

# IBM z13 and Java 8: Taking Java to New Heights on IBM z Systems

Marcel Mitran – IBM DE, CTO z Systems Software Performance





## Trademarks, Copyrights, Disclaimers

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at "[Copyright and trademark information](http://www.ibm.com/legal/copytrade.shtml)" at <http://www.ibm.com/legal/copytrade.shtml>

Other company, product, or service names may be trademarks or service marks of others.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.

© Copyright International Business Machines Corporation 2014. All rights reserved.

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.




# Java on z Systems? 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



# Reasons to Love IBM Java and WAS on z Systems

## HCSC – 14.5 million health insurance members

WebSphere on z/OS has been selected at HCSC as a preferred platform to support development and deployment of mission-critical Java applications for the following reasons:

### Co-location:

WAS on IBM z minimizes physical tiers

3-4x improvement for one of HCSC’s largest WAS applications when moving from distributed to z/OS

### High Volume Transaction Rates:

Could not meet business needs with distributed

### Qualities of Service

Horizontal scaling

Continuous availability and fail-over



### IBM JVM Performance Dividends

30% improvement with Java601

10% improvement with Java7.1

[www.slideshare.net/elenan3403/reasons-to-love-ibm-java-and-web-sphere-application-server-on-z-system](http://www.slideshare.net/elenan3403/reasons-to-love-ibm-java-and-web-sphere-application-server-on-z-system)

## How Java became the NextGen Application of Choice for Fiducia

Fiducia IT AG - Full service provider for 750 Banks with about 10,550 Points of sales

- Java support in **IMS MPRs and BMPs**
  - Support of Java SE (31 bit) for building “Business logic“
  - JVM is resident in the MPR for its hole life
- In MPRs: the main program is COBOL
- Inter Language Communication (ILC) between COBOL and Java allows:
  - COBOL to call Java and Java to call COBOL (cascading)
- Java and COBOL run within the same transaction (UOW) with database support for:
  - DB2 from COBOL (static SQL) and Java (JDBC Type-2)
  - DL1 from COBOL and Java
  - MQ access from Java (future)

IBM

Insight2014

The Conference for Big Data and Analytics

How Java became the NextGen Application of Choice for Fiducia  
IDM-6387A

Pascal Meyer, Executive Architect Fiducia IT AG

28. October 2014

SEIZE THIS  
MOMENT

#ibminsight

© 2014 IBM Corporation



**JZOS** in use for mission critical applications since 2013

- Huge amounts of data is manipulated
- Throughput is very important
- Strict timing window for the business



# Evolving Java as an Optimized Workload on IBM z

*Enable integration of Java-based applications with core z/OS backend database environment for high performance, reliability, availability, security, and lower total cost of ownership*

## Portable and consumable

- First-class IBM Java SDK for z/OS and Linux on z Systems
- Providing seamless portability across platforms

## Pervasive and integrated across the IBM z eco-system

- Java business logic runs with all IBM z middleware (IMS, CICS, WAS etc)
- Inter-operability with legacy batch and OLTP assets

## Deep z Systems exploitation

- SDK extensions enabled IBM z QoS for full integration with z/OS
- zAAP/zIIP/IFL specialty engines provide low-cost Java capacity

## Performance

- A decade of hardware/software innovation and optimization
- Industry leading performance with IBM J9 Virtual Machine
- Enabling tight data locality for high-performance and simplified systems





# Java Execution Environments and Interoperability

Capitalize on pre-existing assets, artifacts, processes, core competencies, platform strengths

## IBM Java Execution Offerings

- Transactional/Interactive
  - WebSphere for z/OS (WAS z/OS)
  - WebSphere Process Server for z/OS (WPS)
  - Java CICS
  - IMS Java
  - DB2 Stored Procedures
- Batch oriented
  - WebSphere Compute Grid (WAS-CG)
    - WAS/JEE runtime extensions
  - IMS Java Batch regions (JMP)
  - JZOS component of z/OS SDK
    - JES/JSE-based environment
  - z/OS V1R13 Java/COBOL Batch Runtime Env.\*
    - JES/JSE-based, designed to inter-op with DB2 while maintaining transaction integrity

## Open Source or non-IBM vendor Application Server and Frameworks

- Tomcat, JBoss
- iBatis, Hibernate, Spring
- Ant

## COBOL/Native Interoperability

- COBOL Invoke maps to JNI
- RDz and JZOS\*\* have tooling to map COBOL copy books to Java classes
- JCICS
- IMS Java, JMP/JBP
- WAS CG, WOLA
- etc

\* See <http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?subtype=ca&infotype=an&supplier=897&letternum=ENUS211-252>



# 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

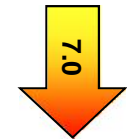
- Support for dynamic languages
- Improve ease of use for SWING
- New IO APIs (NIO2)
- Java persistence API
- JMX 2.x and WS connection for JMX agents
- Language Changes

### 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 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, FF

### 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

### 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 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 7% 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**



## Java 8 -- Lambdas

New syntax to allow for concise and expressive code snippets

- can be thought of as 'anonymous methods'

```
Collections.sort(people, new Comparator<Person>() {  
    public int compare(Person x, Person y) {  
        return x.getLastName().compareTo(y.getLastName());  
    }  
});
```



```
people.sort(Comparing(Person::getLastName));
```

## Java 8 – Lambdas for Streaming Operations

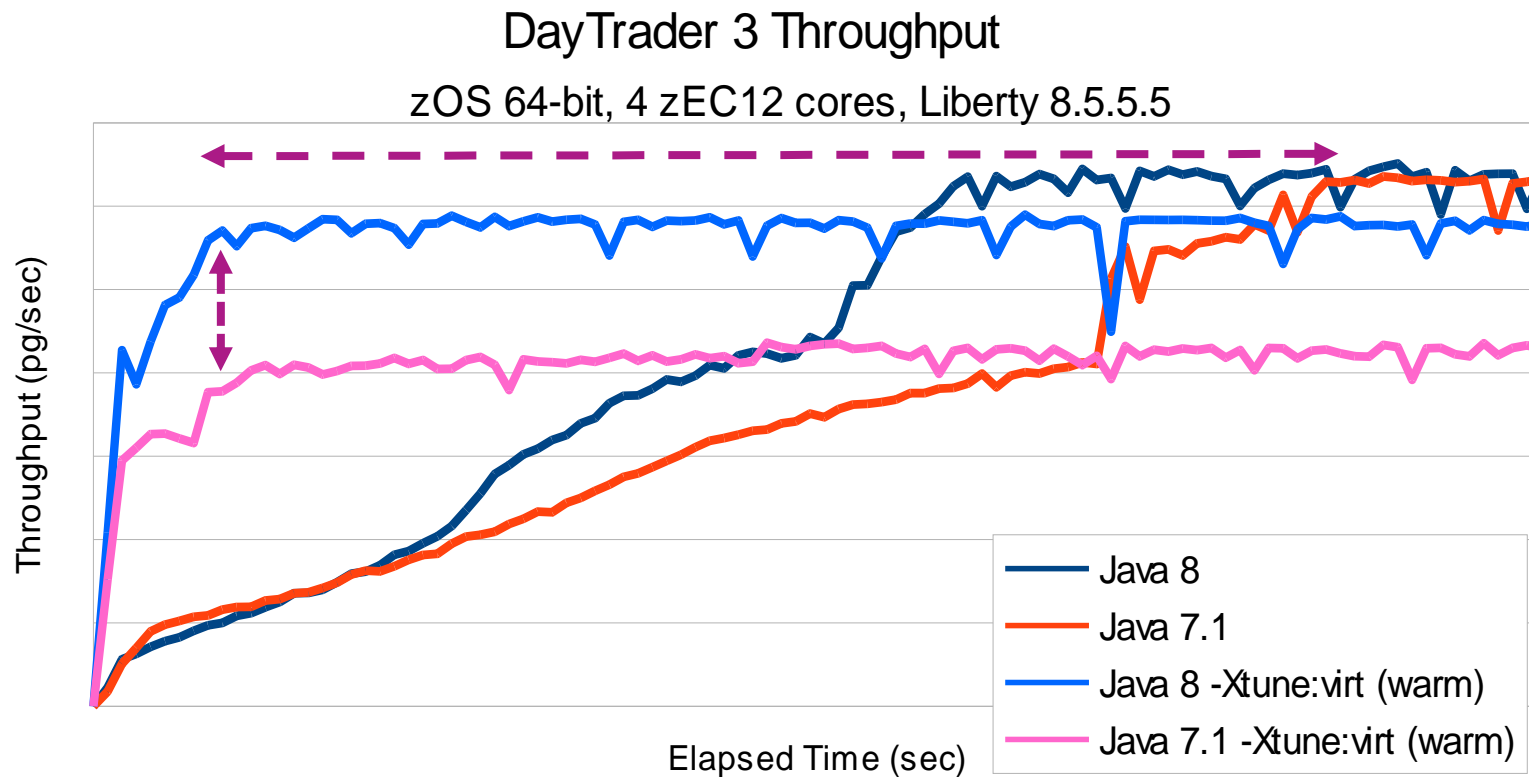
Lambdas can be pipelined to enable data stream operations

- Intermediate operations on streams produce new streams
- Terminal operations produce results

```
int totalWeight = widgets.stream()  
    .filter(w->w.getColor() == RED)  
    .mapToInt(w->w.getWeight())  
    .SUM();
```

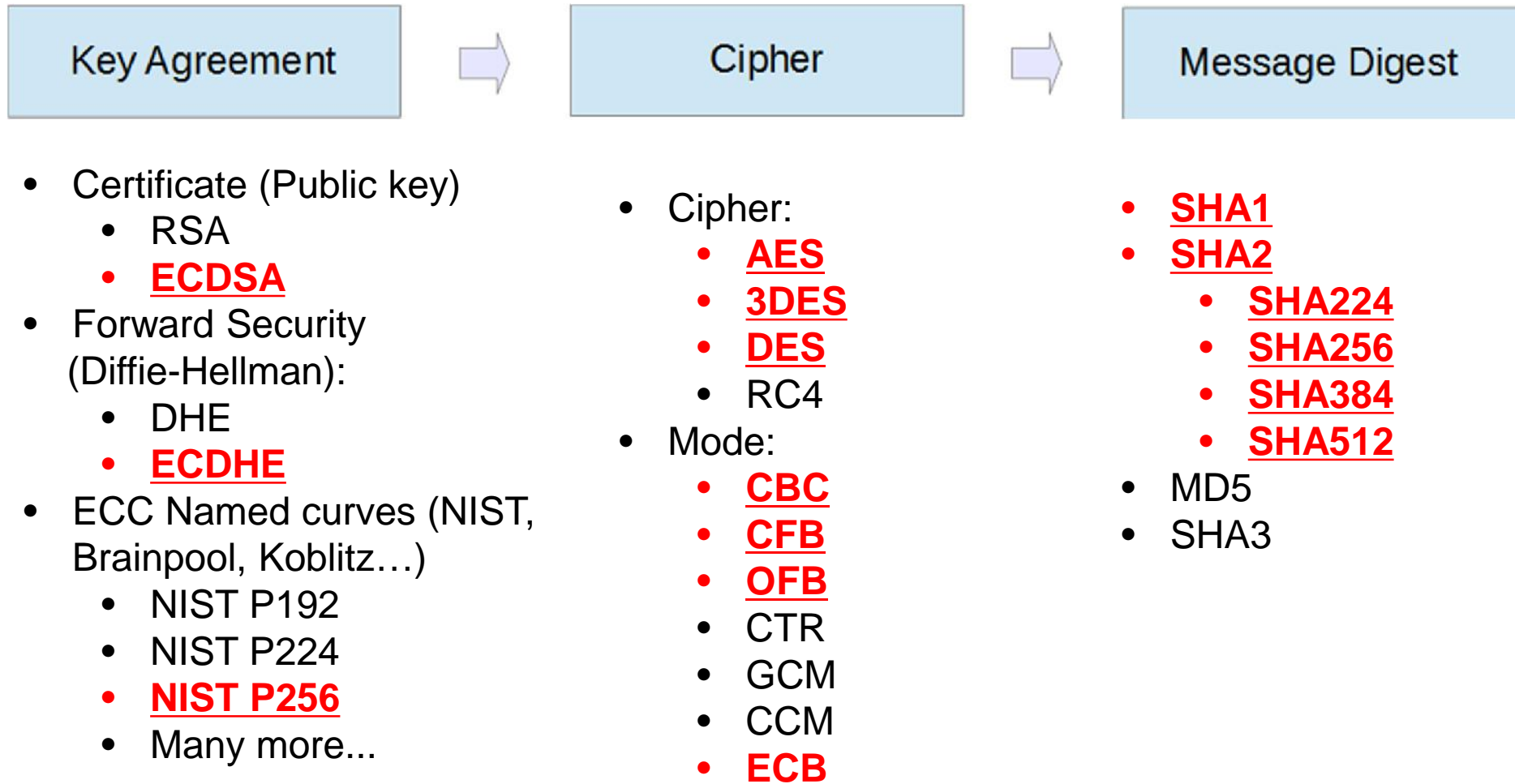
Enables exploitation of parallelism and supports multi-core programming

