Java on z13 – A Performance Update

Linux on z Systems Live Virtual Class – 01 July 2015

Marc Beyerle (marc.beyerle@de.ibm.com)

System z Specialist, Senior Java Performance Engineer

Much of this material was borrowed from Marcel Mitran and team – thanks, Marcel!

IBM **Client Center** - Systems and Software, IBM Germany Lab

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States, other countries, or both.

Not all common law marks used by IBM are listed on this page. Failure of a mark to appear does not mean that IBM does not use the mark nor does it mean that the product is not actively marketed or is not significant within its relevant market.

Those trademarks followed by ® are registered trademarks of IBM in the United States; all others are trademarks or common law marks of IBM in the United States.

For a more complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:

*BladeCenter®, CICS®, DataPower®, DB2®, e business(logo)®, ESCON, eServer, FICON®, IBM®, IBM (logo)®, IMS, MVS, OS/390®, POWER6®, POWER6+, POWER7®, Power Architecture®, PowerVM®, PureFlex, PureSystems, S/390®, ServerProven®, Sysplex Timer®, System p®, System p5, System x®, System z®, System z9®, System z10®, WebSphere®, X-Architecture®, z13™, z Systems™, z9®, z10, z/Architecture®, z/OS®, z/VM®, z/VSE®, zEnterprise®, zSeries®

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

*** All other products may be trademarks or registered trademarks of their respective companies.**

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured Sync new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained Sync the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Agenda

- **Motivation**
- **Simultaneous Multi-Threading (SMT)**
- **Single Instruction Multiple Data (SIMD)**
- **Other Java on z13 improvements**

Motivation

- **Question: *Why should you care* about (a) Java® on the mainframe in general and (b) Java performance in particular?**

- **Answer to (a) is rather easy: Java is the *de facto standard* for new application development projects, also on IBM® z Systems™**
  - Java does not necessarily mean WebSphere® Application Server – there is a large amount customers who currently run Java projects in IMS™, CICS®, Batch, etc.

- **The answer to (b) has many aspects and one of the most important ones is that historically, people are very *performance / resource consumption sensitive* on IBM z Systems**
  - On the z platform, we have an enormous portfolio of tooling around measuring resource consumption and / or performance (RMF™, SMF, *Hardware Instrumentation Services* (HIS), *Application Performance Analyzer* (APA), z/VM® Performance Toolkit, Tivoli® product suites, etc.)

Marc Beyerle (marc.beyerle@de.ibm.com) – Java on z13 – A Performance Update

Agenda

- Motivation
- **Simultaneous Multi-Threading (SMT)**
- Single Instruction Multiple Data (SIMD)
- Other Java on z13 improvements

Simultaneous Multi-Threading

- **Implementing *Simultaneous Multi-Threading* (SMT) in the new IBM z13™ is a big step for Java, since many Java workloads benefit greatly from SMT**

- **However, *not all* Java workloads will benefit from SMT, therefore one *cannot* make a general statement like *"…you will always get x% improvement…"***

Workloads that *will* benefit:

- **Generally speaking: all *lightweight* applications that require lots of threads**
- ***Lightweight* in this context mostly means *small cache footprint***
- **Example: lightweight *web applications* (front-end-only type of web appl.)**
- **Other example: lightweight *transaction processing* (transaction just updating 1 row in the DB)**

Workloads that *will not* benefit so much:

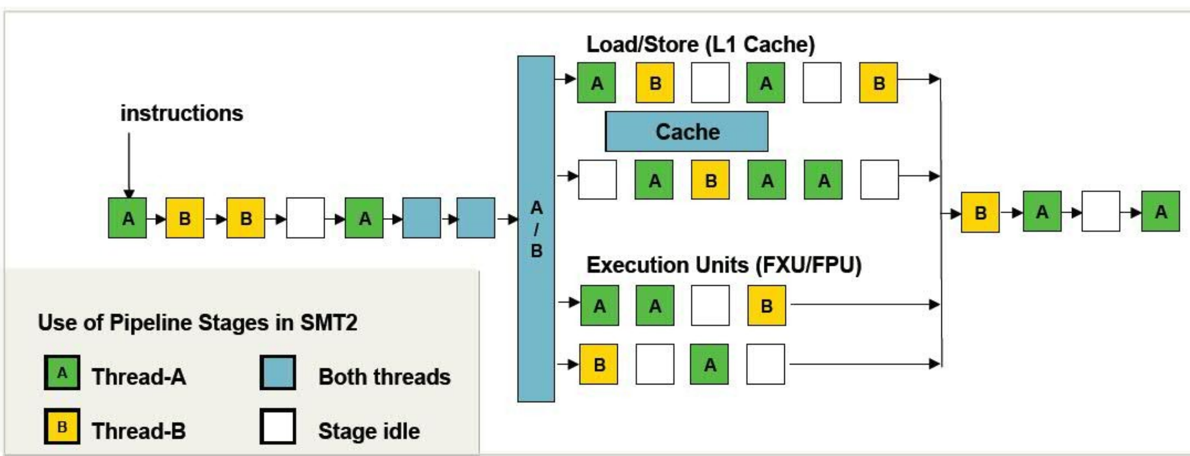
- **Generally speaking: all *heavyweight* applications, regardless of the number of threads**
- **Example: *payroll applications* (more or less single-threaded, lots of DB accesses)**
- **Other example: overnight *Java batch runs* (same characteristics as payroll)**

