Marcel Mitran, STSM, Chief Architect IBM JVM on System z
Ken Irwin, IBM, zOS Java L2 Service Support

IBM Java on System z

Session 12353

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" 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.

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.

IBM and Java

- **Java is critically important to IBM**
  - Fundamental infrastructure for IBM's software portfolio
  - Websphere, Lotus, Tivoli, Rational, Information Management (IM)
- **IBM is investing strategically for Java in virtual machines**
  - As of Java 5.0, single JVM support
    - JME, JSE, JEE
  - New technology base (J9/TR Compiler) on which to deliver improved performance, reliability, serviceability
- **IBM also invests in, and supports public innovation in Java**
  - OpenJDK, Eclipse, Apache (XML, Aries, Derby, Harmony, Tuscany, Hadoop …)
  - Broad participation in relevant open standards (JCP, OSGi)

Java Road Map

Language Updates

Java 5.0

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

Java 6.0

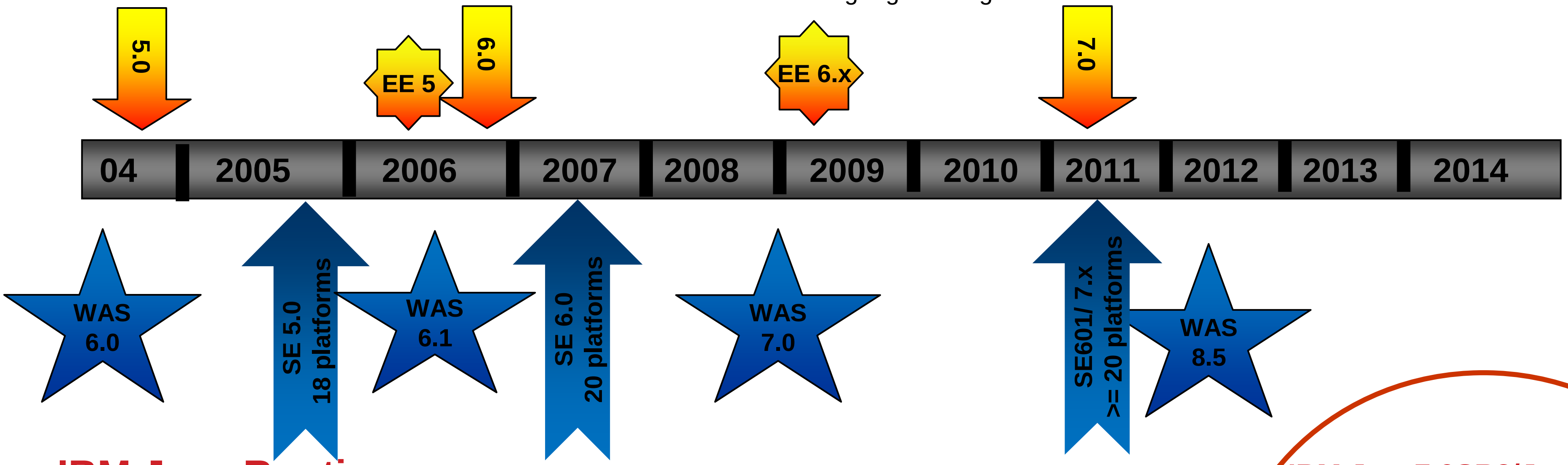
- Performance Improvements
- Client WebServices Support

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 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/Java7.0 (J9 R26)

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

IBM Java7.0SR3/Java.Vnext**

- Improvements in
  - Performance
  - GC Technology
- zEC12™ Exploitation
  - Transactional Execution
  - Runtime Instrumentation
  - Flash 1Meg pageable LPs
  - 2G large pages
  - Hints/traps
- Data Access Accelerator
- **Cloud:** Multi-tenancy/Virtualization

**Timelines and deliveries are subject to change.

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)
  - JCICS
  - IMS Java
  - DB2 Stored Procedures
- Batch oriented
  - WebSphere Compute Grid (WAS-CG)
    - *WAS/JEE runtime extensions*
- 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

** Alphaworks only, and hence currently un-supported

IBM Java Runtime Environment

- IBM's implementation of Java 5/6/7 are built with **IBM J9 Virtual Machine** and **IBM Testarossa JIT Compiler** technology
  - 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 System z exploitation – zEC12/z196/z10/z9/z990 exploitation
  - Cost-effective for z - zAAP Ready!

