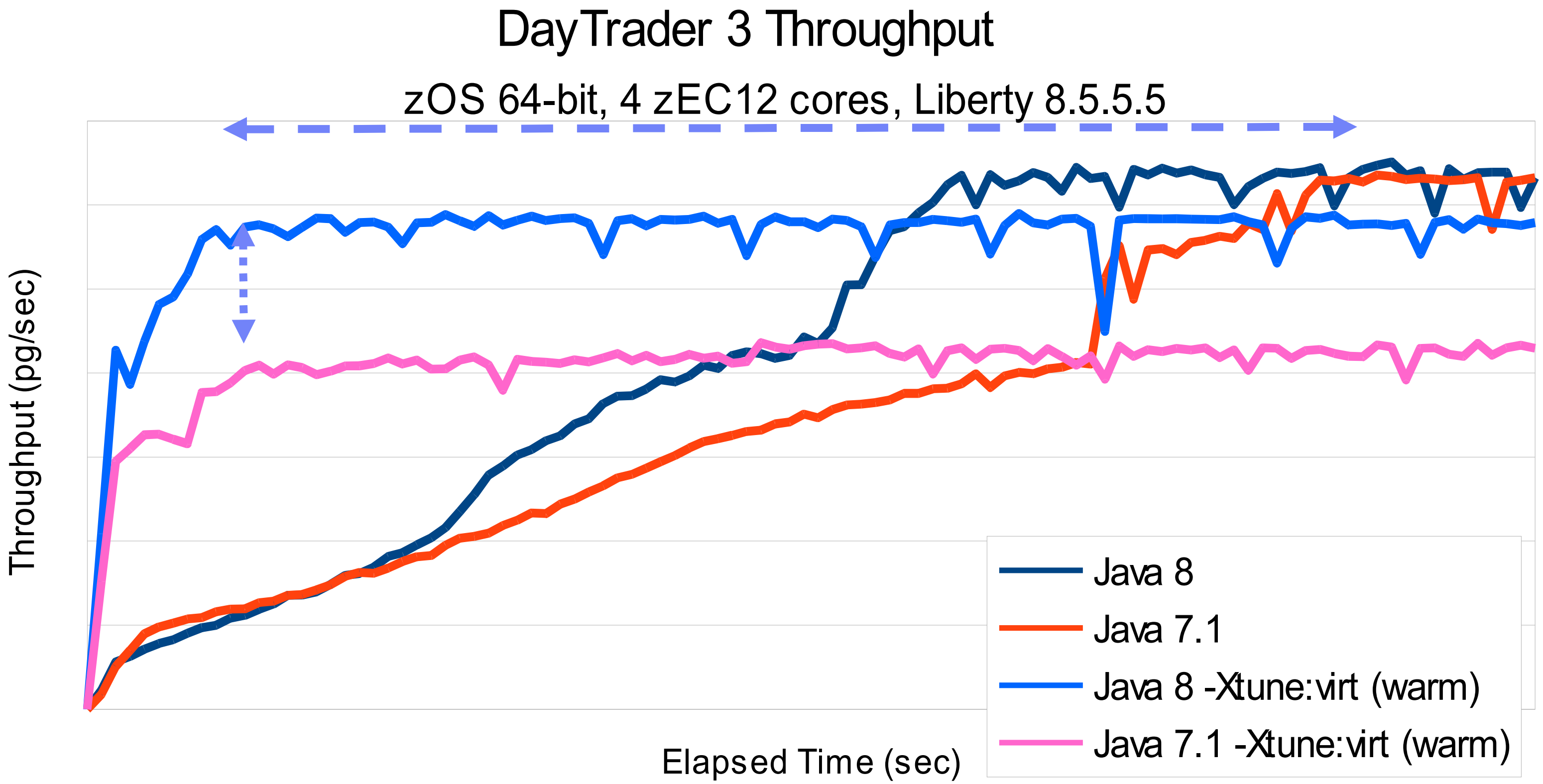
- http://www-01.ibm.com/support/knowledgecenter/SSYKE2_8.0.0/com.ibm.java.lnx.80.doc/user/classdatasharing.html?lang=en
- http://www-01.ibm.com/support/knowledgecenter/SSYKE2_8.0.0/com.ibm.java.lnx.80.doc/diag/understanding/jit_faq.html?lang=en

Ramp-up Performance: Runtime Instrumentation

- Hardware profiling infrastructure added in zEC12
- Low overhead, highly granular data
- Designed and built for dynamic runtimes like Java!

