IBM Java 8 and z13 - Hardware and Software Co-Design at Its Finest

Iris Baron – IBM Java JIT Compiler Development

Session 17635
Thursday, August 13, 2015: 08:30 AM - 09:30 AM
Dolphin, Asia 3

Follow on twitter @JavaOnZ

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

Evolving Java as a Workload Optimized System on Z

Enable integration of Java-based applications with core Z 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
  - Providing seamless portability across platforms
- **Pervasive and integrated across the z eco-system**
  - Java business logic with all z middleware (IMS, CICS, WAS, etc.)
  - Inter-operability with legacy batch and OLTP assets
- **Deep z Systems exploitation**
  - SDK extensions enabled z QoS for full integration with z/OS
  - zAAP/zIIP specialty engines provide low-cost Java capacity
- **Performance**
  - A decade of hardware/software innovations and optimizations
  - Industry leading performance with IBM J9 Virtual Machine
  - Enabling tight data locality for high-performance and simplified systems

IBM Java Runtime Environment

- IBM's implementations of Java 5, 6, 7, 8 are built with **IBM J9 Virtual Machine** and **IBM Testarossa JIT Compiler** technologies
  - Independent clean-room JVM runtime & JIT compiler
- Combines best-of breed from embedded, development and server environments… from a cell-phone to a mainframe!
  - Lightweight flexible/scalable technology
  - World class garbage collection – gencon, balanced GC policies
  - Startup & Footprint - Shared classes, Ahead-of-time (AOT) compilation
  - 64-bit performance - Compressed references & Large Pages
  - Deep z Systems exploitation – z13/zEC12/z196/z10/z9/z990 exploitation
  - Cost-effective for z - zIIP Ready!
- Millions of instances of J9/TR compiler

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:

WASz minimizes physical tiers

3-4x improvement for one of HCSC's largest WAS applications when moving from distributed to zOS

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

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:

WASz minimizes physical tiers

3-4x improvement for one of HCSC's largest WAS applications when moving from distributed to zOS

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

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/

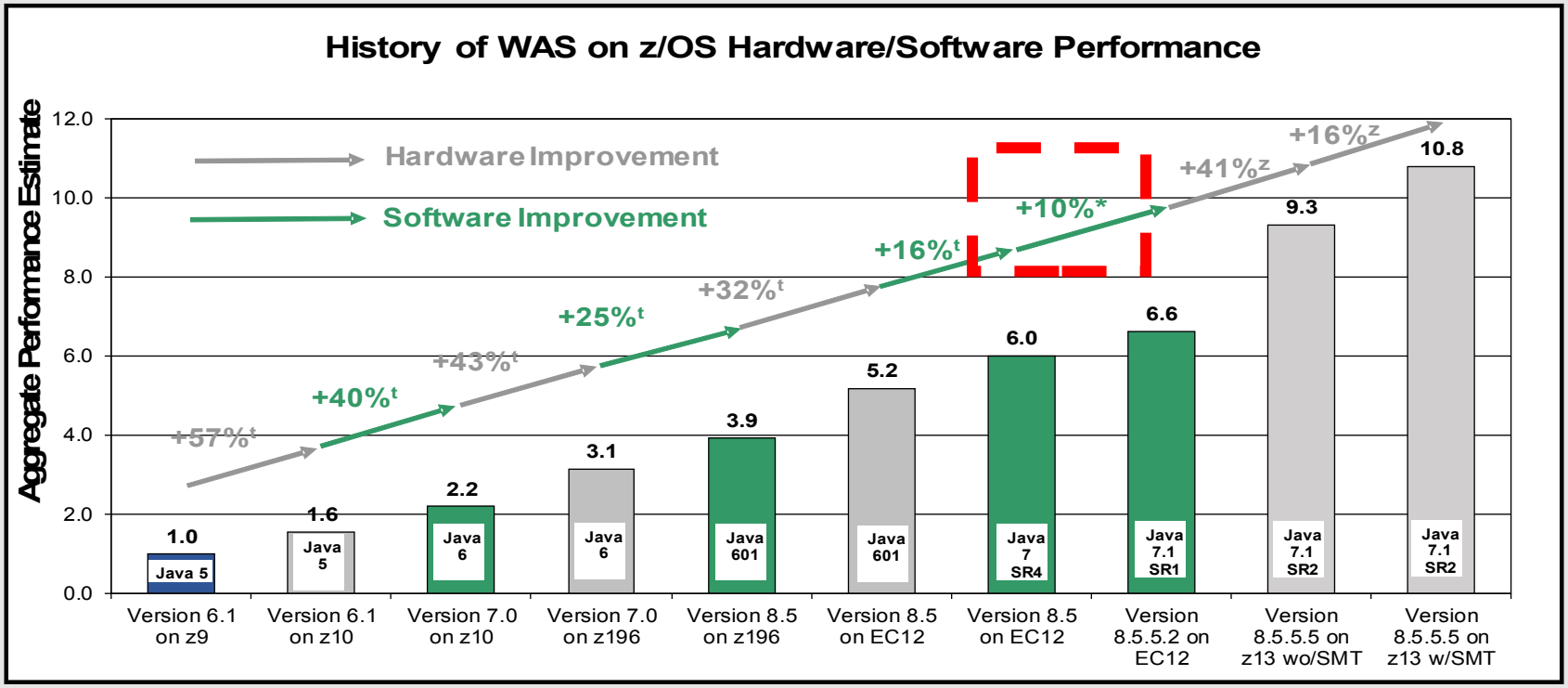
IBM SDK for z/OS, Java Tech. Edition, Version 7 Release 1 (IBM Java 7R1)

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**

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

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

(Controlled measurement environment, results may vary)