- Millions of instances of J9/TR compiler

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 1MB large pages using flash

Better versatility of managing memory

New software hints/directives

Data usage intent improves cache management

Branch pre-load improves branch prediction

New trap instructions

Reduce over-head of implicit bounds/null checks

New 5.5 GHz 6-Core Processor Chip

Large caches to optimize data serving

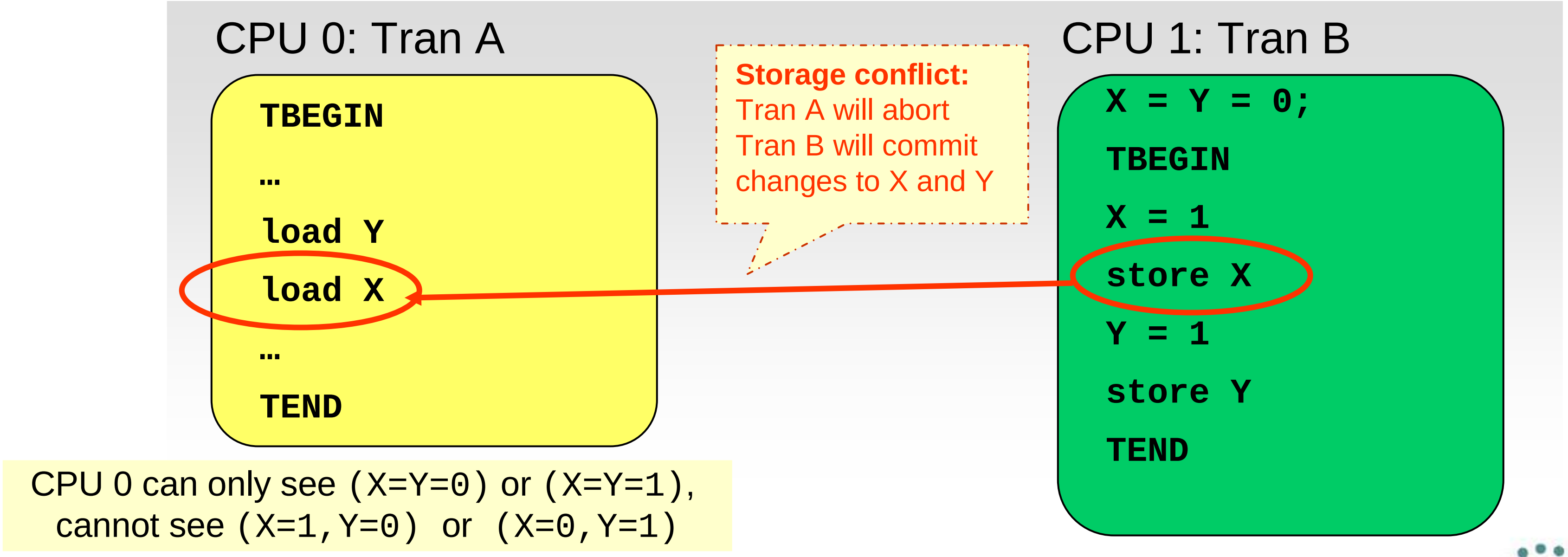
Second generation OOO design

*Up-to **60%** improvement in throughput amongst Java workloads measured with zEC12 and Java7SR3*

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

Hardware Transactional Memory (HTM)

- ***Allow lockless interlocked execution of a block of code called a 'transaction'***
  - **Transaction:** *Segment of code that appears to execute 'atomically' to other CPUs*
    - Other processors in the system will either see **all-or-none** of the storage up-dates of transaction
- ***How it works:***
  - TBEGIN instruction starts speculative execution of 'transaction'
  - Storage conflict is detected by hardware if another CPU writes to storage used by the transaction
  - Conflict triggers hardware to roll-back state (storage and registers)
    - transaction can be re-tried, or
    - a fall-back software path that performs locking can be used to guarantee forward progress
  - Changes made by transaction become visible to other CPUs after TEND