# z/OS 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%

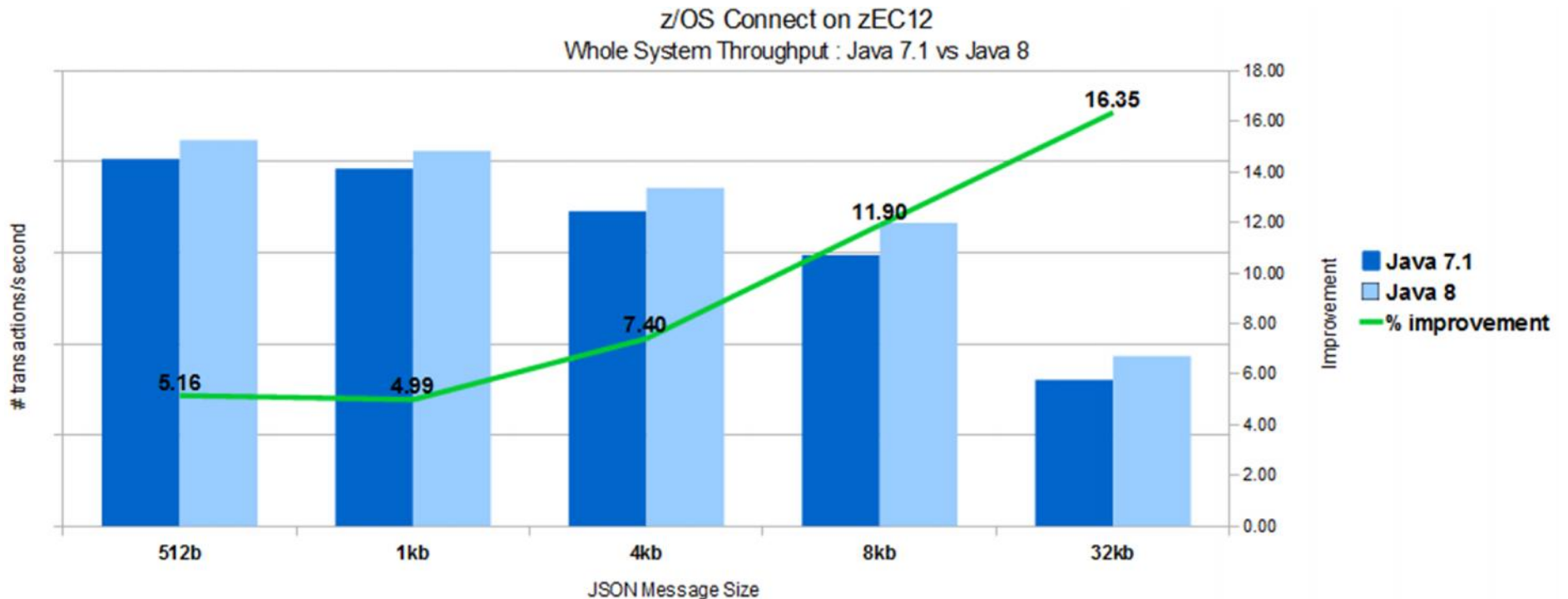
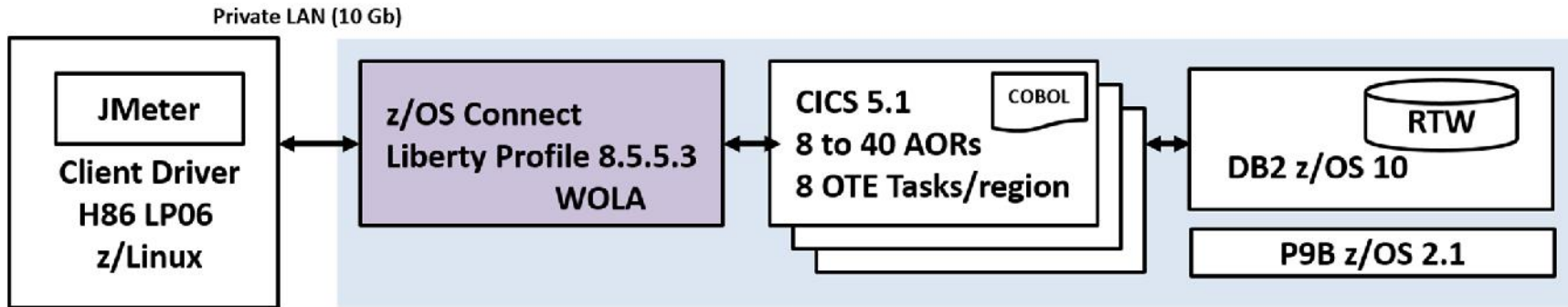
# Java 8 Summary of Crypto Acceleration







# Mobile on z Systems – z/OS Connect on IBM Java 8



**5-16.4% throughput improvement from IBM Java 8 and IBM zEC12**

# JMX Beans for Precise CPU Monitoring

## New JMX Beans for reporting CPU usage categorized by:

1. JVM System threads (JIT, GC, etc)
2. Application threads
3. Monitoring threads (to be able to excl. monitoring overhead)

## Intended use-cases

- Reporting transaction cpu usage
- Identifying "expensive" transactions
- Reporting JVM overhead over specific intervals
- Foundation for future work on tracking idle behaviour

## New classes

- `com.ibm.lang.management.JVMCpuMonitorMXBean` (Bean to request Data)
  - `getThreadsCpuUsage()`
  - `setThreadCategory() / getThreadCategory()`
- `com.ibm.lang.management.JVMCpuMonitorInfo` (Object with Data)

## Overhead may be visible on some platforms

Option to trade-off more precise GC-time reporting vs. reduced overhead

`-XX:+ReduceCPUMonitorOverhead(default.) / -XX:-ReduceCPUMonitorOverhead`  
(z/OS cannot enable more precise GC-time reporting today)

# JZOS – SMF Logging

## SMF Logging to Record type 121 subtype 1

- JZOS\_JVM\_SMF\_LOGGING environment variable to enable
- Captures JVM runtime information
  - Uptime, number of live threads and GC statistics
- Record is logged during JVM shutdown

## FUTURE function being considered\*\*

- SMF records to include breakdown of Application, JVM system, GC and JIT CPU-time
- Information available on a per-thread basis
- Captured periodically at user-defined intervals

# 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

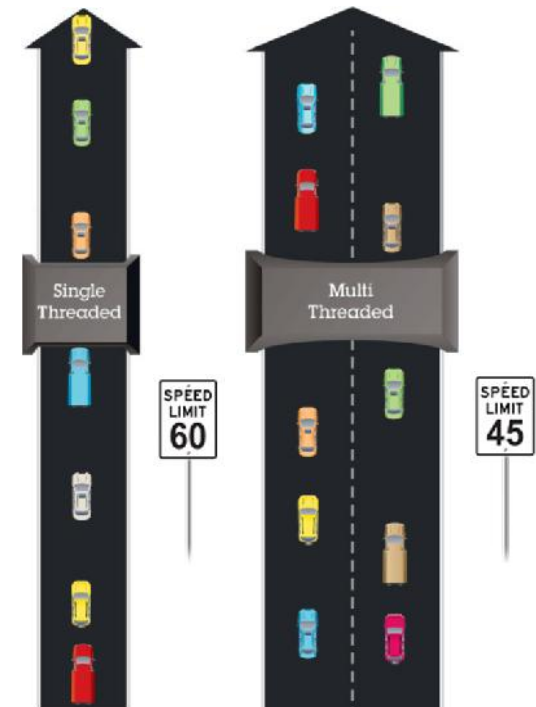
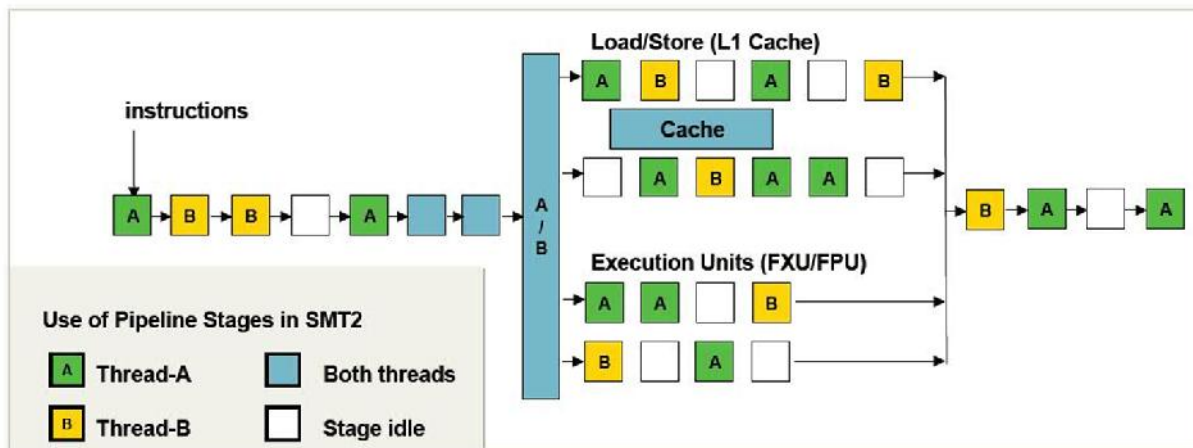