- How is the JIT using Runtime Instrumentation (RI)?
  - Software-based sampling is challenged in detecting ‘hot’ methods in large, flat Java applications
    - Tens of thousands of compilations → overhead
  - RI provides more granular data
    - JIT initially compiles using cheaper optimizations.
    - RI data to identify ‘important’ methods to recompile

zOS Liberty Ramp-up with IBM Java8

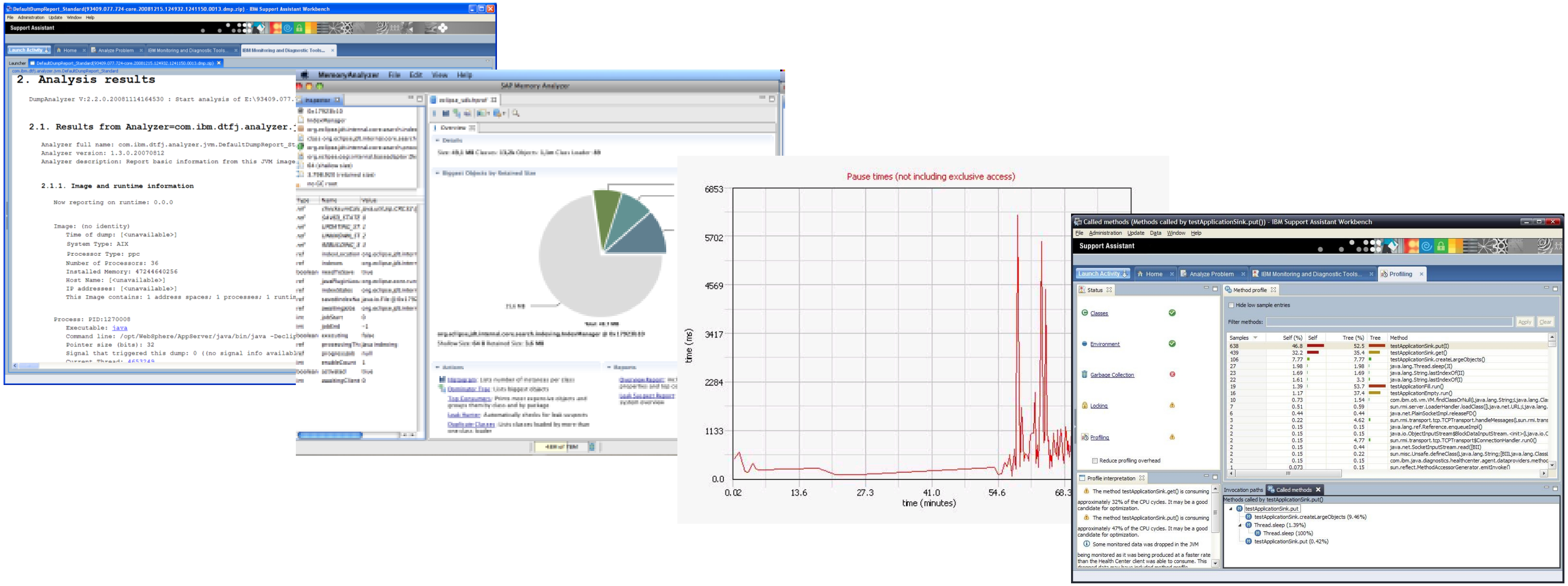


- IBM Java8 with –Xtune:virtualized improves DayTrader3/Liberty 8.5.5.5 ramp-up by 88%
- Default IBM Java8 vs IBM Java7.1 ramp-up improved by 22%

(Controlled measurement environment, results may vary)

IBM Monitoring and Diagnostic Tools for Java

- A ***free*** unified suite of tools to understand various aspects of Java applications
- Provides ***visualizations*** as well as ***recommendations***
- Provides APIs to enable you to ***extend*** the tools or ***create your own***
- Fully IBM ***supported*** and now ***updated*** for Java 8



DefaultDumpReport_Standard(93409.077.724-core.20081215.124932.1241150.0013.dmp.zip) - IBM Support Assistant Workbench
File Administration Update Window Help
Support Assistant
Launch Activity
Home
Analyze Problem
IBM Monitoring and Diagnostic Tools...
IBM Monitoring and Diagnostic Tools...
Launcher
DefaultDumpReport_Standard(93409.077.724-core.20081215.124932.1241150.0013.dmp.zip)
com.ibm.dtfj.analyzer.jvm.DefaultDumpReport_Standard

```
2. Analysis results

  DumpAnalyzer V:2.2.0.20081114164530 : Start analysis of E:\93409.077.

  2.1. Results from Analyzer=com.ibm.dtfj.analyzer.

    Analyzer full name: com.ibm.dtfj.analyzer.jvm.DefaultDumpReport_St
    Analyzer version: 1.3.0.20070812
    Analyzer description: Report basic information from this JVM image

    2.1.1. Image and runtime information

      Now reporting on runtime: 0.0.0

      Image: (no identity)
        Time of dump: [<unavailable>]
        System Type: AIX
        Processor Type: ppc
        Number of Processors: 36
        Installed Memory: 47244640256
        Host Name: [<unavailable>]
        IP addresses: [<unavailable>]
        This Image contains: 1 address spaces; 1 processes; 1 runtin

      Process: PID:1270008
        Executable: java
        Command line: /opt/WebSphere/AppServer/java/bin/java -Declip
        Pointer size (bits): 32
        Signal that triggered this dump: 0 ((no signal info availabl
        Current Thread: 4653249
```