CPU 0: Tran A

```
TBEGIN
…
load Y
load X
…
TEND
```

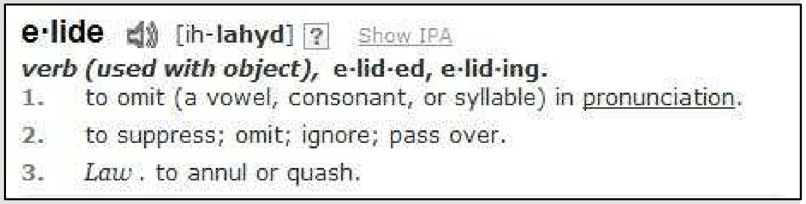
Storage conflict:
Tran A will abort
Tran B will commit
changes to X and Y

CPU 1: Tran B

```
X = Y = 0;
TBEGIN
X = 1
store X
Y = 1
store Y
TEND
```

CPU 0 can only see `(X=Y=0)` or `(X=Y=1)`,
cannot see `(X=1,Y=0)` or `(X=0,Y=1)`

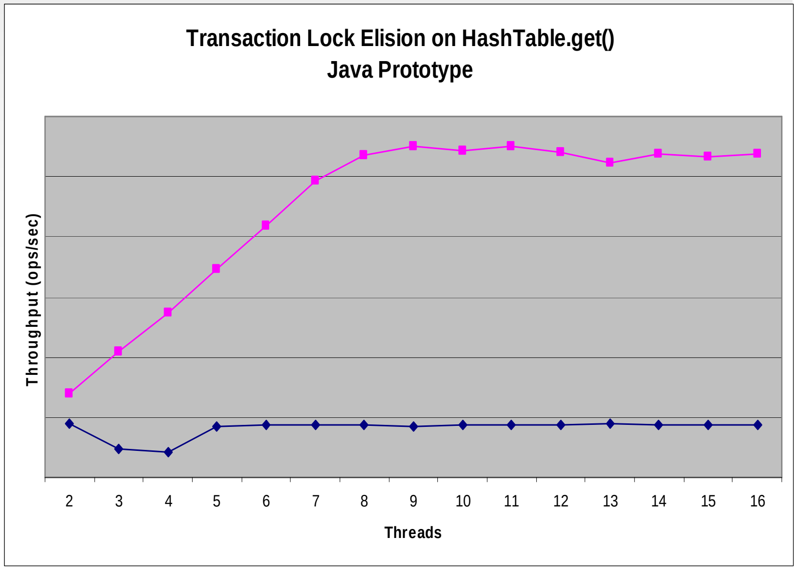
HTM Example: Transactional Lock Elision (TLE)



e·lide [ih-lahyd] Show IPA

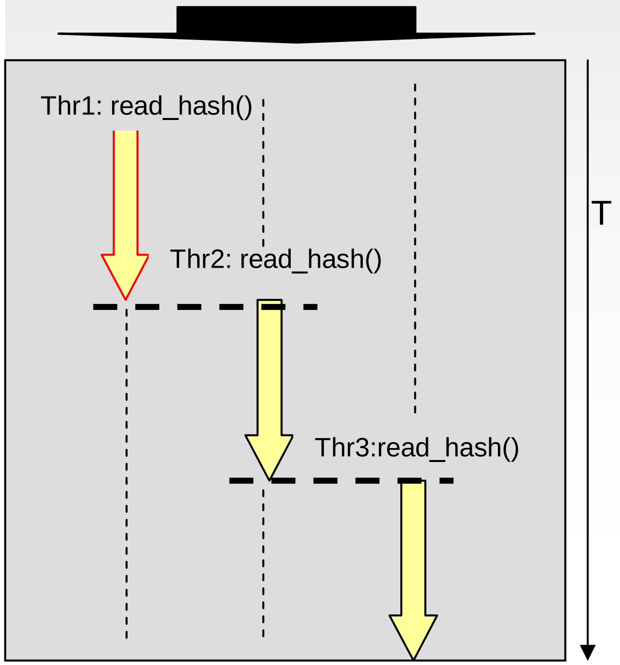
verb (used with object), **e·lid·ed, e·lid·ing.**

1. to omit (a vowel, consonant, or syllable) in pronunciation.
2. to suppress; omit; ignore; pass over.
3. *Law* . to annul or quash.