Up to **50%** improvement in throughput for generic applications

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

# IBM z13: SMT – Simultaneous Multi-Threading

- Double the number of hardware threads per core
  - Independent threads can be more effectively utilizing pipeline
- Threads share resources – may impact single thread perf
  - Pipeline (eg. physical registers, fxu, fpu, lsu etc)
  - Cache
- Throughput improvement is workload dependent







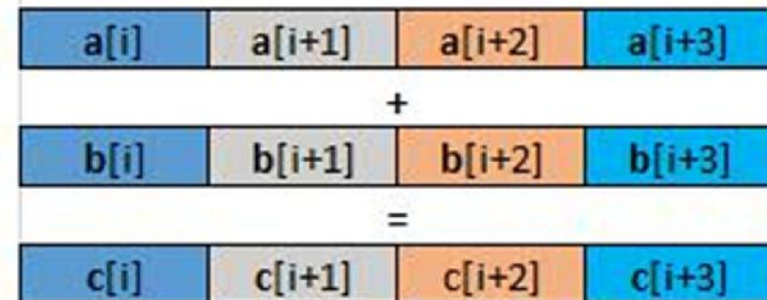
## Vector Processing - Single Instruction Multiple Data (SIMD)

Operate on multiple data-elements (vectors) simultaneously

Can offer dramatic speed-up to data-parallel operations (matrix ops, string processing, etc)

Vector registers are 128-bits wide and can be used to operate concurrently on:

- Two 64-bit floating point values
- Four 32-bit floating point values
- Sixteen 8-bit characters
- Two 64-bit integer values
- etc



**4x less iterations in the SIMD loop**

```
//SISD C example, adding two arrays
for (i=0;i<128;i++)
{
  c[i] = a[i] + b[i];
}
```

```
//SIMD C example, adding two arrays
for (i=0;i<32;i++)
{
  vec_add(c[i*4], a[i*4], b[i*4]);
}
```

## String, Character Conversion and Loop Acceleration with SIMD

### IBM z13 running Java 8 on z/OS 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 (byte2char)

#### Auto-SIMD

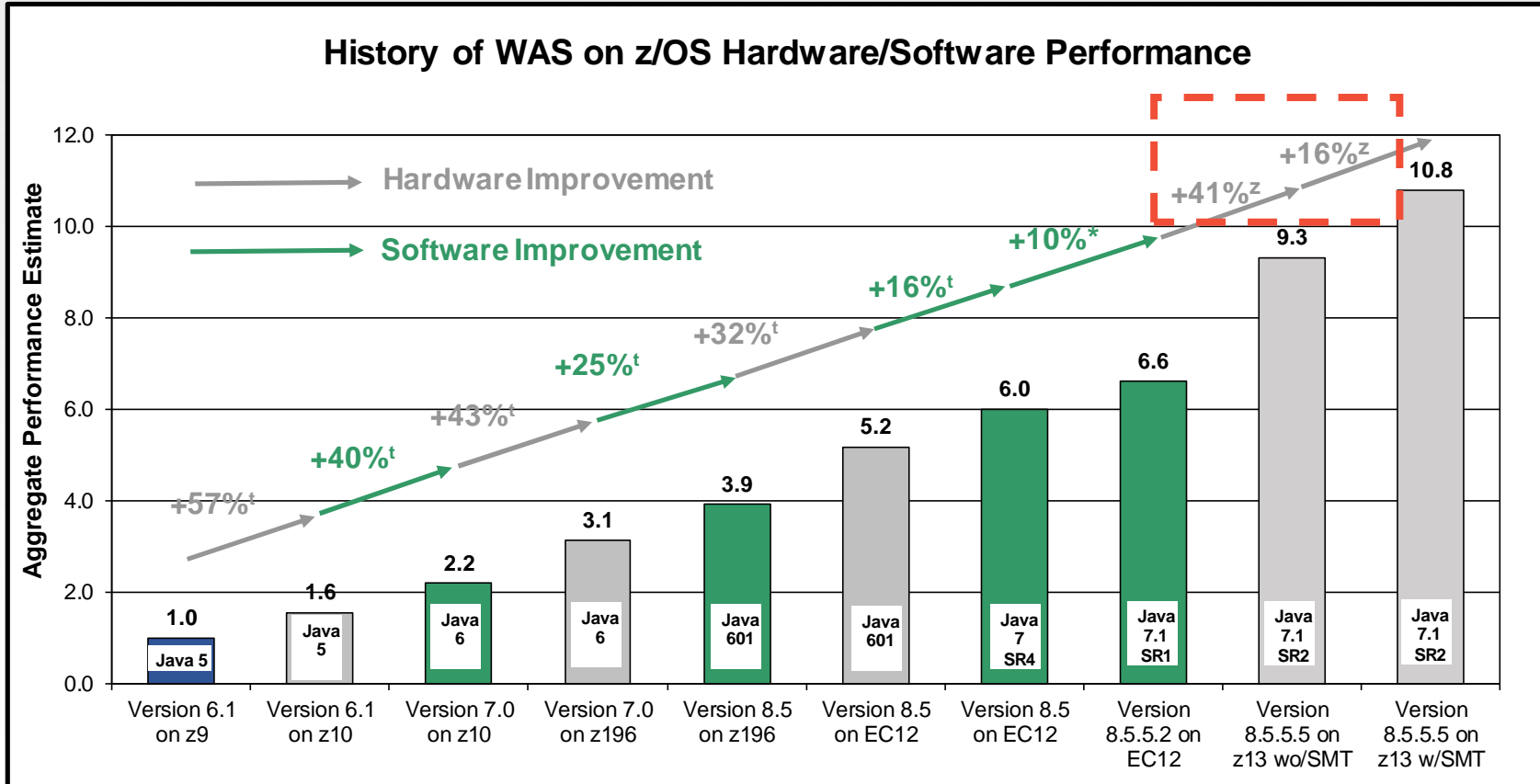
- Simple loops  
(eg. Matrix multiplication)

**Primitive operations are between 1.6x and 60x faster with Java8**



# 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

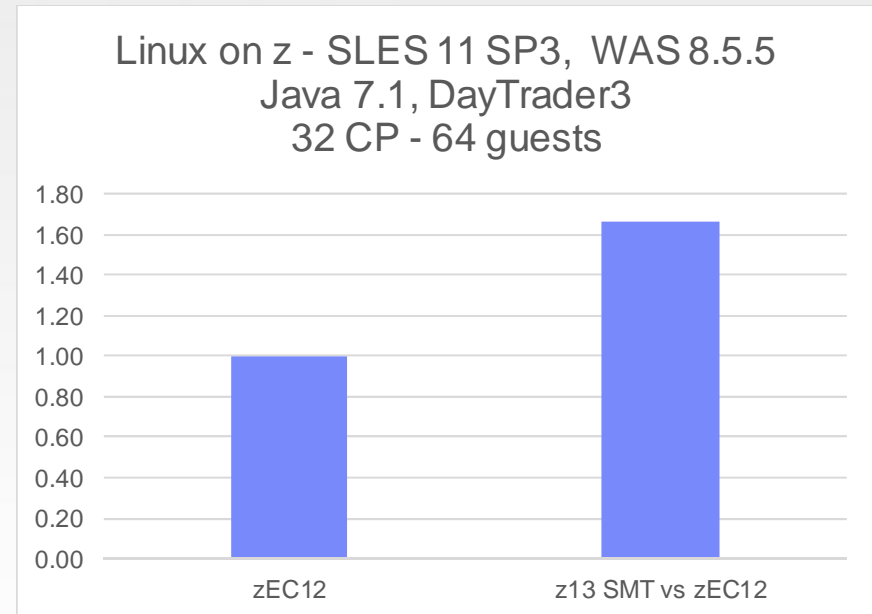
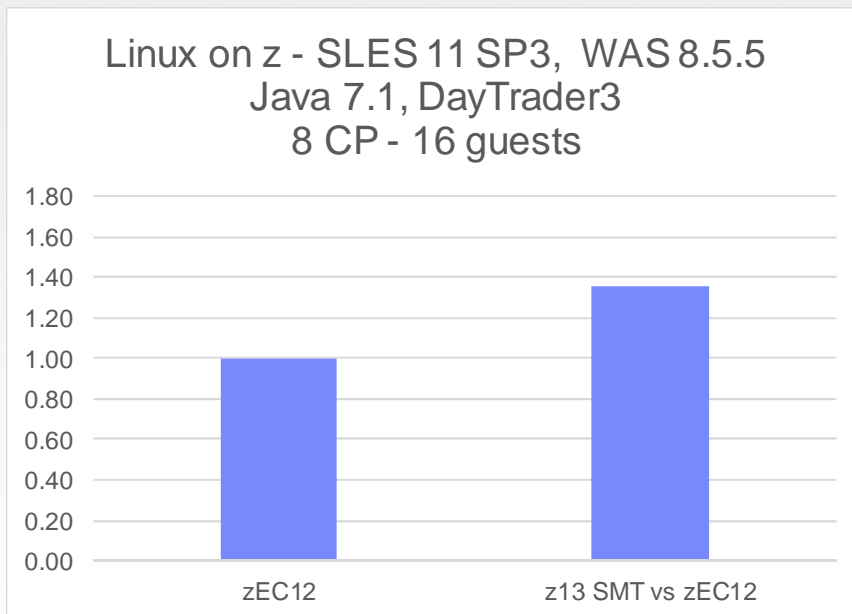


**10.8x aggregate hardware and software improvement comparing  
WAS 6.1 IBM Java5 on z9 to WAS 8.5.5.2 IBM Java7R1 on z13 w/SMT**