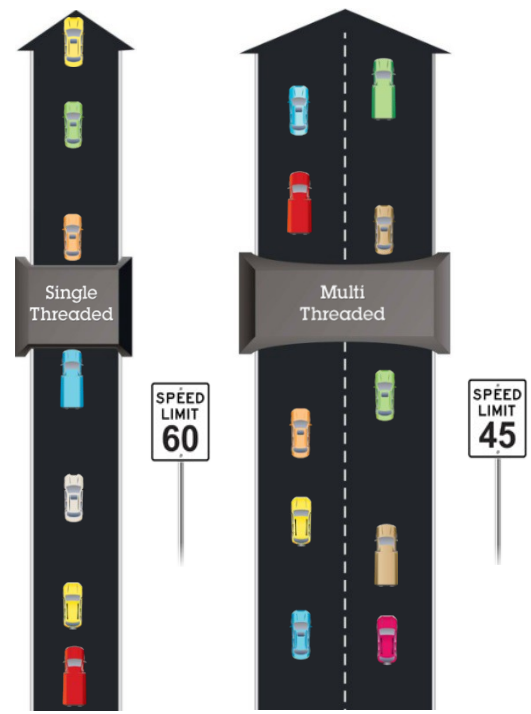
Simultaneous Multi-Threading, *cont.*

- **Double the number of hardware threads per core**
  - Independent threads can be more effective utilizing pipeline

- **Threads share resources – may impact single thread perf.**
  - Pipeline (eg. physical registers, fxu, fpu, lsu, etc.)
  - Cache

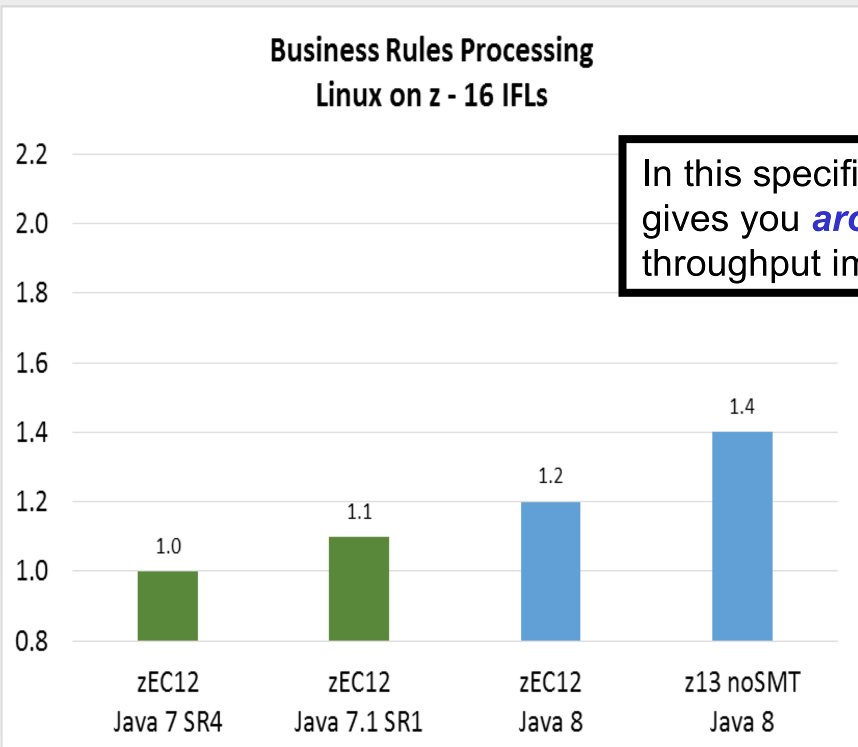
- **Throughput improvement is workload dependent**