Java Road Map

Language Updates

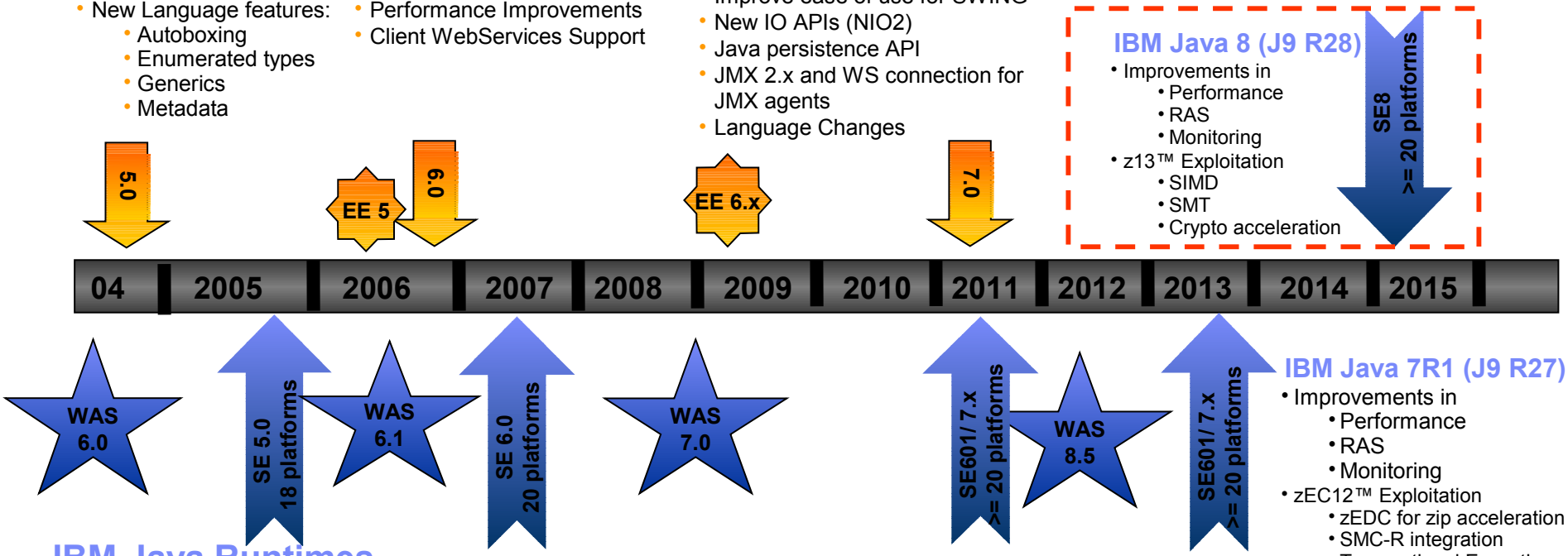
Java 5.0
- New Language features:
  - Autoboxing
  - Enumerated types
  - Generics
  - Metadata

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, 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 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 7 (J9 R26 SR3)
- Improvements in
  - Performance
- zEC12™ Exploitation
  - Transactional Execution
  - Flash 1Meg pageable LPs
  - 2G large pages
  - Hints/traps

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

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

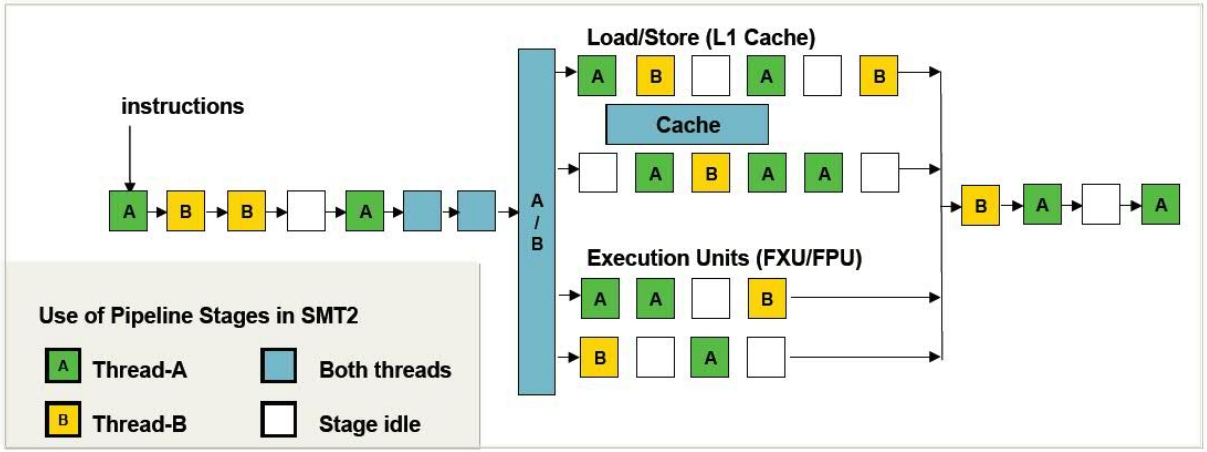
Up to **50%** improvement in throughput for generic applications

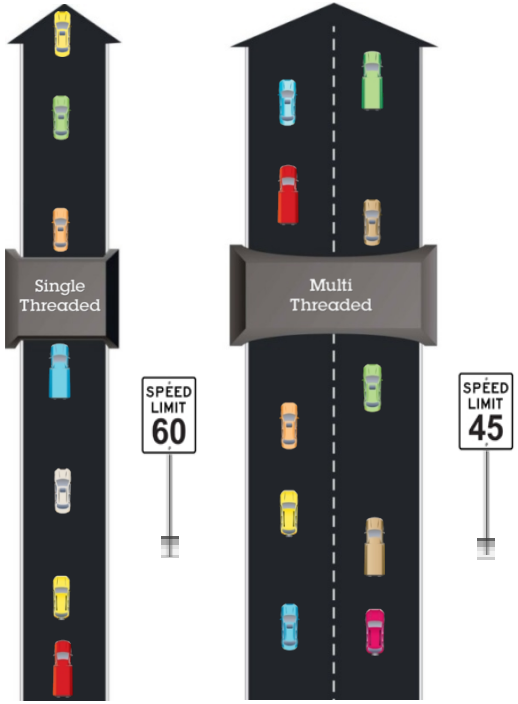
Up to **2X** improvement in throughput per core for security enabled applications

- 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 z13: SMT – Simultaneous Multi-Threading