z zIIPs DayTrader 3  
\* DayTrader3  
t DayTrader2



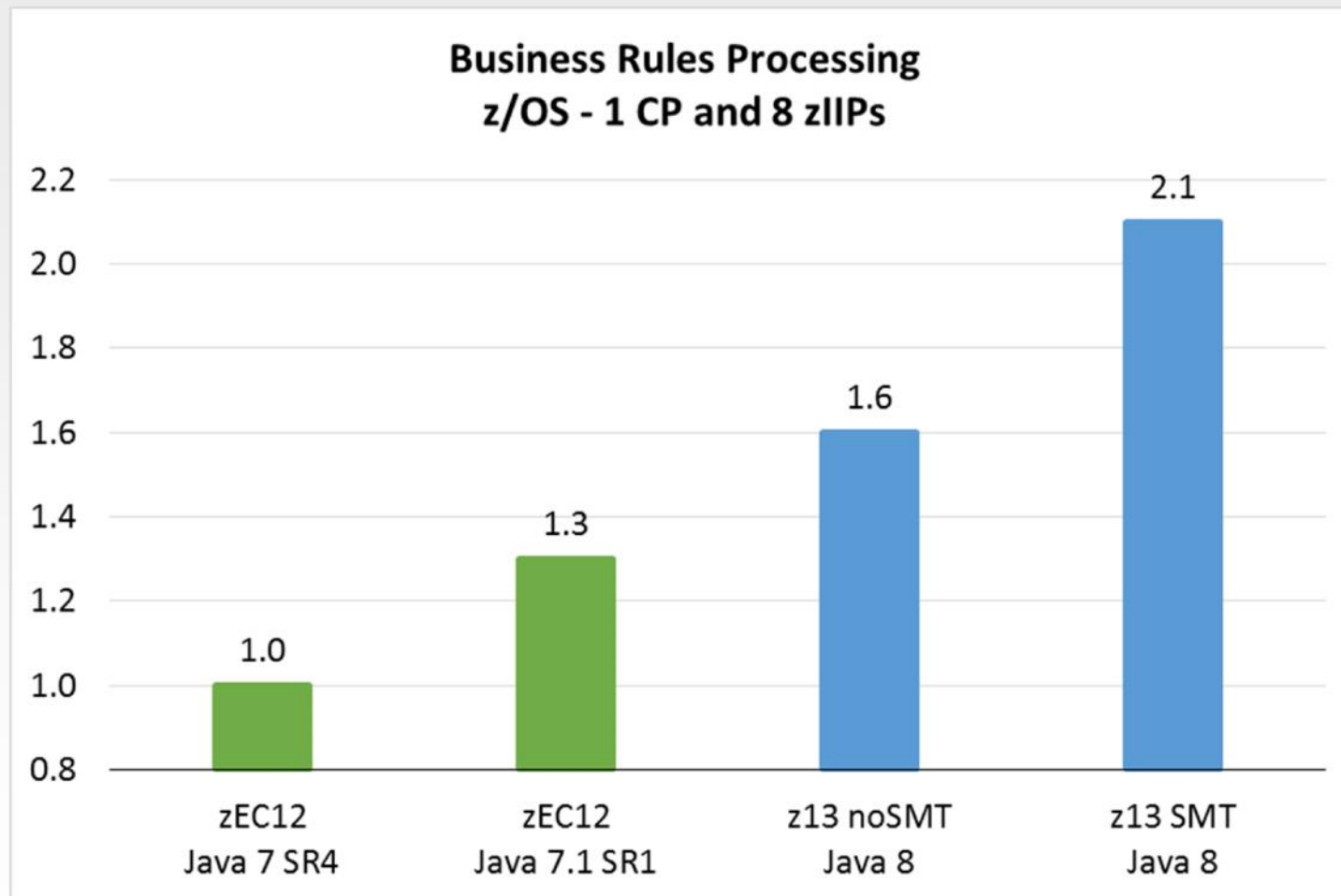
# WebSphere – Linux on z Systems Virtualized Cluster



**Between 1.36x and 1.66x improved throughput for a virtualized WAS cluster running DayTrader on z13 when compared to zEC12**



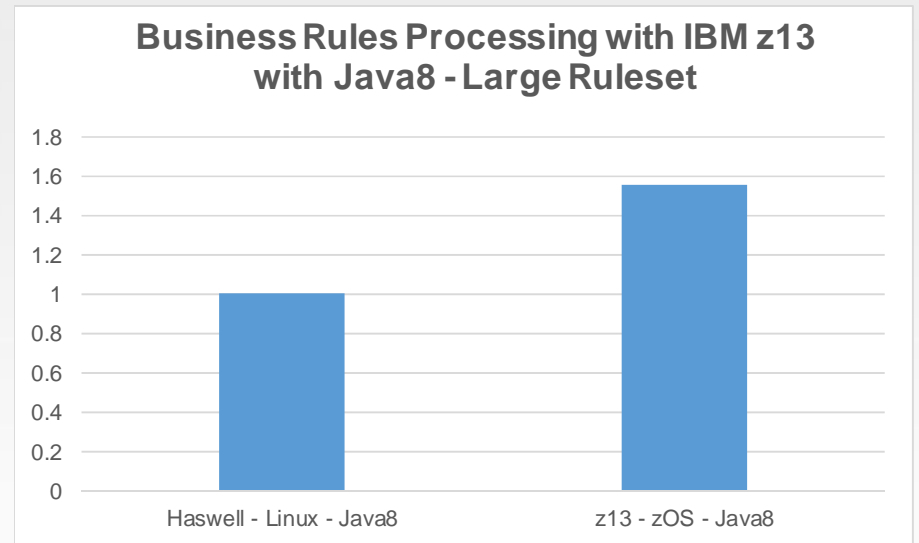
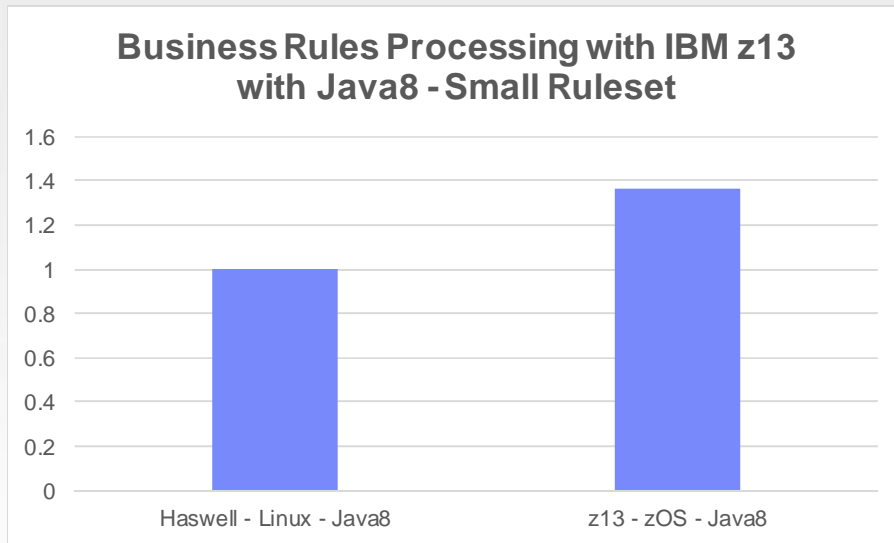
## z/OS - IBM Business Rules Processing with IBM Java 8 and z13



**z/OS - aggregate 2x improvement from IBM Java 8 and IBM z13**



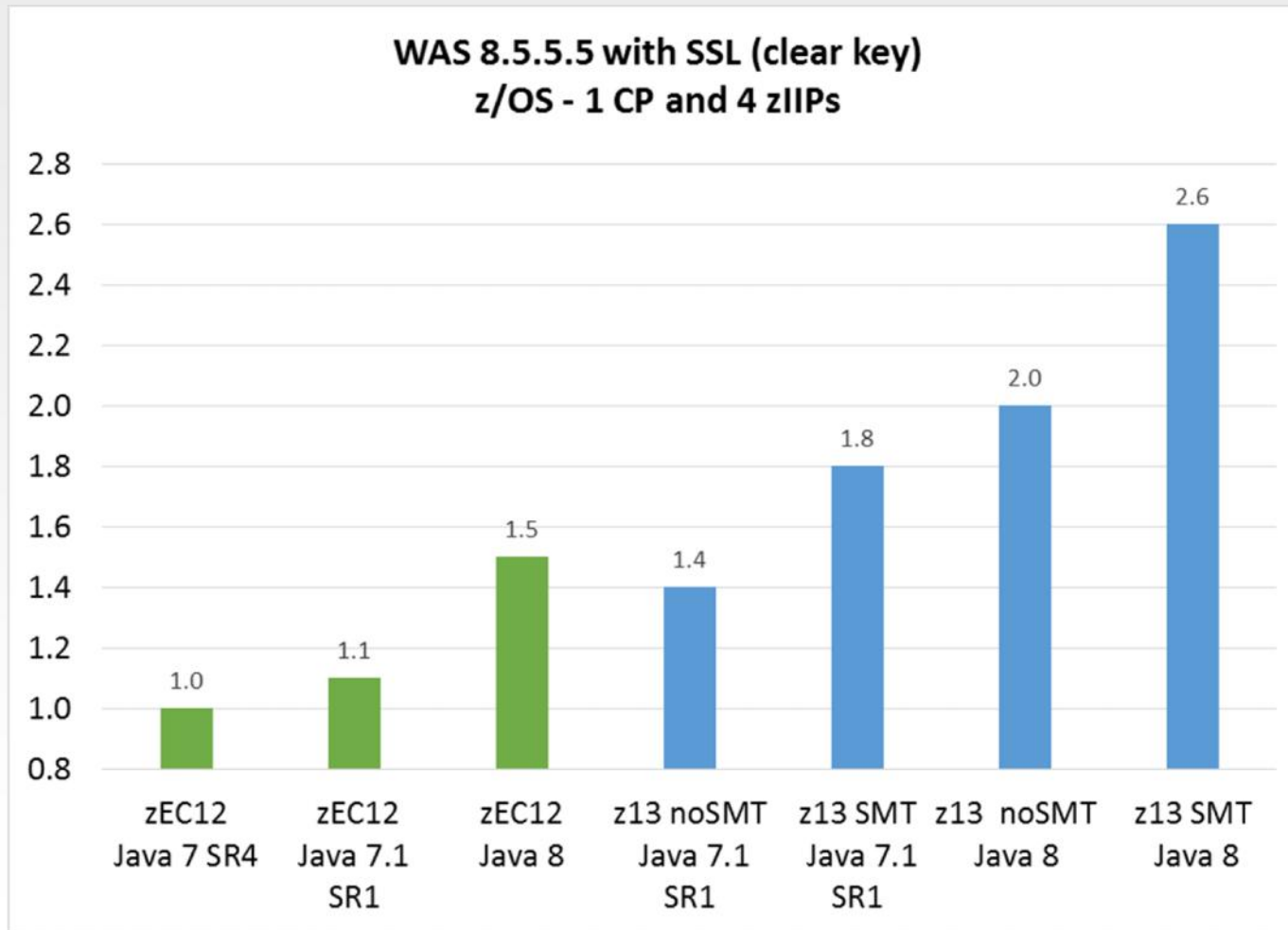
# Business Rules Processing – IBM z13 vs Intel Haswell