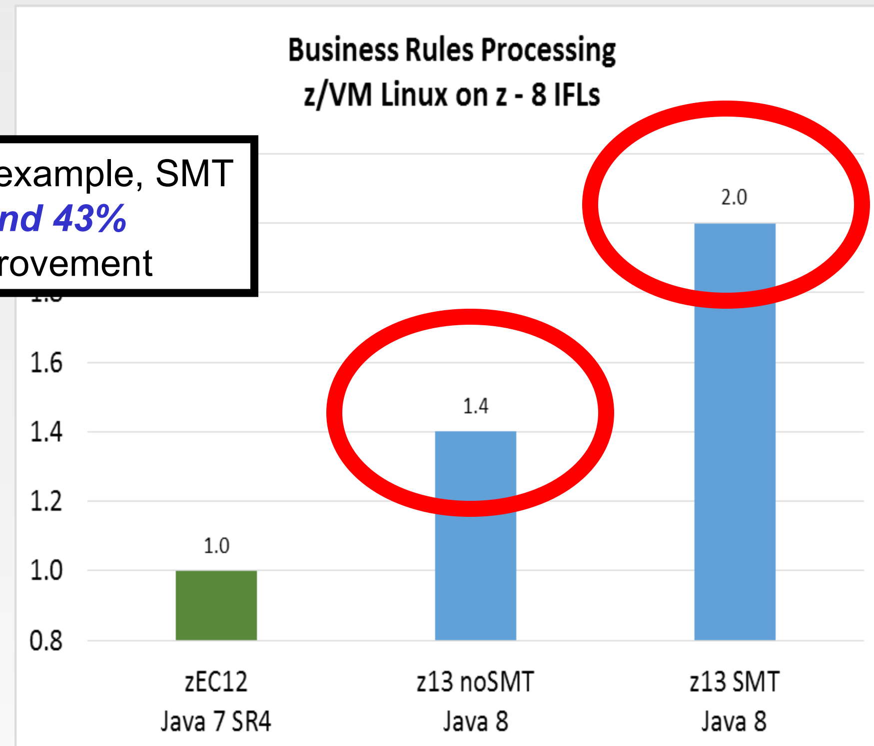


Two lanes handle more traffic overall

IBM Business Rules Processing with IBM Java 8 and z13

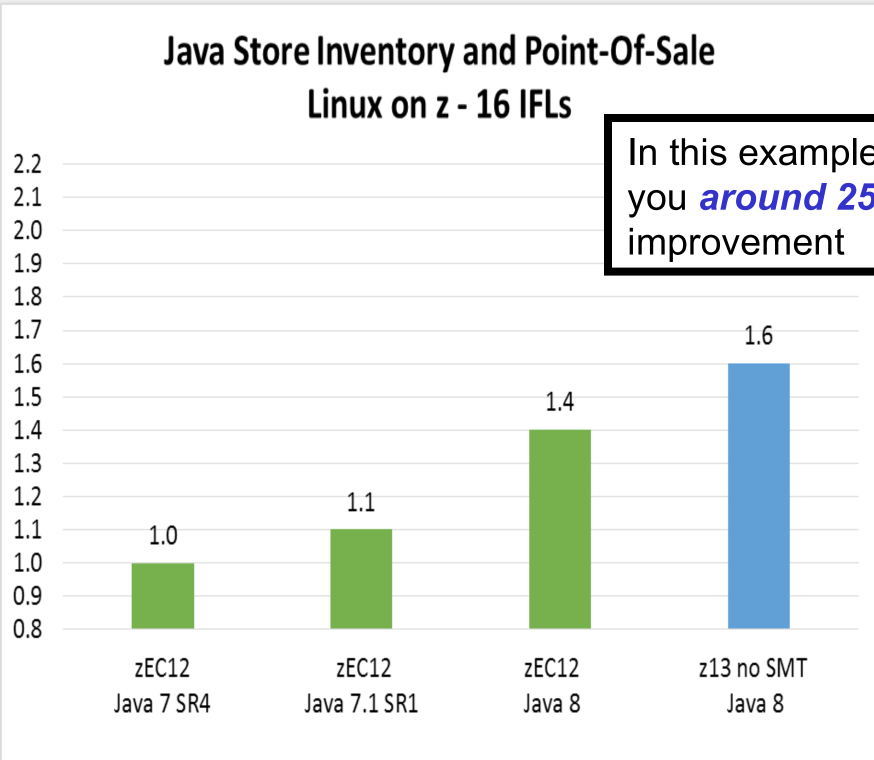


In this specific example, SMT gives you ***around 43%*** throughput improvement

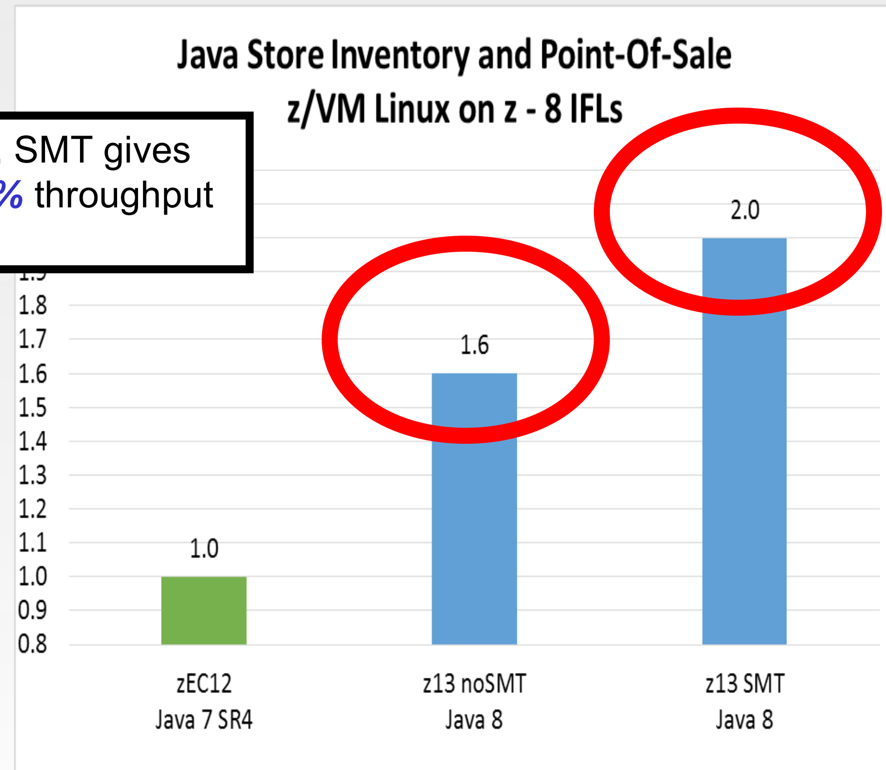


(Controlled measurement environment, results may vary)