- 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

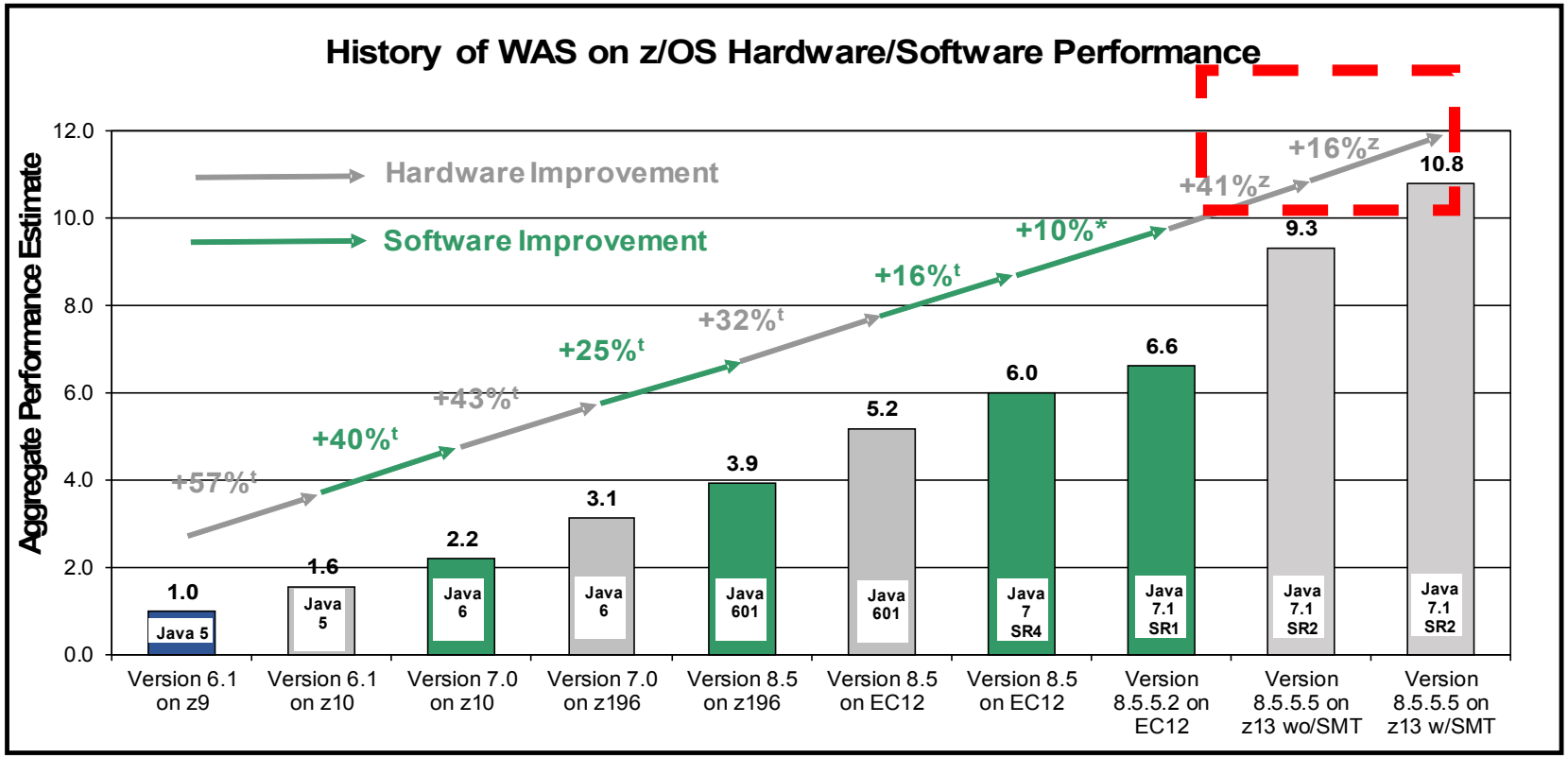




Two zIIP lanes handle more traffic overall

WAS on z/OS – DayTrader

Aggregate HW, SDK and WAS Improvement: WAS 6.1 (IBM Java 5) on z9 to WAS 8.5.5.5 (IBM Java 7R1) on z13



10.8x aggregate hardware and software improvement comparing

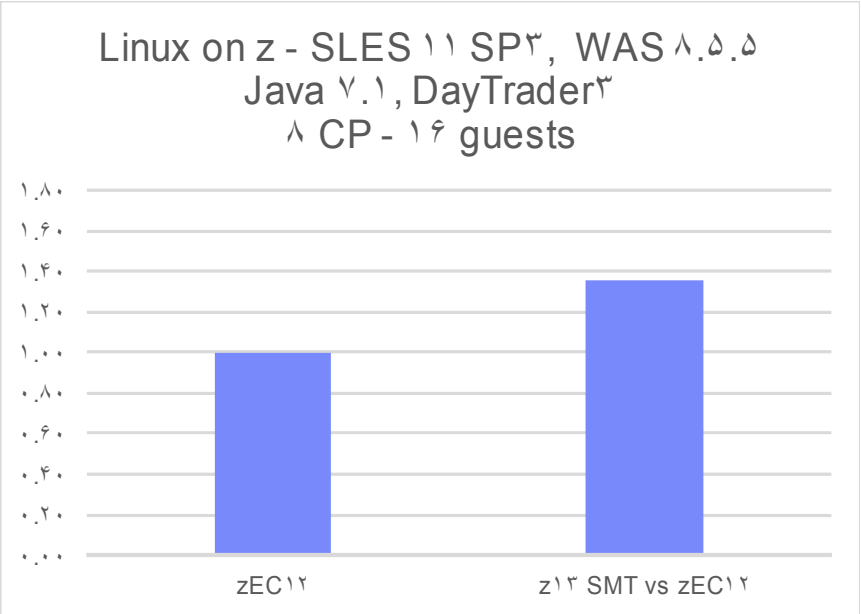
WAS 6.1 IBM Java5 on z9 to WAS 8.5.5.2 IBM Java7R1 on z13 w/SMT

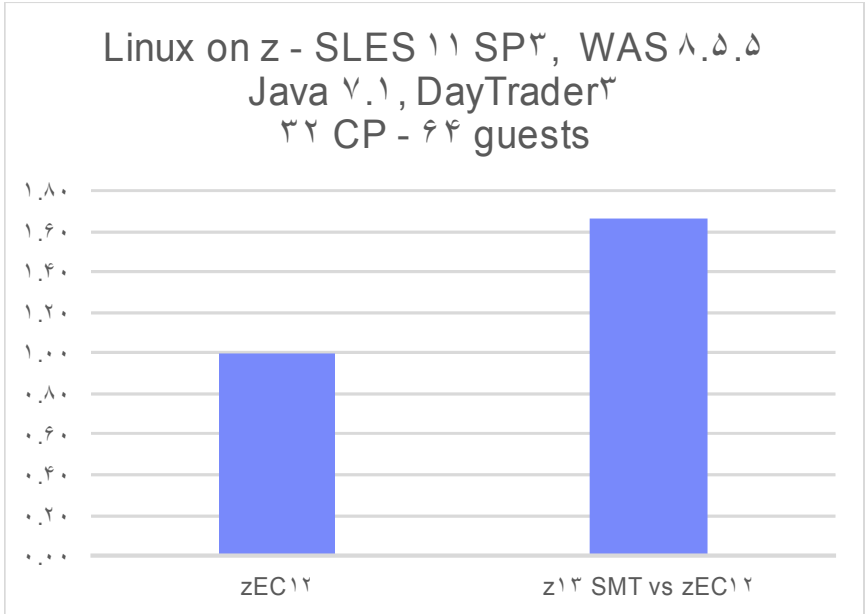
z zIIPs DayTrader 3
* DayTrader3
t DayTrader2

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

(Controlled measurement environment, results may vary)

WebSphere – Linux on z Virtualized Cluster



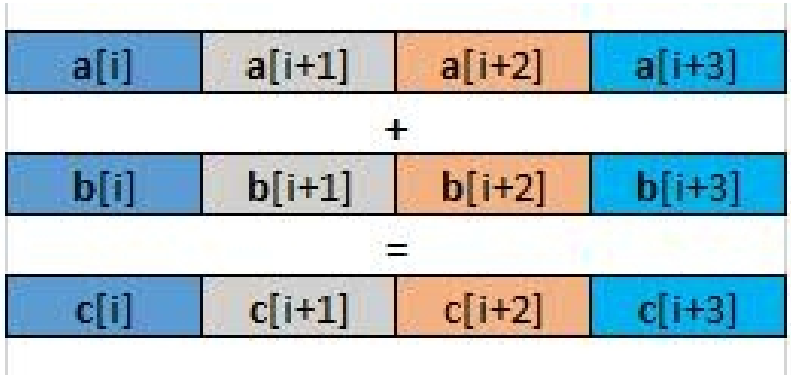


Between 1.36x to 1.66x improved throughput for a virtualized WAS cluster running DayTrader 3.0 on IBM z13 when compared to zEC12

(Controlled measurement environment, results may vary)