**IBM z13 up-to 1.56x better throughput/core processing business rules than Intel Haswell**

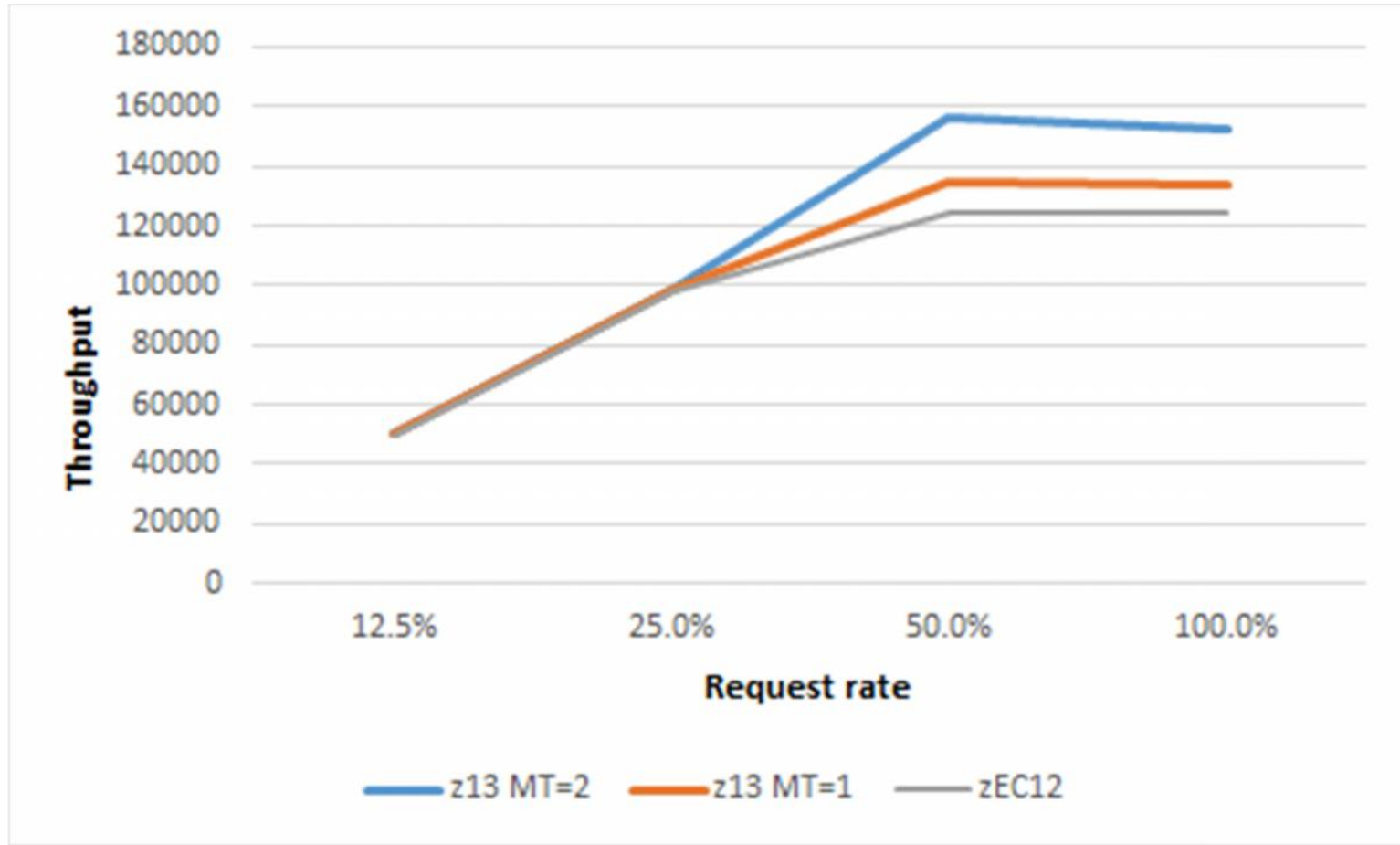


# z/OS WAS/Liberty 8.5.5.5 – SSL-Enabled DayTrader 3.0



**z/OS - 2.6x improvement in throughput with IBM Java 8 and IBM z13**

## Java in CICS – zEC12, z13 and z13 with SMT2

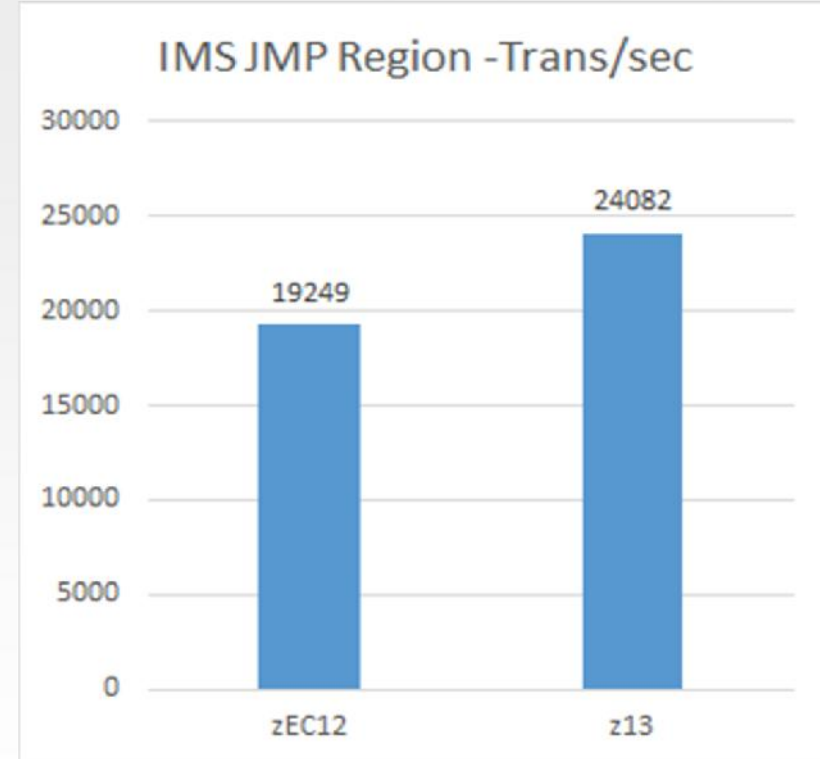
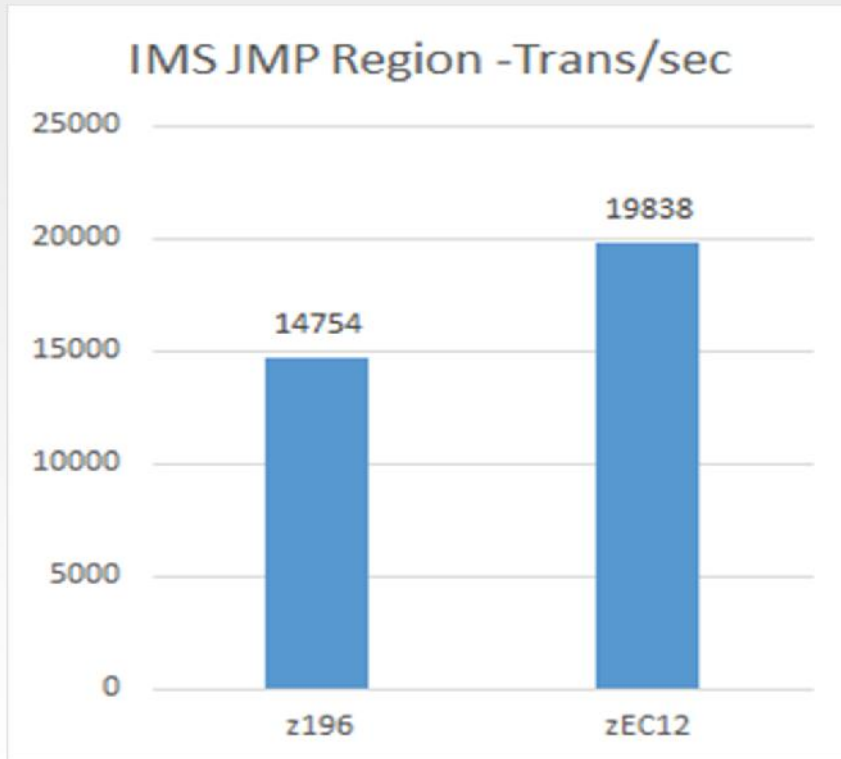


- CICS Transaction Server for z/OS V5.2 with Java V7.1
- Measurements on both IBM z13 and zEC12 obtained using 3 GPs and 1 zIIP

**Disclaimer** Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including 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 results similar to those stated here.