□ Reduce profiling overhead

□ Hide low sample entries

Filter methods:

Samples	Self (%)	Self	Tree (%)	Tree	Method
638	46.8		52.5		testApplicationSink.put(I)
439	32.2		35.4		testApplicationSink.get()
106	7.77		7.77		testApplicationSink.createLargeObjects()
27	1.98		1.98		java.lang.Thread.sleep(JI)
23	1.69		1.69		java.lang.String.lastIndexOf(II)
22	1.61		3.3		java.lang.String.lastIndexOf(I)
19	1.39		53.7		testApplicationFill.run()
16	1.17		37.4		testApplicationEmpty.run()
10	0.73		1.54		com.ibm.oti.vm.VM.findClassOrNull(Ljava.lang.String;Ljava.lang.Cla
7	0.51		0.59		sun.rmi.server.LoaderHandler.loadClass([Ljava.net.URL;Ljava.lang.
6	0.44		0.44		java.net.PlainSocketImpl.releaseFD()
3	0.22		4.62		sun.rmi.transport.tcp.TCPTransport.handleMessages(Lsun.rmi.trans
2	0.15		0.15		java.lang.ref.Reference.enqueueImpl()
2	0.15		0.15		java.io.ObjectInputStream$BlockDataInputStream.<init>(Ljava.io.C
2	0.15		4.77		sun.rmi.transport.tcp.TCPTransport$ConnectionHandler.run0()
2	0.15		0.44		java.net.SocketInputStream.read([BII)
2	0.15		0.22		sun.misc.Unsafe.defineClass(Ljava.lang.String;[BIILjava.lang.ClassL
2	0.15		0.15		com.ibm.java.diagnostics.healthcenter.agent.dataproviders.methoc
1	0.073		0.15		sun.reflect.MethodAccessorGenerator.emitInvoke()

Profile interpretation

The method testApplicationSink.get() is consuming approximately 32% of the CPU cycles. It may be a good candidate for optimization.

The method testApplicationSink.put() is consuming approximately 47% of the CPU cycles. It may be a good candidate for optimization.

Some monitored data was dropped in the JVM being monitored as it was being produced at a faster rate than the Health Center client was able to consume. This dropped data may have included method profile

Methods called by testApplicationSink.put()

- testApplicationSink.put
  - testApplicationSink.createLargeObjects (9.46%)
  - Thread.sleep (1.39%)
    - Thread.sleep (100%)
  - testApplicationSink.put (0.42%)

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

SHARE in Orlando 2015

IBM Monitoring and Diagnostic Tools for Java

- **Memory Analyzer**
  - Analyze heap dumps to identify application memory leaks and optimize usage
  - Extensions provides additional capabilities for IBM products (WebSphere, CICS-TG)
- **Garbage Collector and Memory Visualizer (GCMV)**
  - Analyze Java verbose GC logs, providing insight into application behaviour
  - Visualize a variety of GC data and Java heap statistics over time
  - Heuristic-based recommendations to help you tune GC performance
- **Health Center**
  - Proactive diagnostic tool
  - Real-time monitoring and profiling in an Eclipse-based GUI
    - Method profiling, lock analysis, garbage collection, CPU usage, heap and native memory usage, thread activities and deadlock detection, class loading, object allocations, file I/O, environment settings
- **Installing the tools is easier than ever before**
  - Available from Eclipse Marketplace, Liberty Repository, IBM Support Assistant
  - https://www.ibm.com/developerworks/java/jdk/tools/

Summary

- zEC12 and Java7R1 performance
  - zEDC, SMC-R, DAA, LP
- IBM z13 is the fastest Java platform on the planet
- Java 8 delivers significant performance improvement
  - Exploitation of z13 capabilities (SMT, SIMD) boosts application throughput
  - Exploitation of CPACF enables massive speedup in cryptographic processing
  - Rampup and throughput improvements thanks to better JIT compiler heuristics
- Monitoring and diagnostic tools
- **Follow us on Twitter @JavaOnZ**

Thank You!

- Please complete your session evaluations!

Session 17651:
IBM Java JVM Tuning For Maximum Performance

www.share.org/Orlando-Eval

Iris Baron

Email: ibaron@ca.ibm.com

Tomorrow ***Thursday, August 13, 2015: 08:30 AM - 09:30 AM,*** *Dolphin, Asia 3*
Session 17635: IBM Java 8 and z13 - Hardware and Software Co-Design at Its Finest

Important references

- z/OS Java web site
  - http://www.ibm.com/systems/z/os/zos/tools/java/
- IBM® SDK, Java™ Technology Edition, Version 8 - Performance
  - https://www-01.ibm.com/support/knowledgecenter/SSYKE2_8.0.0/com.ibm.java.zos.80.doc/performance.html
- 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