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



In this example, SMT gives you ***around 25%*** throughput improvement

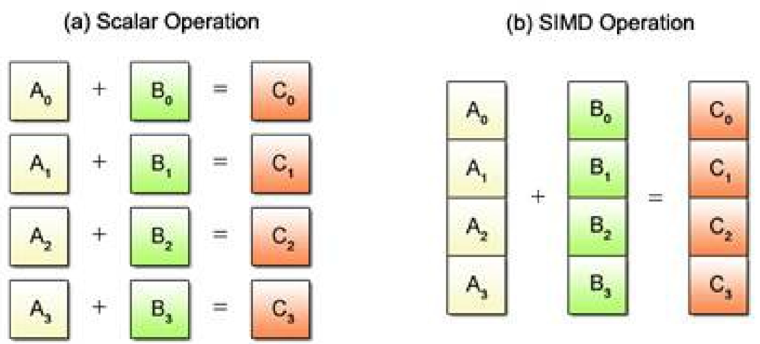


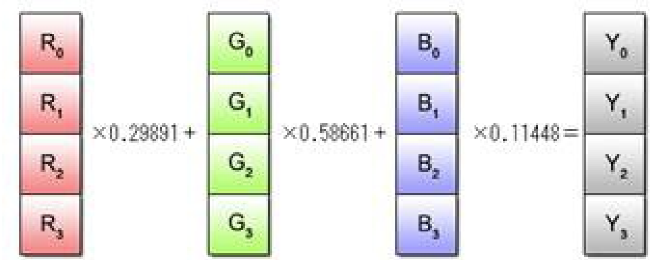
Agenda

- Motivation
- Simultaneous Multi-Threading (SMT)
- **Single Instruction Multiple Data (SIMD)**
- Other Java on z13 improvements

Single Instruction Multiple Data

- **Quick recap – the following pictures illustrate the principle of *Single Instruction Multiple Data* (SIMD):**





- **When I first heard that z13 was going to implement SIMD, I didn't see the value for Java *business applications* in it, since I only knew about SIMD advantages in scientific applications like *image processing*, for example – but I was wrong…**

Single Instruction Multiple Data, *cont.*

- **Basically, SIMD is very well suited whenever one has to process *large arrays of data of the same type*, which also means large arrays of character data – also known as *strings***

- **Character array:**

H	e	l	l	o	,		W	o	r	l	d	!

- **Situations when processing on character arrays occurs:**
  - String ***comparison***
  - Single character / substring ***search***
  - String ***conversion***

- **All of the above mentioned operations are *heavily used* by Java application programmers**

Marc Beyerle (marc.beyerle@de.ibm.com) – Java on z13 – A Performance Update

String, Character Conversion and Loop Acceleration with SIMD

IBM z13 running Java 8
Single Instruction Multiple Data (SIMD) vector engine exploitation

java.lang.String exploitation
- compareTo
- compareToIgnoreCase
- contains
- contentEquals
- equals
- indexOf
- lastIndexOf
- regionMatches
- toLowerCase
- toUpperCase
- getBytes

java.util.Arrays
- equals (primitive types)

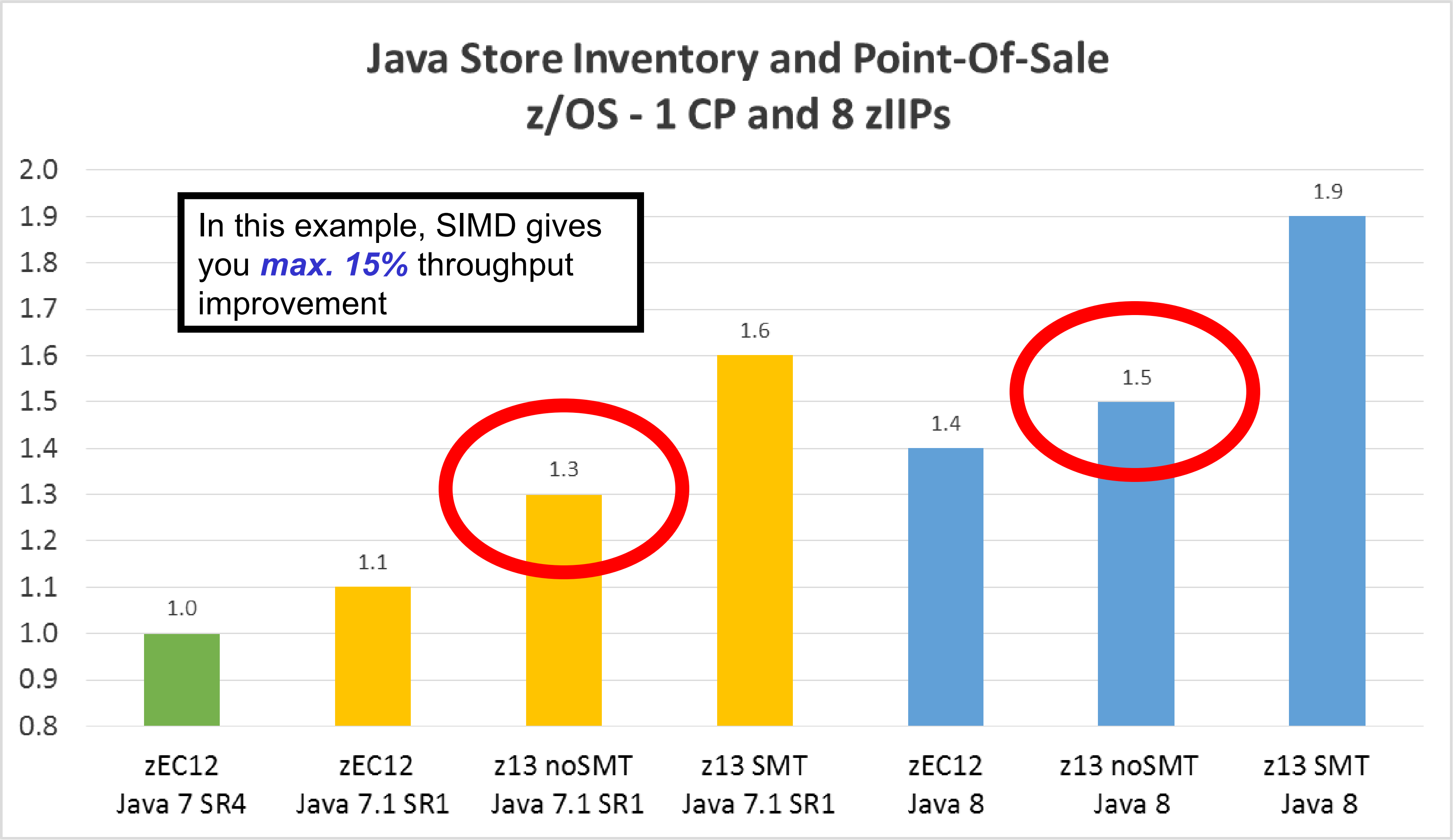
String encoding converters
For ISO8859-1, ASCII, UTF8, and UTF16
- encode (char2byte)
- decode (byte2har)