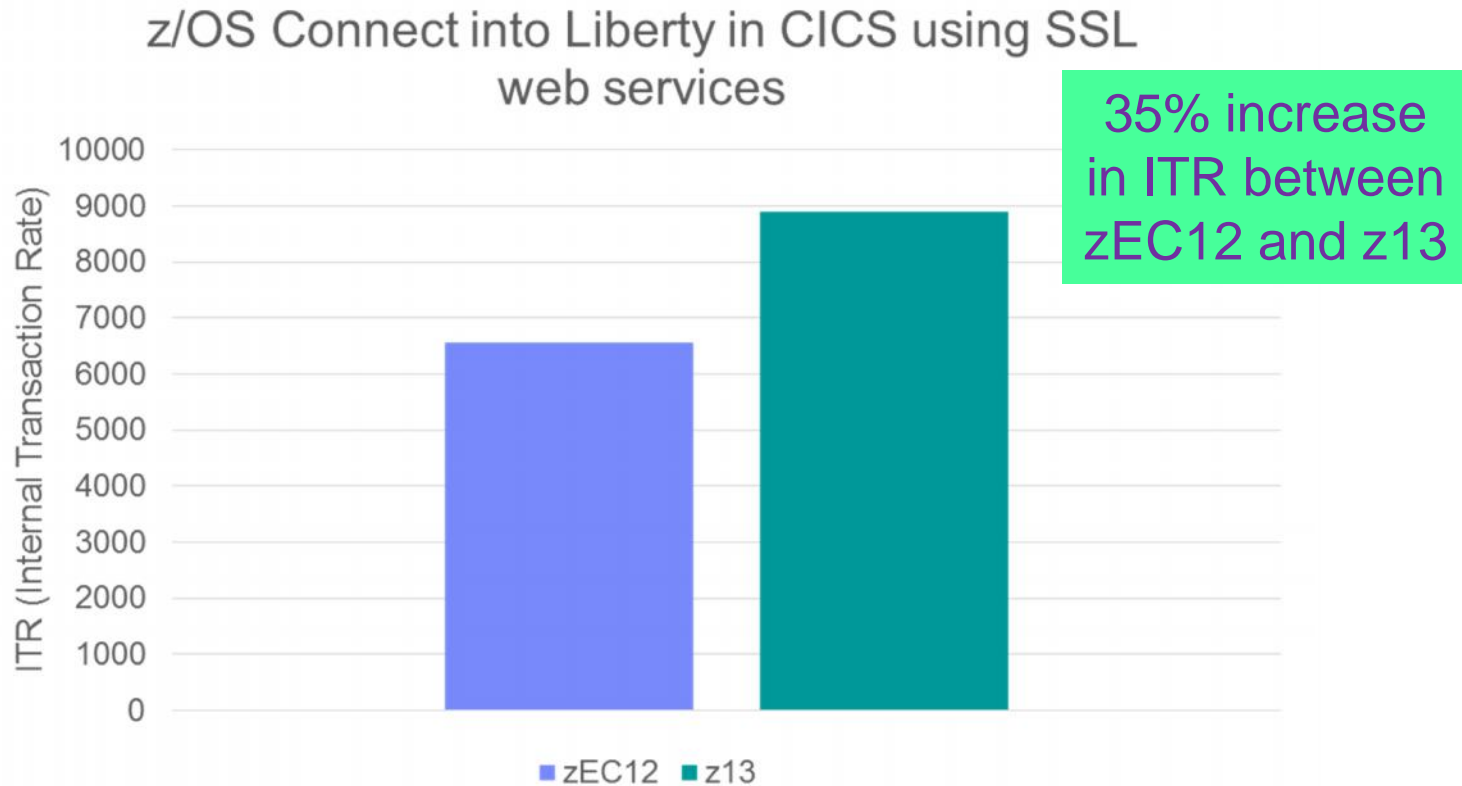
# IMS JMP Regions Overhead



JMP regions 25% better throughput on z13 vs zEC12

JMP regions 34% better through on zEC12 vs z196

# z/OS Connect with CICS



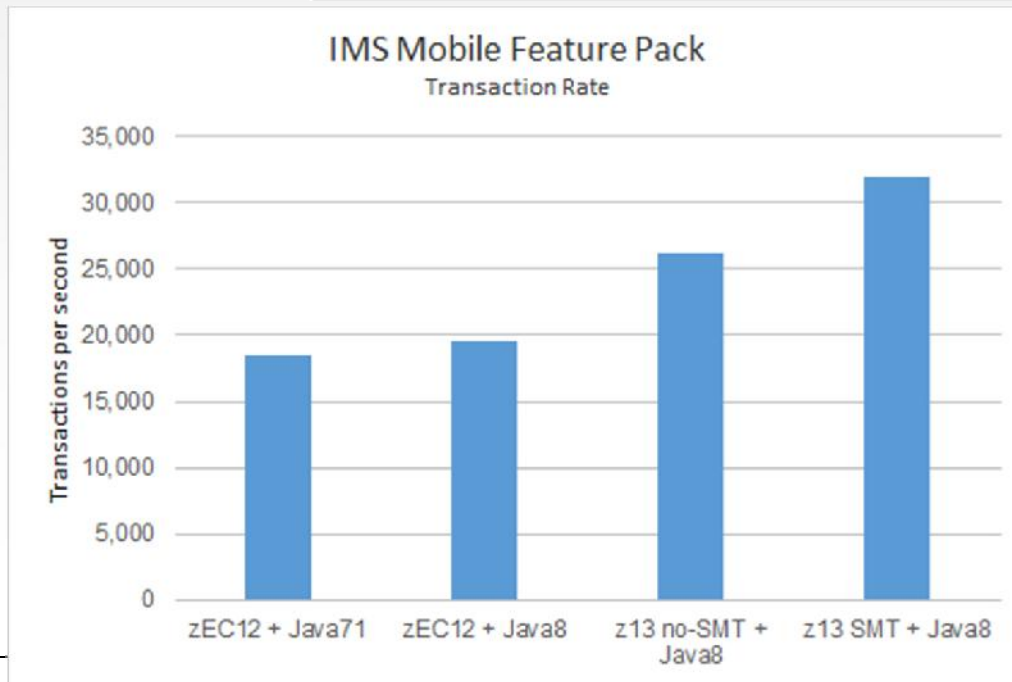
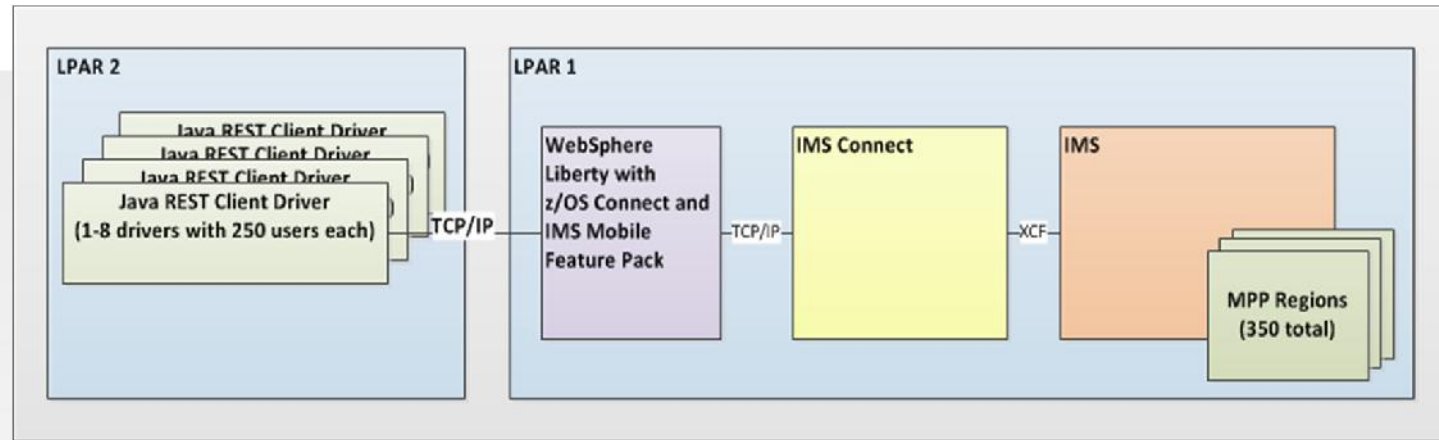
- CICS TS V5.3 open beta developmental code with Liberty V8.5.5.3
- Java V7.1 with IBMJCECCA support enabled
- Measurements on both IBM z13 and zEC12 obtained using 3 GPs and 1 zIIP

**Disclaimer** Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including 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 results similar to those stated here. Some measurements were obtained using beta developmental code.





# IMS Mobile Feature Pack



IMS Mobile Feature Pack **73%** aggregate improvement in throughput with IBM z13 and Java8

# Questions?



Thank  
You





## 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>

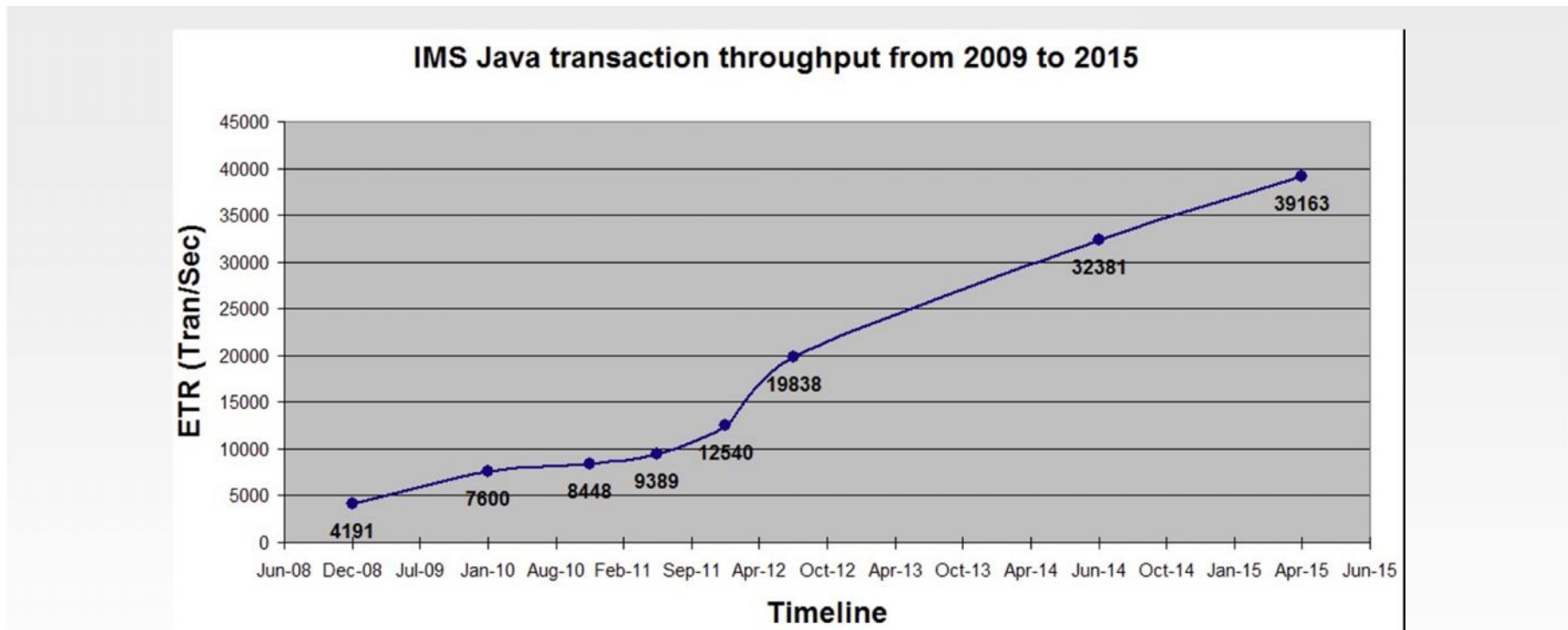


## New and existing supported Java products – z/OS

- IBM 31-bit SDK for z/OS, Java Technology Edition, Version 8.0
  - Web available on March 6th, 2015 at Java SE 8 level
  - Product 5655-DGG, supported on z/OS V1.13 and above
- IBM 64-bit SDK for z/OS, Java Technology Edition, Version 8.0
  - Web available on March 6th, 2015 at Java SE 8 level
  - Product 5655-DGH, supported on z/OS V1.13 and above
- IBM 31-bit SDK for z/OS, Java Technology Edition, Version 7 Release 1
  - Web available on December, 2013 at Java SE 7 level
  - Product 5655-W43, supported on z/OS V1.12 and above
- IBM 64-bit SDK for z/OS, Java Technology Edition, Version 7 Release 1
  - Web available on December, 2013 at Java SE 7 level
  - Product 5655-W44, supported on z/OS V1.12 and above
- IBM 31-bit SDK for z/OS, Java Technology Edition, Version 7.0
  - Web available on October 4, 2011 at Java SE 7 level
  - Product 5655-W43, 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 7 level
  - Product 5655-W44, supported on z/OS V1.10 and above
- 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](#)



# IMS JMP Speed-Test

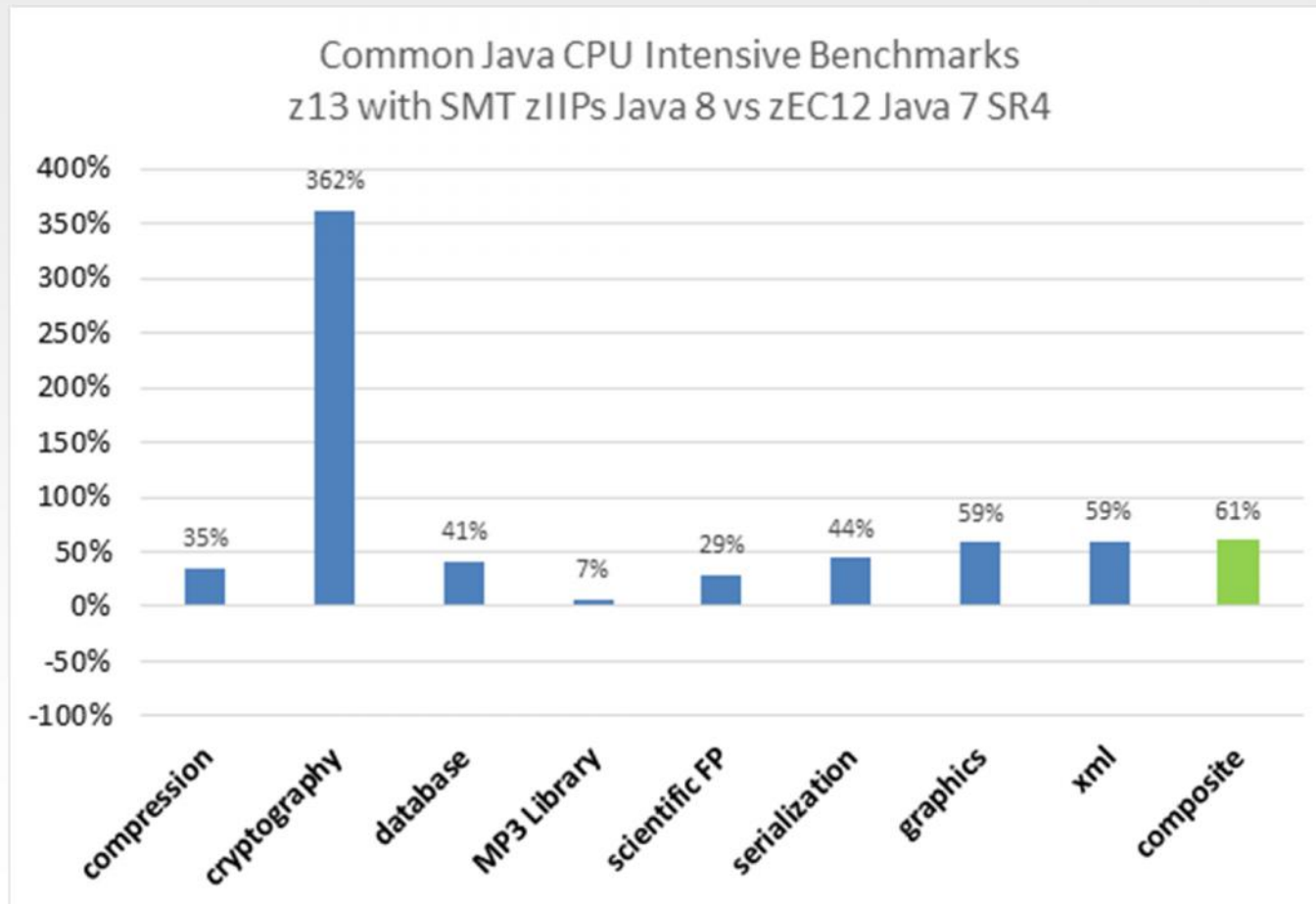


**Over 9.3x aggregate throughput improvement from 2009 to 2015 due to the following enhancements**

- Java version to version performance improvements
- IMS improvements
- Hardware improvements
- DASD improvements