IBM z13: SIMD – Single Instruction Multiple Data

- Hardware for exploiting data-parallelism
  - Large uniform data-set that needs the same operation performed on each element
  - Can offer dramatic speedup to data-parallel operations (matrix ops, string processing, 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]);
    }
```

IBM Java 8 - String, Character Conversion and Loop Acceleration with SIMD

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

- **java/lang/String**
  - compareTo
  - compareToIgnoreCase
  - contains
  - contentEquals
  - equals
  - indexOf
  - lastIndexOf
  - regionMatches
  - toLowerCase
  - toUpperCase
  - getBytes

- **java/util/Arrays**
  - equals (primitive types)
- **String encoding converters**
  - ISO8859-1
  - ASCII
  - UTF-8 / UTF-16
- **Auto-SIMD**
  - Simple loops
  - (e.g. Matrix Multiplication)

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

(Controlled measurement environment, results may vary)

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

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**

Java 8 – Lambdas

New syntax to allow for concise and expressive code snippets

Lambda expression:
(argument List) → Body

*Can be thought of as 'anonymous functions'

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

```
Collections.sort(people, (Person x, Person y) ->x.getLastName().compareTo(y.getLastName()) );
```

Compiler can often infer parameter types in a lambda expression

```
Collections.sort(people, (x, y) ->x.getLastName().compareTo(y.getLastName()) );
```

http://www.dzone.com/links/presentation_languagelibraryvm_coevolution_in_jav.html

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

Java 8 – Virtual Extension Methods

- Extend well established data structures while retaining compatibility
- Language enhancement to provide default implementations in interfaces
  - Interface declarations run if classes do not provide an implementation