Auto-SIMD
- Simple loops
  (eg. matrix multiplication)

Primitive operations are between 1.6x and 60x faster with SIMD

(Controlled measurement environment, results may vary)

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



(Controlled measurement environment, results may vary)

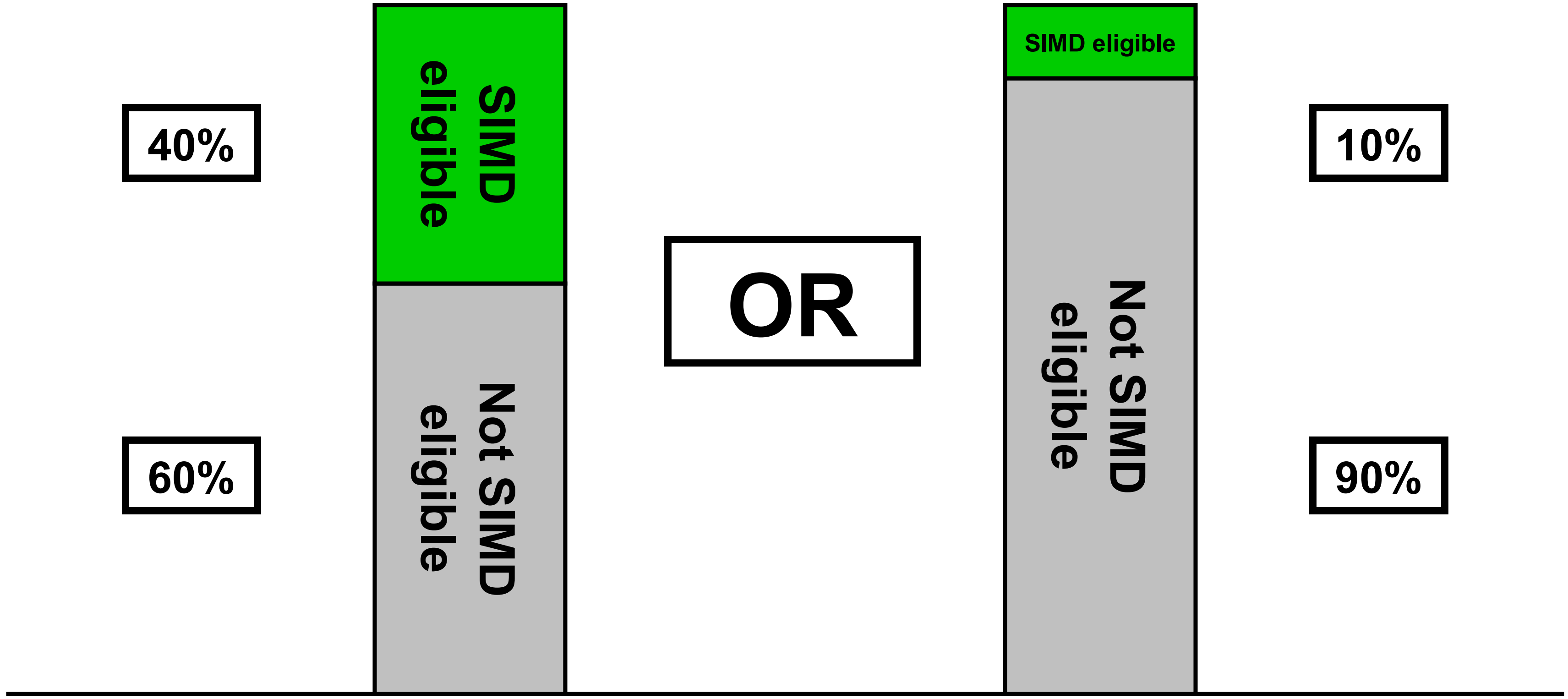
Comment

- ***Important comment*** **– mostly for the people who cannot attend the live presentation, but only have the slide deck**