# IBM Java 8: CPU-Intensive Benchmark



IBM z13 and IBM Java 8 show a composite improvement of 61% over zEC12 and Java7 SR4 running the CPU Intensive benchmark



## 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)  $\leftrightarrow$  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  $\leftrightarrow$  Primitive: Convert Packed Decimal(PD), External Decimal(ED), Unicode Decimal(UD)  $\leftrightarrow$  primitive types (int, long)

Decimal  $\leftrightarrow$  Decimal: Convert between dec. types (PD, ED, UD)

Decimal  $\leftrightarrow$  Java: Convert dec. types (PD, ED, UD)  $\leftrightarrow$  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:**

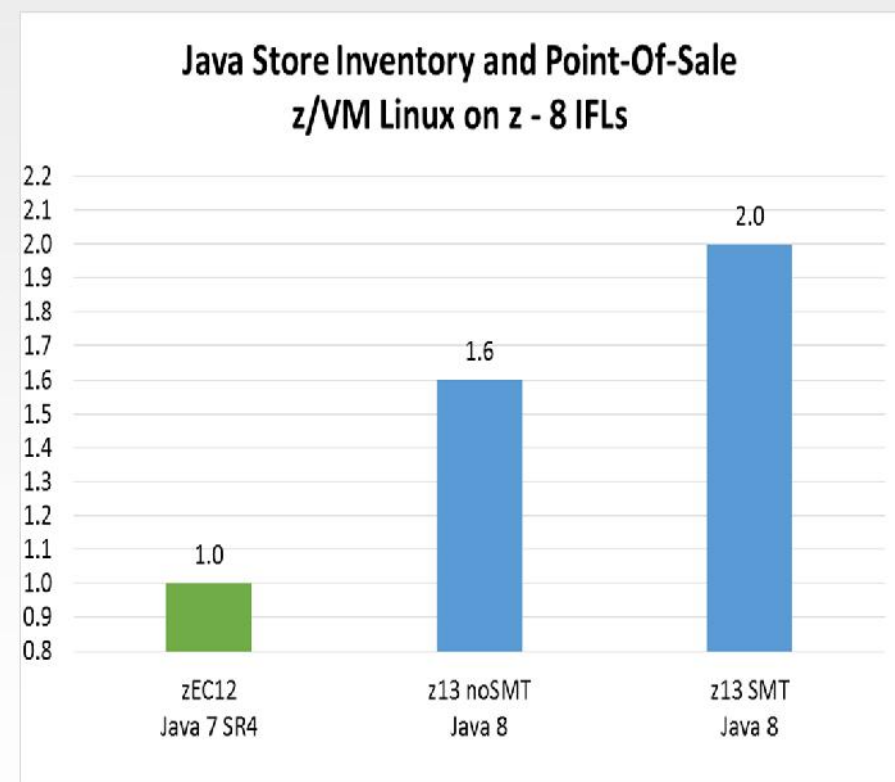
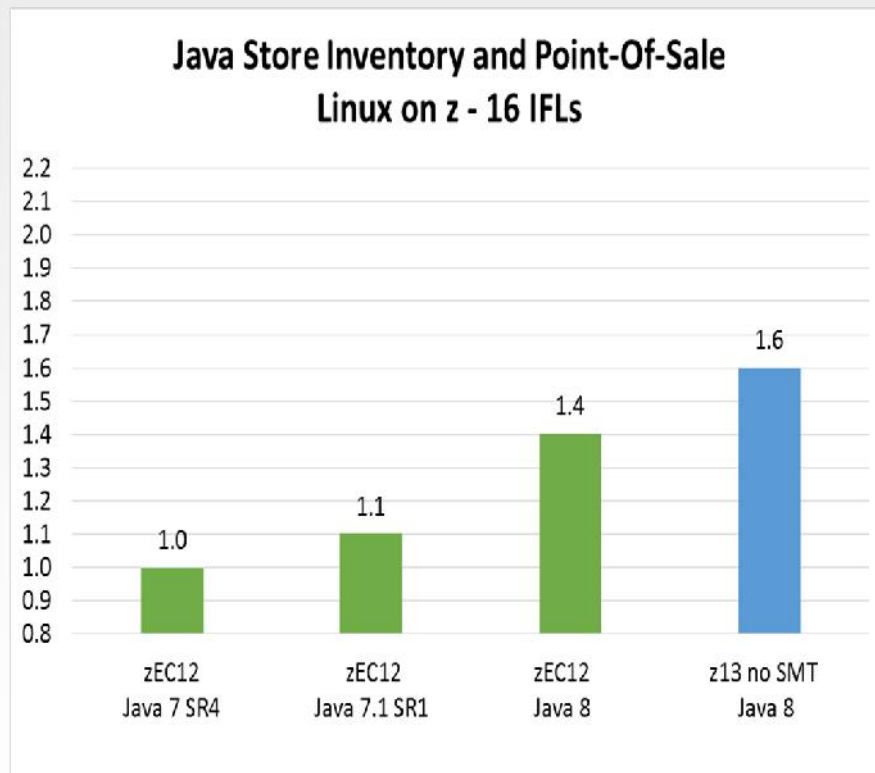
```

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

```



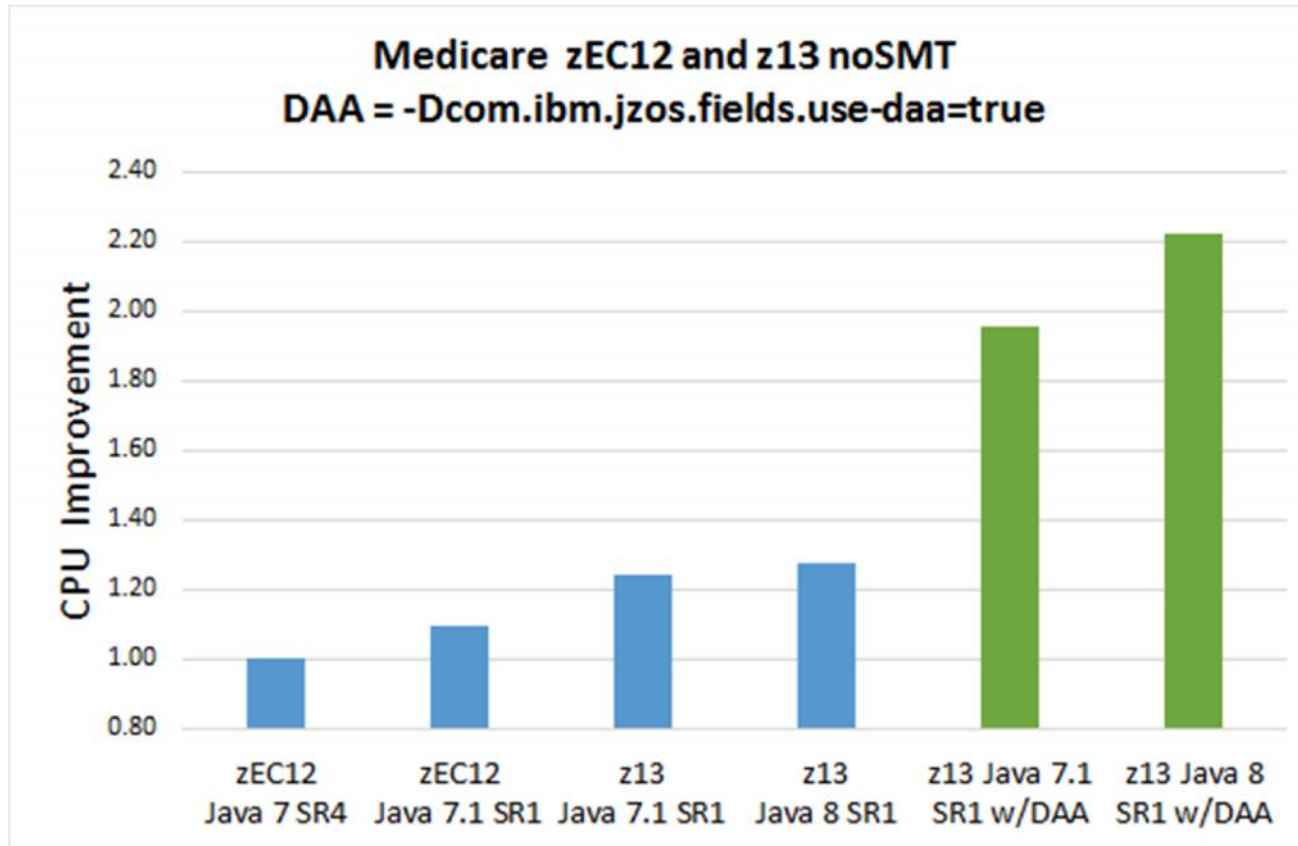
## Linux on z Systems - Java Store, Inventory and Point-of-Sale App with IBM Java 8 and z13