```
public interface Iterator,E> {
    public boolean hasNext();
    public E next();
    ...
    public default skip(int i){
        for(; i > 0 && hasNext(); i--)
            next();
      }
}
```

Crypto Acceleration (SSL)

Crypto acceleration across the entire SSL connection

Key Agreement (public key)

- ECDHE and ECDSA (ECC)
  - NIST P256 curve

Cipher

- Cipher: AES, 3DES, DES
- Mode: CBC,CFB, OFB, ECB

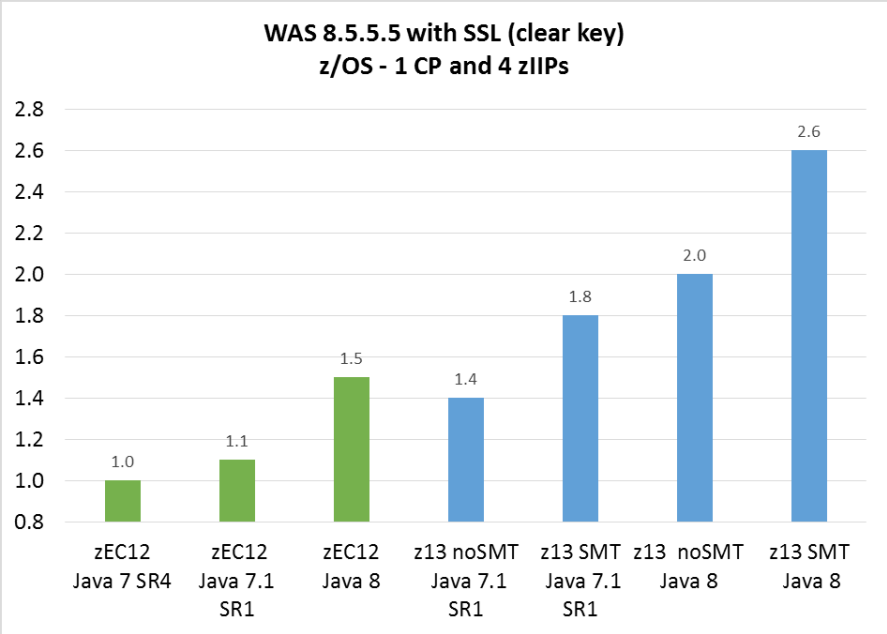
Message Digest (hashing)

- SHA1
- SHA2

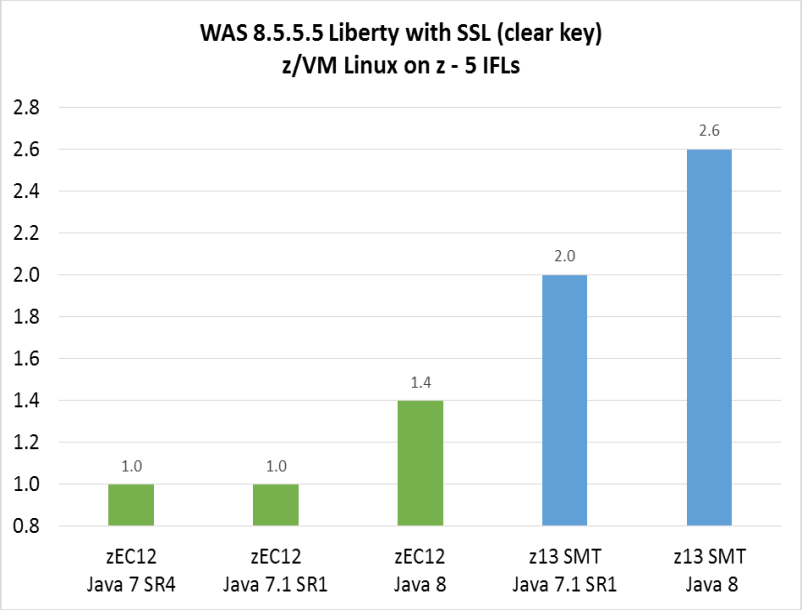
CPACF exploitation

- Java 8 exploitation of CPACF is the default for z9 and above on both z/OS and zLinux
- Crypto acceleration is used in IBMJCE provider (clear key), default in the IBM JDK
  - e.g. EF transparently leverages the new acceleration by using IBMJCE
    Encryption of text files and SVC dumps completed in half the elapsed time and one third the CPU time.

WAS Liberty and z13