- **In the previous performance chart, I said max. x%** ***by intention*****. This is because the change from Java V7.1 to Java V8 includes both** ***general*** **IBM** ***Java Virtual Machine*** **(JVM™) improvements and also** ***SIMD*** **improvements.**

- **Or in other words, SIMD can –** ***at a maximum / as an upper bound*** **– be responsible for x% improvement, since the general JVM improvements also contribute to the x% value. Very probably, SIMD is responsible for less than x%, but definitely not more than x%.**

SIMD exploitation is heavily based on workload characteristics



The big question is: How much "SIMD eligible" code contains your Java workload?

- String operations (compares, searches, etc.), conversions
- Array compares, Auto-SIMD

Marc Beyerle (marc.beyerle@de.ibm.com)

Method for estimating the percentage of SIMD eligible code in your workload

- **This method is suitable for measuring the percentage of SIMD-eligible instructions *at the Java language level***

- **Disclaimer #1: This method is *not 100% accurate*, since it is impossible to measure the percentage of Auto-SIMD instructions at the Java language level. Measuring the percentage of Auto-SIMD instructions would imply instrumenting the *Just-In-Time* (JIT) compiler.**

- **This method uses Java *bytecode instrumentation*, which adds some additional cost compared to the execution of un-instrumented Java workload**

- **Disclaimer #2: This method is not 100% accurate, since bytecode instrumentation itself *slightly skews the measurement***

Method for estimating the percentage of SIMD eligible code in your workload, *cont.*

- ***Question***: **What is this method based on?**

- ***Answer***: **Tool being used for this method is *Jinsight*, a Java profiling agent (or Java *profiler*) for IBM z Systems**

- ***Profiler*** **in this context means the tool *hooks into* the JVM and gets notified whenever a particular event occurs**
  – This event listening mechanism is based on ***Java bytecode instrumentation***

- **Jinsight has 2 execution modes:**
  – ***Profiling mode*** (recording sequence of events, extremely intrusive)
  – ***CPU mode*** (recording duration of *transactions*, slightly intrusive)

- **In order to estimate the SIMD eligible code percentage, Jinsight's *CPU mode* has to be used**
  – Tracing an entire application can only be realized with CPU mode, since profiling mode would produce terabytes of trace data

Marc Beyerle (marc.beyerle@de.ibm.com) – Java on z13 – A Performance Update

Method for estimating the percentage of SIMD eligible code in your workload, *cont.*

- ***Question***: **What else do you need?**

- ***Answer***: **Besides the Jinsight profiling agent, you also need a *configuration file* for CPU mode – *tailored to your application* – that turns transaction time recording on for the SIMD eligible Java instructions**

- **If you put the application *under typical load* with Jinsight CPU mode turned on, you will get the amount of CPU time spent for SIMD eligible instructions**
  - ***Important***: If you put load on the application that does not correspond to real life usage, your estimation ***will not reflect reality***

- **Divide the time spent for SIMD eligible instructions by the total CPU time for the Java address space and you will get an idea whether your application will benefit from SIMD or not**

Marc Beyerle (marc.beyerle@de.ibm.com) – Java on z13 – A Performance Update

Method for estimating the percentage of SIMD eligible code in your workload, *cont.*

- **I did this exercise for a *very small Java web application* (based on JavaServer Faces), just to see if it works. The following summarizes my observations.**

- **The configuration file has to be *adapted for every application***
  - Due to the way CPU mode transactions work
  - Web applications: define tx start when the request enters the application server
  - CICS / IMS: define tx start when the Java application gets called

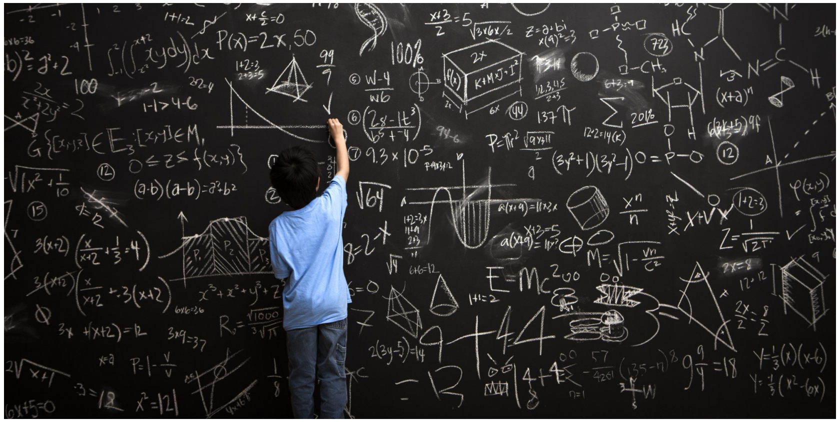
- **SIMD eligible instructions sometimes *call themselves*, so you should only count the *outermost / first-level* ones**
  - Otherwise, you will end up with wrong assumptions