**2x improvement in throughput with IBM Java 8 and IBM z13**

## JZOS Medicare Record Benchmark : With and without DAA

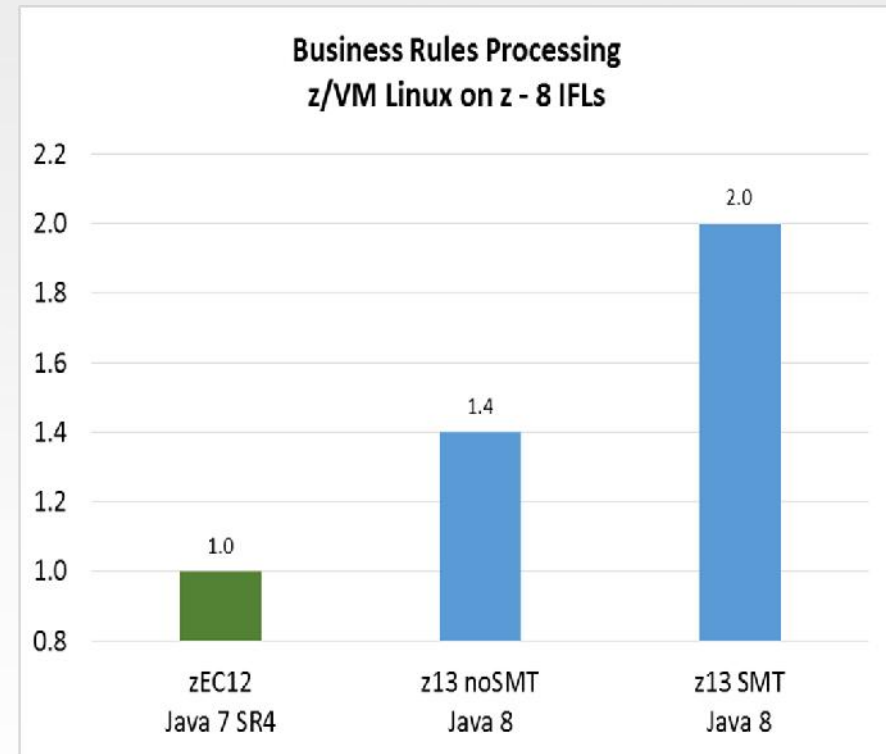
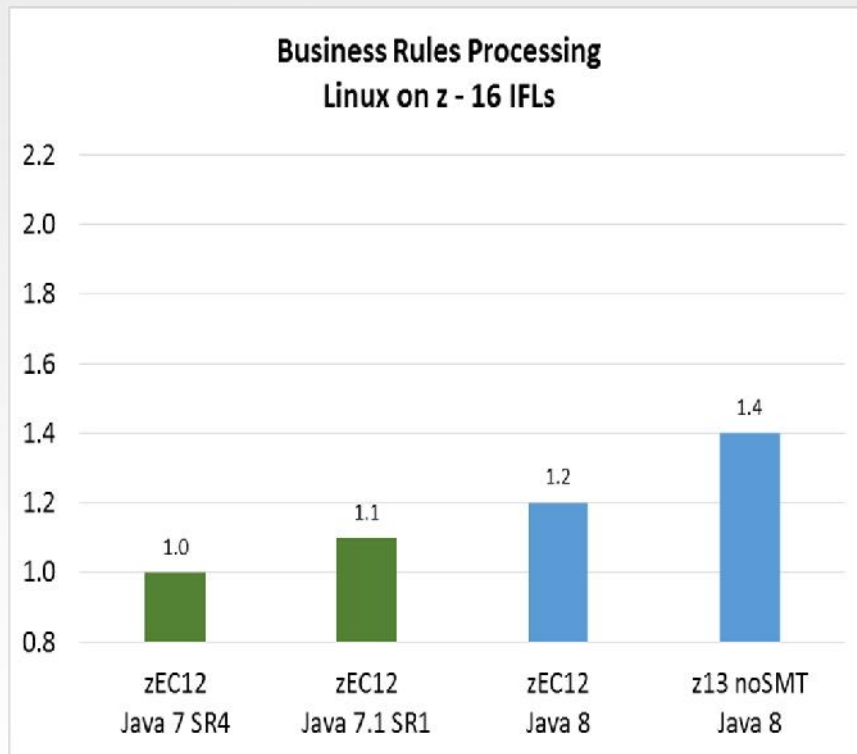
CPU-time processing 5 million medical records



- 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



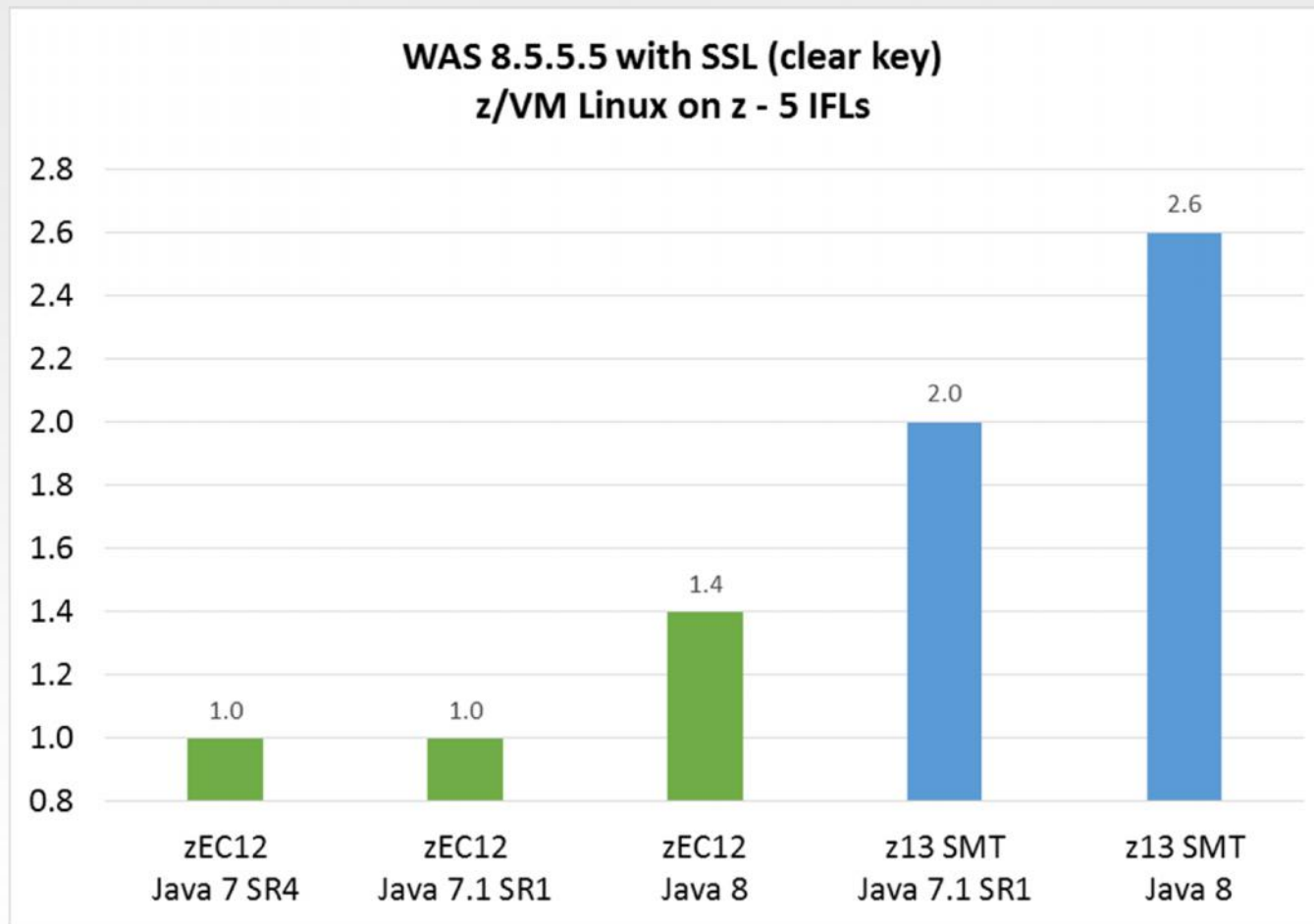
## Linux on z Systems - IBM Business Rules Processing with IBM Java 8 and z13



**z/VM IFL aggregate 2x improvement from IBM Java 8 and IBM z13**



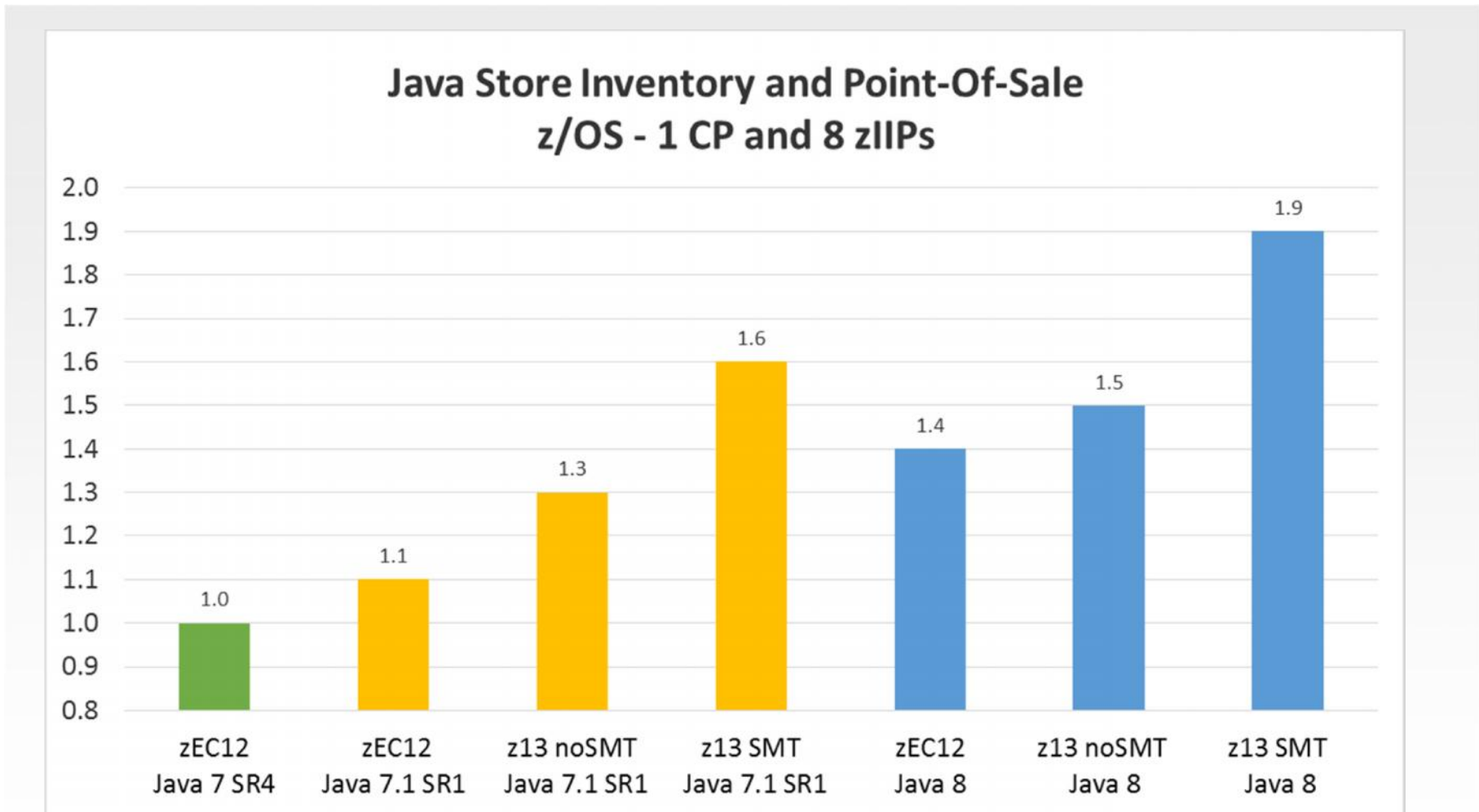
## z/VM IFLs WAS/Liberty 8.5.5.5 – SSL-Enabled DayTrader 3.0



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



## z/OS - Java Store, Inventory and Point-of-Sale App with IBM Java 8 and z13



**z/OS - 1.9x improvement in throughput with IBM Java 8 and IBM z13**

## Section Header