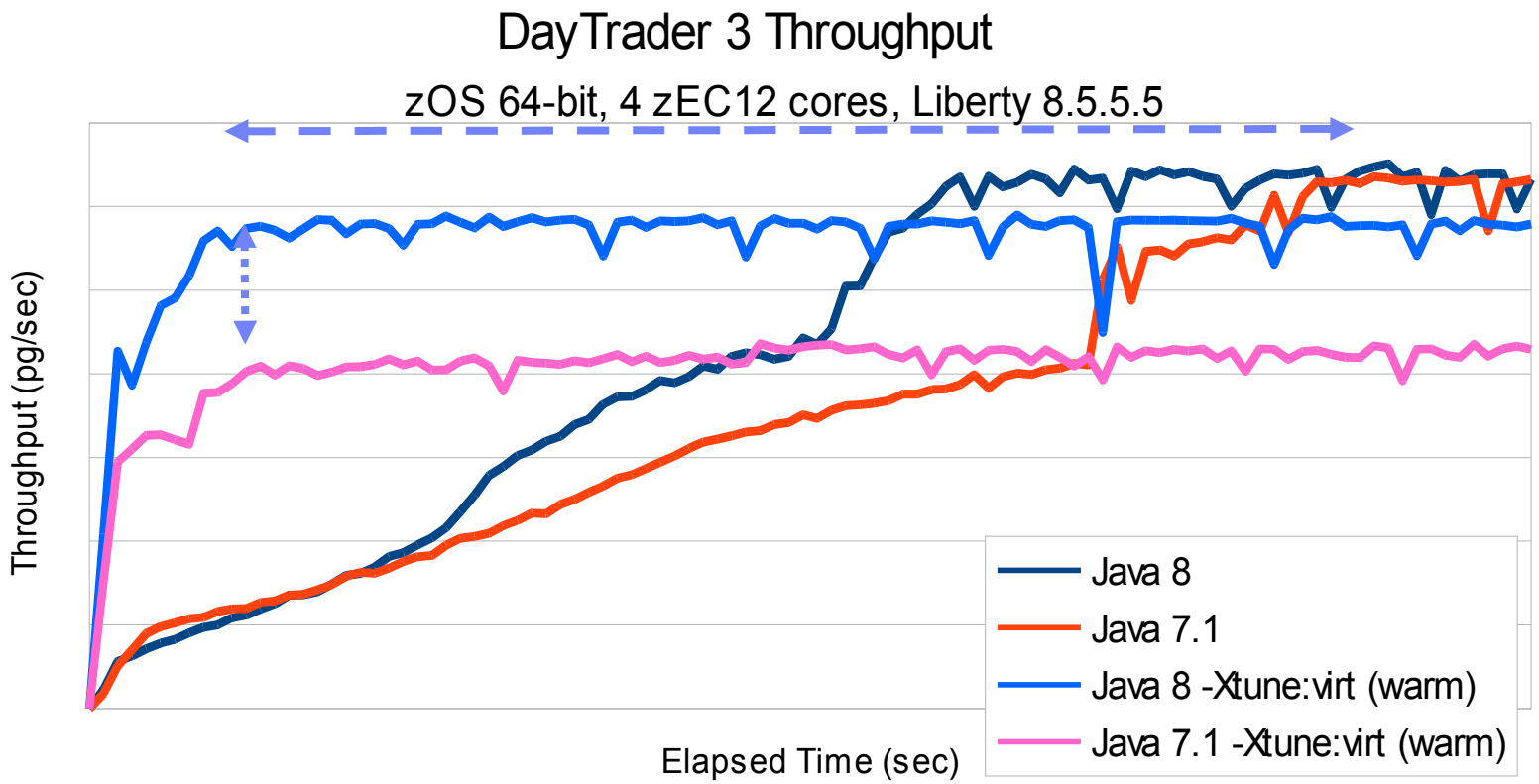
2.6X improvement in throughput for SSL-enabled DayTrader 3.0 with WAS Liberty 8.5.5.5 on **z/OS** using Java 8 and z13 with SMT2, compared with Java 7 SR4 on zEC12



2.6X improvement in throughput for SSL-enabled Day Trader 3.0 and IBM Java 8 under z/VM **Linux on z** on a z13 compared with zEC12

(Controlled measurement environment, results may vary)

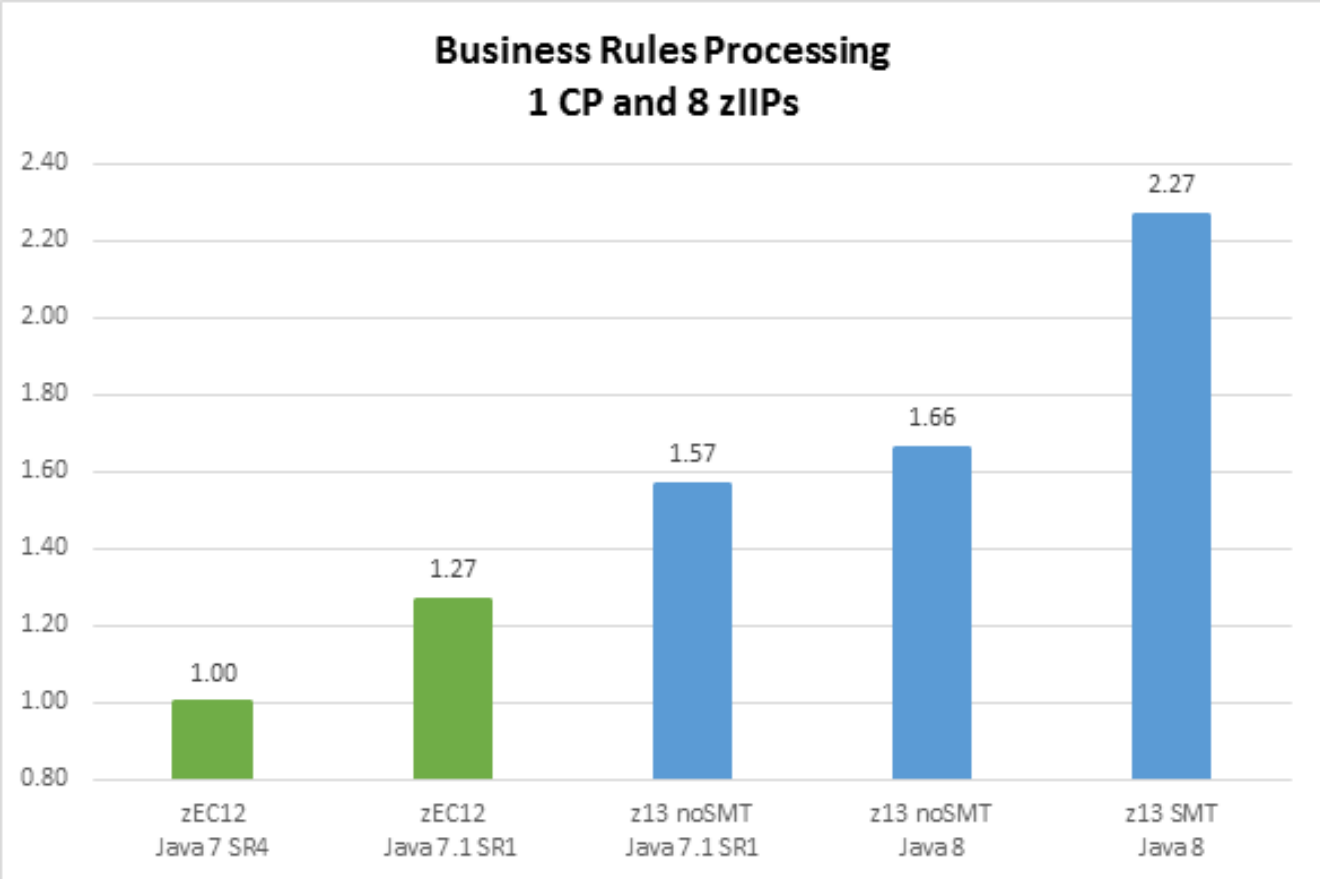
zOS Liberty Ramp-up with IBM Java 8



- IBM Java 8 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)

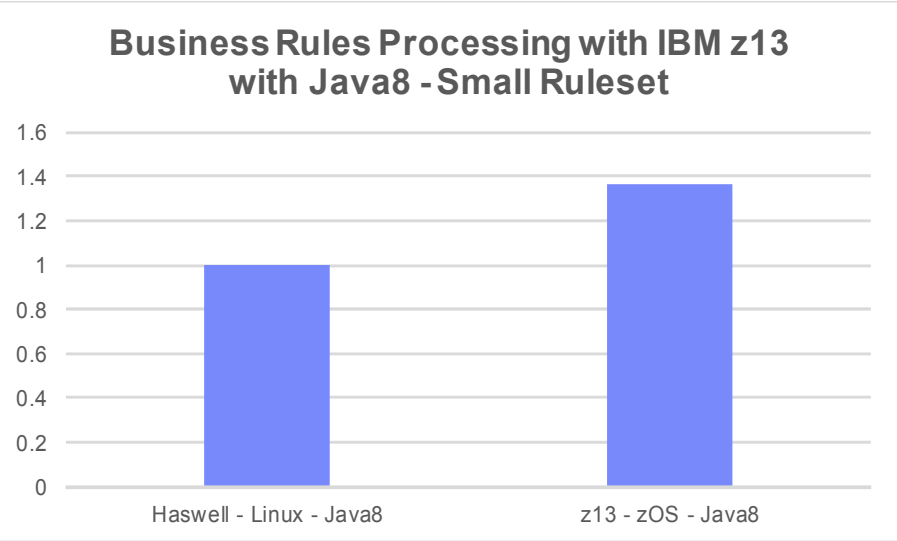
Business Rules Processing with IBM Java 8 and z13

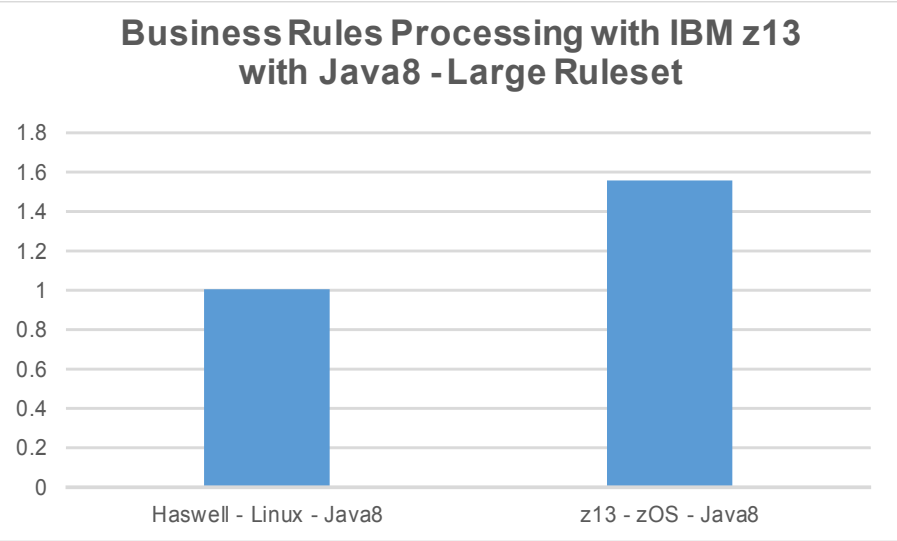


Aggregate 2.27x improvement from IBM Java 8 and IBM z13

(Controlled measurement environment, results may vary)

Business Rules Processing – IBM z13 vs Intel Xeon E5-26xx v3



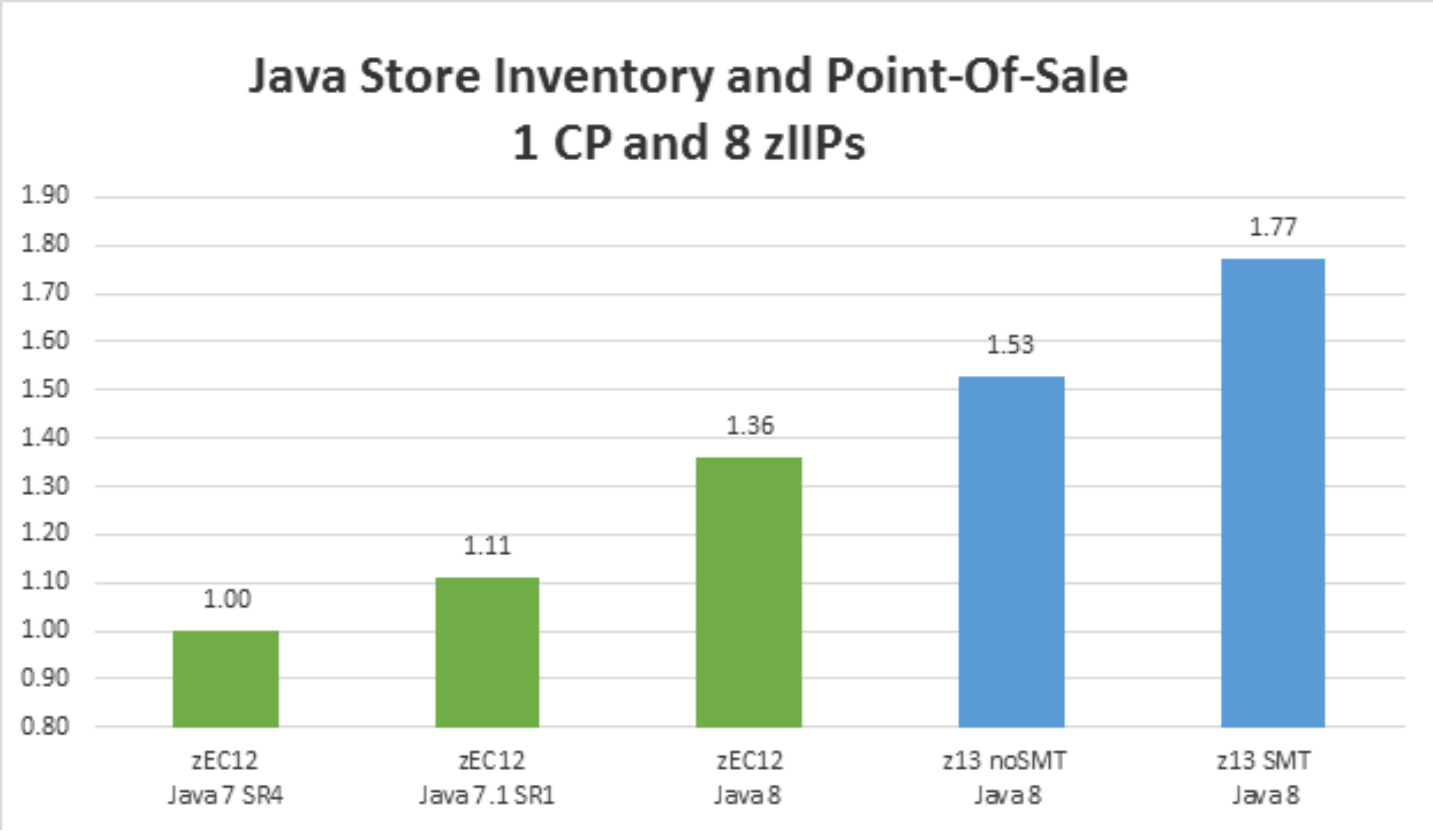


IBM z13 up-to 1.5x better throughput/core processing business rules than Intel Xeon E5-26xx v3 (Haswell)