Method for estimating the percentage of SIMD eligible code in your workload, *cont.*

- **In my example, there were no character conversion operations or array compares, just *string operations***
  - This is just an example and does not necessarily represent all Java web applications

- **Nevertheless, the percentage of CPU time spent for SIMD eligible instructions was *16.5%* of the total CPU time consumed by the application**

- **Now lets do some math…**



Marc Beyerle (marc.beyerle@de.ibm.com) – Java on z13 – A Performance Update

Agenda

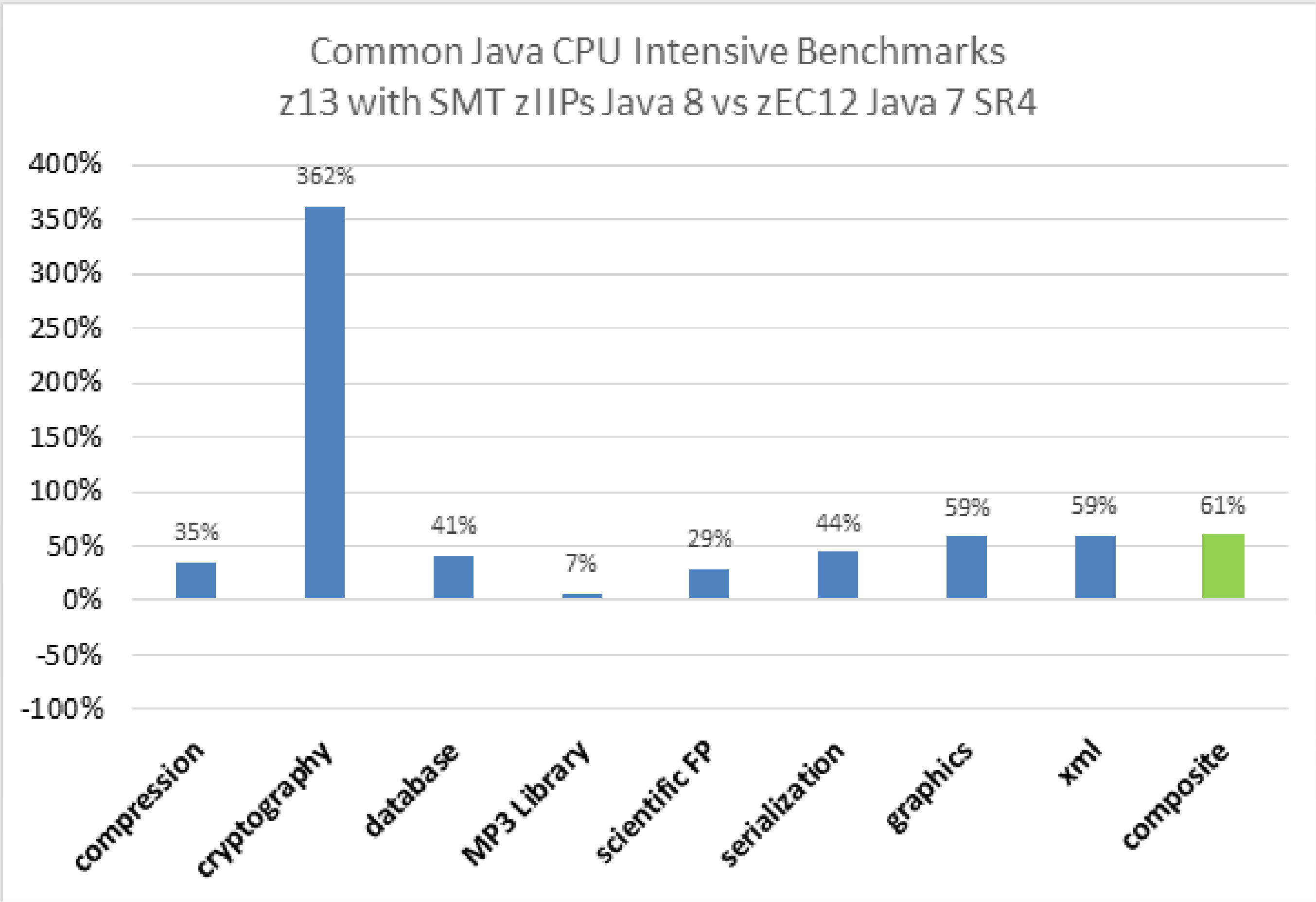
- Motivation
- Simultaneous Multi-Threading (SMT)
- Single Instruction Multiple Data (SIMD)
- **Other Java on z13 improvements**

Marc Beyerle (marc.beyerle@de.ibm.com)

IBM SDK for z/OS, Java Tech. Edition, Version 8 (IBM Java 8)