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

(Controlled measurement environment, results may vary)

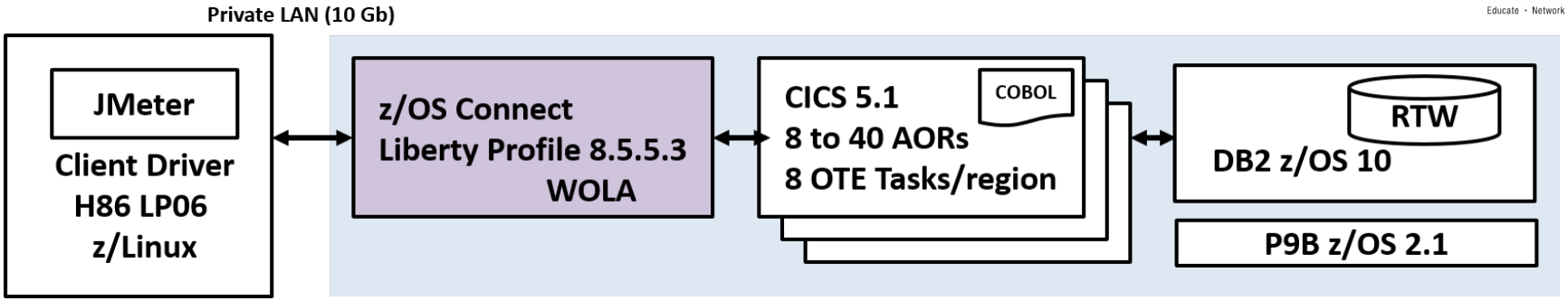
Java Store, Inventory and Point-of-Sale App with IBM Java 8 and IBM z13

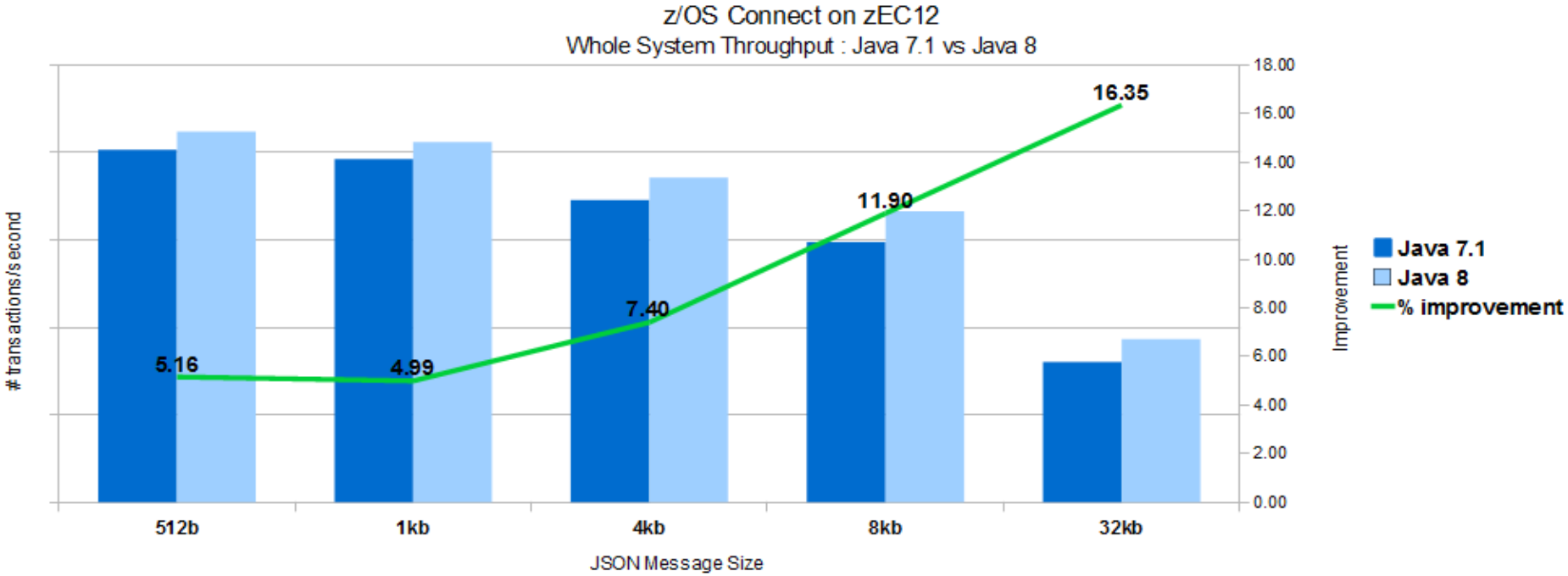


1.77x improvement in throughput with IBM Java 8 and IBM z13

(Controlled measurement environment, results may vary)

Mobile on z – z/OS Connect on IBM Java 8 and zEC12

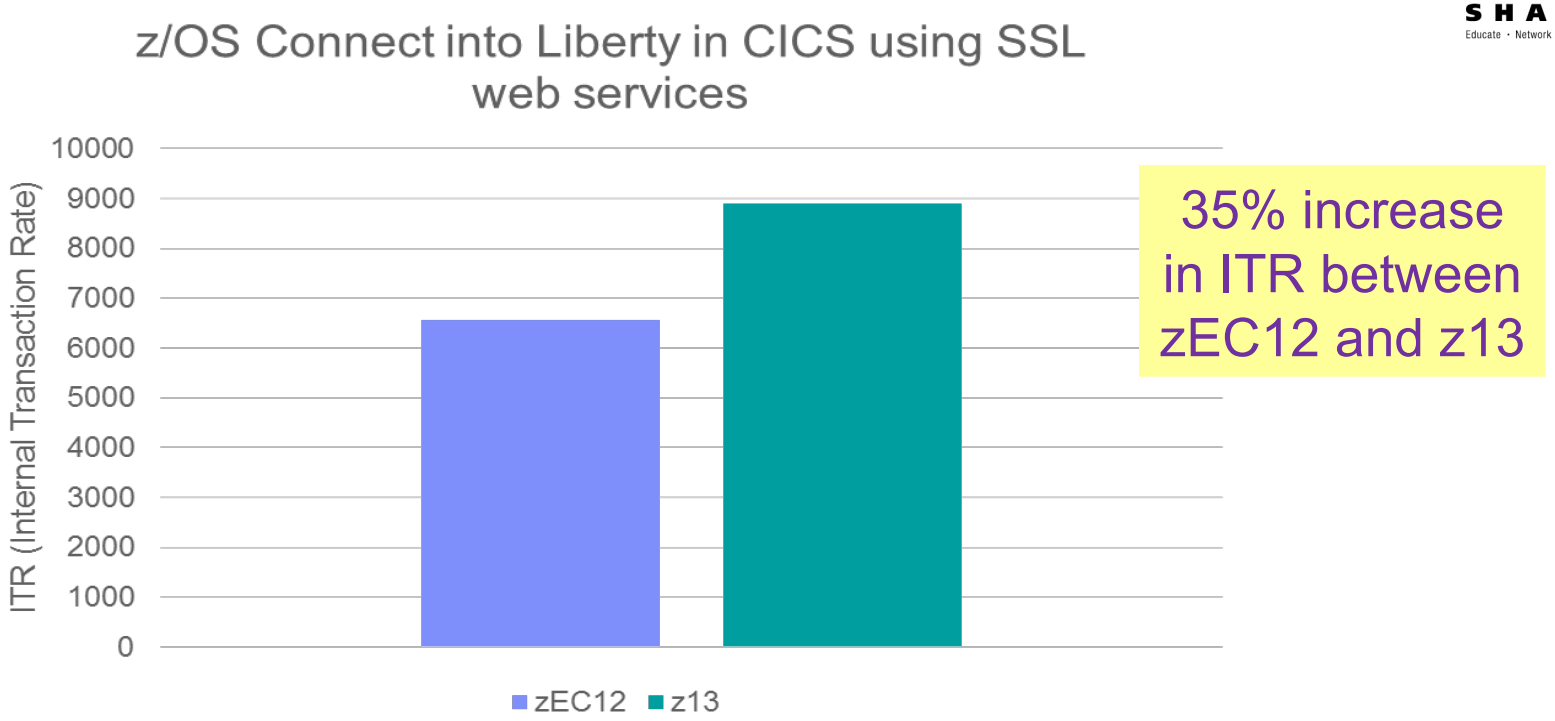




5-16.4% throughput improvement from IBM Java 8 and IBM zEC12

(Controlled measurement environment, results may vary)

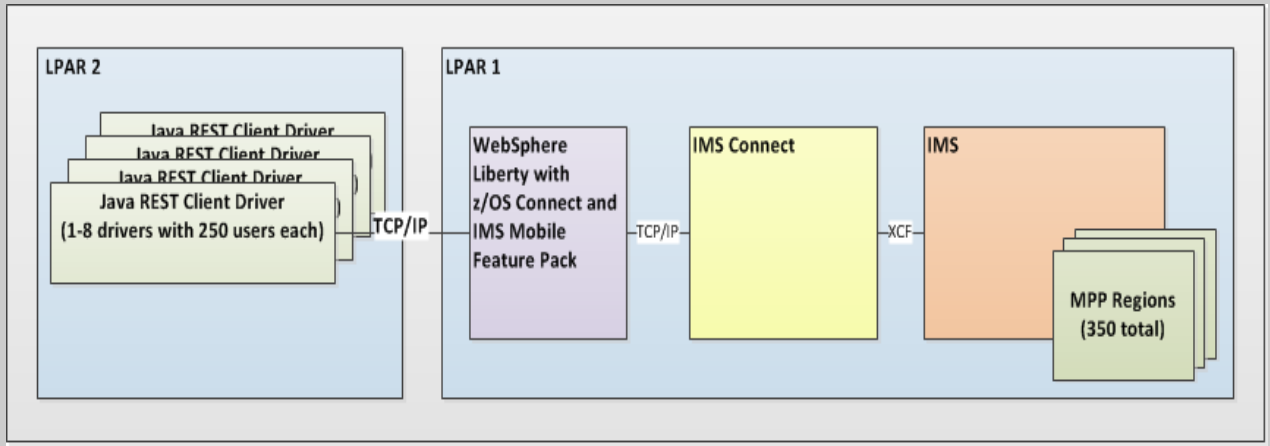
z/OS Connect with CICS

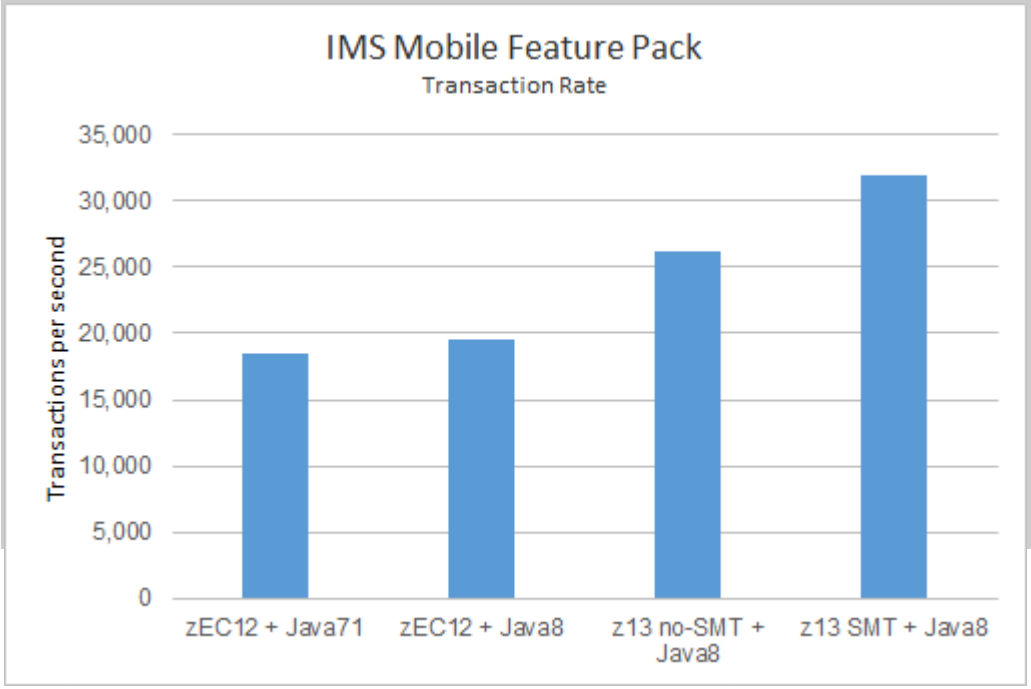


- CICS TS V5.3 open beta developmental code with Liberty V8.5.5.3
- Java V7.1 with IBMJCECCA support enabled
- Measurements on both IBM z13 and zEC12 obtained using 3 GPs and 1 zIIP

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 used beta developmental code.

z/OS Connect with IMS (Mobile Feature Pack)





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

(Controlled measurement environment, results may vary)

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 excluded from 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

**Timelines and deliveries are subject to change.

Thank You!

- Please complete your session evaluations!

Session 17635:

IBM Java 8 and z13 - Hardware and Software Co-Design at Its Finest

- www.share.org/Orlando-Eval

Iris Baron

Email: ibaron@ca.ibm.com

Important references

- IBM Java for Linux website
  - http://www.ibm.com/developerworks/java/jdk/linux
- z/OS Java website
  - http://www.ibm.com/systems/z/os/zos/tools/java
- IBM SDK Java Technology Edition Documentation
  - http://www.ibm.com/developerworks/java/jdk/docs.html
- 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