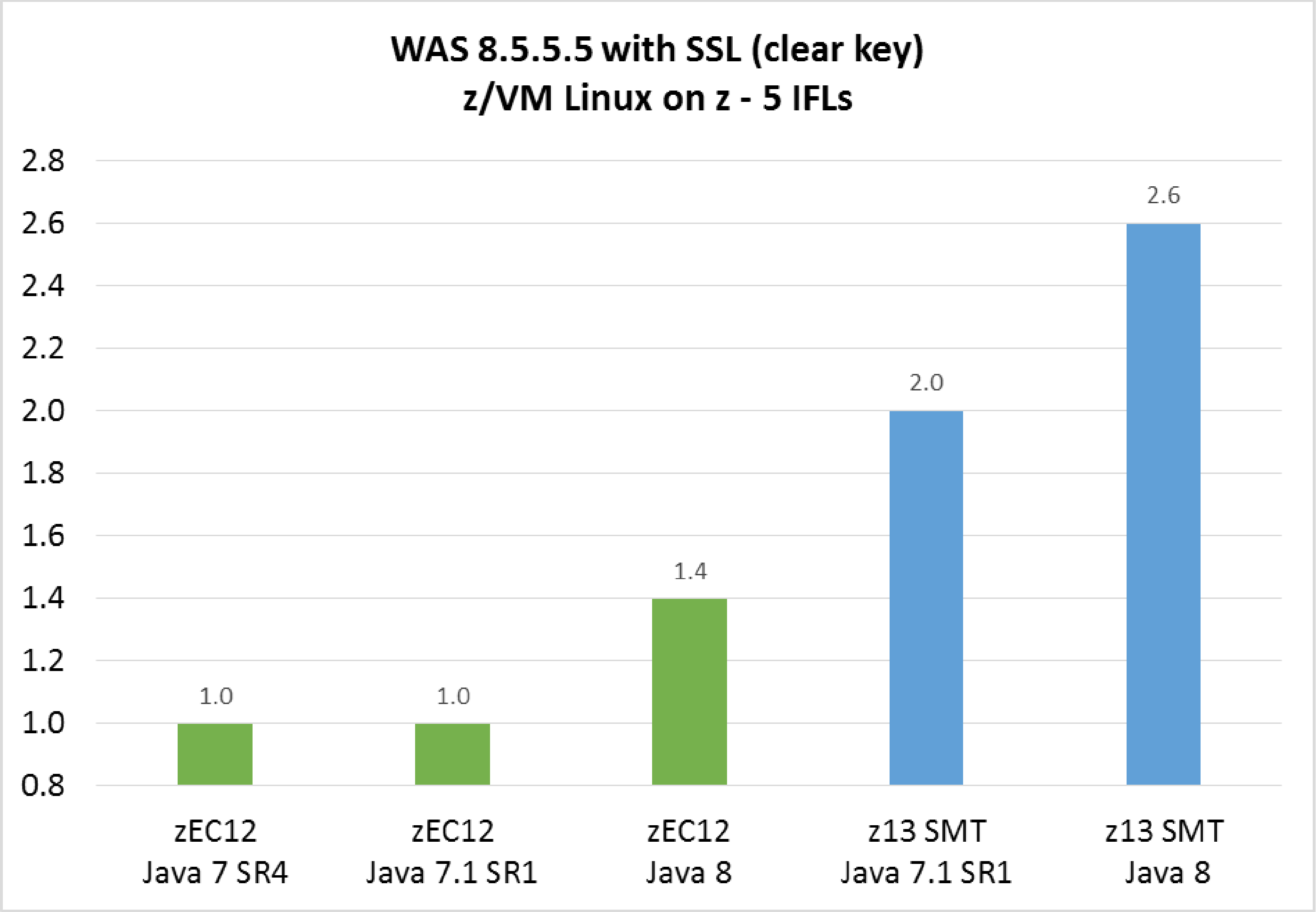
- **New Java8 Language Features**
  - Lambdas, virtual extension methods
- **IBM z13 exploitation**
  - Vector exploitation and other new instructions
  - Instruction scheduling
- **General throughput improvements**
  - Up-to 17% better application throughput
  - Significant improvements to ORB
- **Improved 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**

IBM Java 8: CPU-Intensive Benchmark



(Controlled measurement environment, results may vary)

WAS/Liberty 8.5.5.5 – SSL-Enabled DayTrader 3.0



(Controlled measurement environment, results may vary)

Marc Beyerle (marc.beyerle@de.ibm.com) – Java on z13 – A Performance Update

Summary

- **Generally speaking, both Java V8 and IBM z13 improve performance of Java applications on the IBM mainframe *significantly***
  - The exact percentage of improvement depends heavily on the characteristics of the application (instruction mix, crypto / no crypto, cache intensity, etc.)

- **You have to *upgrade to Java V8* in order to make the most out of z13**
  - WebSphere Liberty Profile is already supported, traditional WebSphere Application Server and CICS / IMS will follow

Thank you

Resources

- **IBM Client Center – Systems and Software, IBM Germany Lab**
  - Part of the IBM Development Lab in Boeblingen, Germany
  - External homepage: http://www.ibm.com/de/entwicklung/clientcenter/index_en.html
  - IBM Intranet: http://clientcenter.de.ibm.com
  - Email: clientcenter@de.ibm.com

- **IBM developer kits: http://www.ibm.com/developerworks/java/jdk**

- **Java on z/OS: http://www.ibm.com/systems/z/os/zos/tools/java**

- **Java Diagnostics Guide: http://www.ibm.com/developerworks/java/jdk/diagnosis**

- **JinsightLive for IBM System z homepage: http://www.ibm.com/systems/z/os/zos/features/unix/tools/jinsightlive.html**