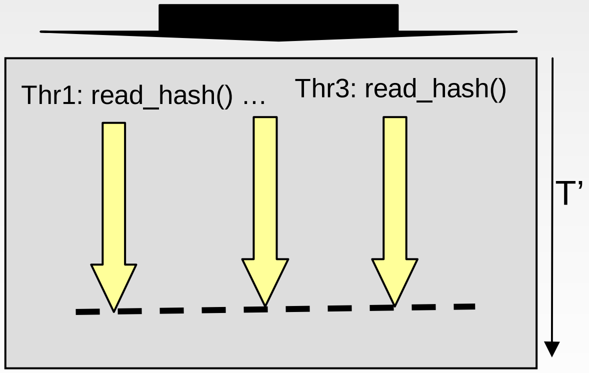
Threads must serialize despite only reading… just in-case a writer updates the hash

```
read_hash(key) {
  Wait_for_lock();
  read(hash, key);
  Release_lock();
}
```



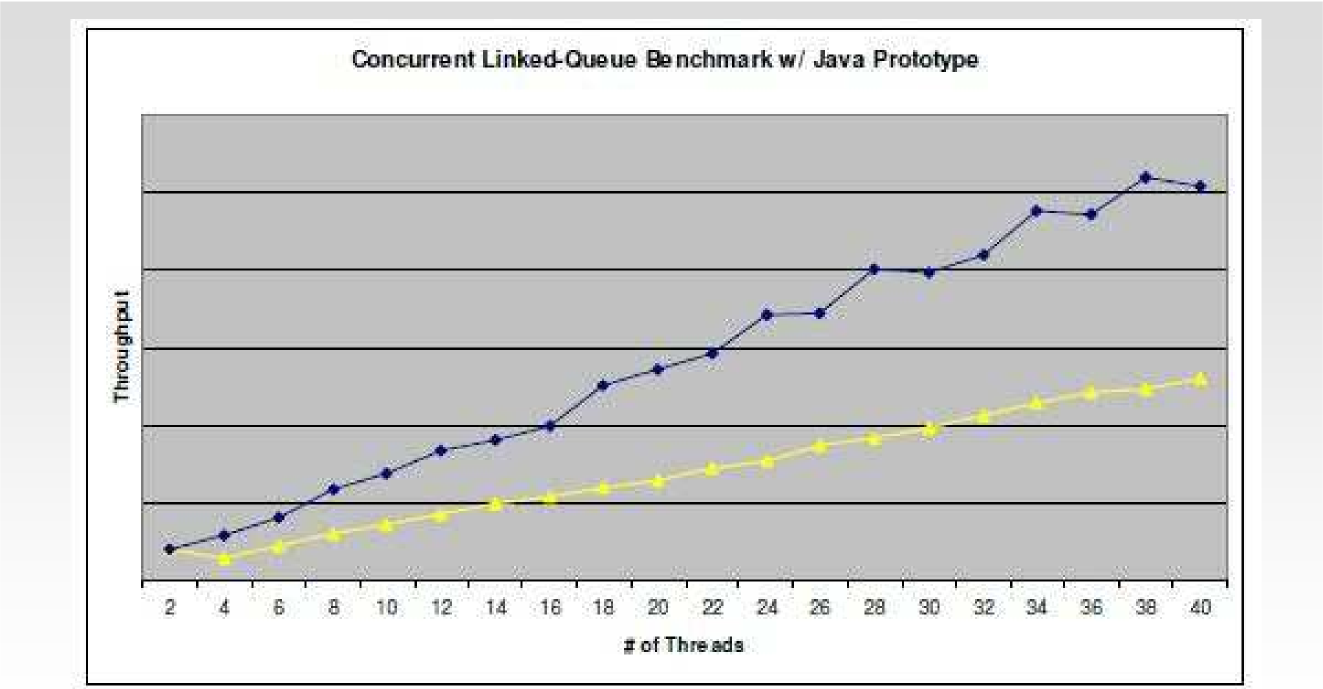
Lock elision allows readers to execute in parallel, and safely back-out should a writer update hash

```
read_hash(key)
  TRANSACTION_BEGIN
  read hash.lock;
  BRNE serialize_on_hash_lock
  read (hash, key);
  TRANSACTION_END
```



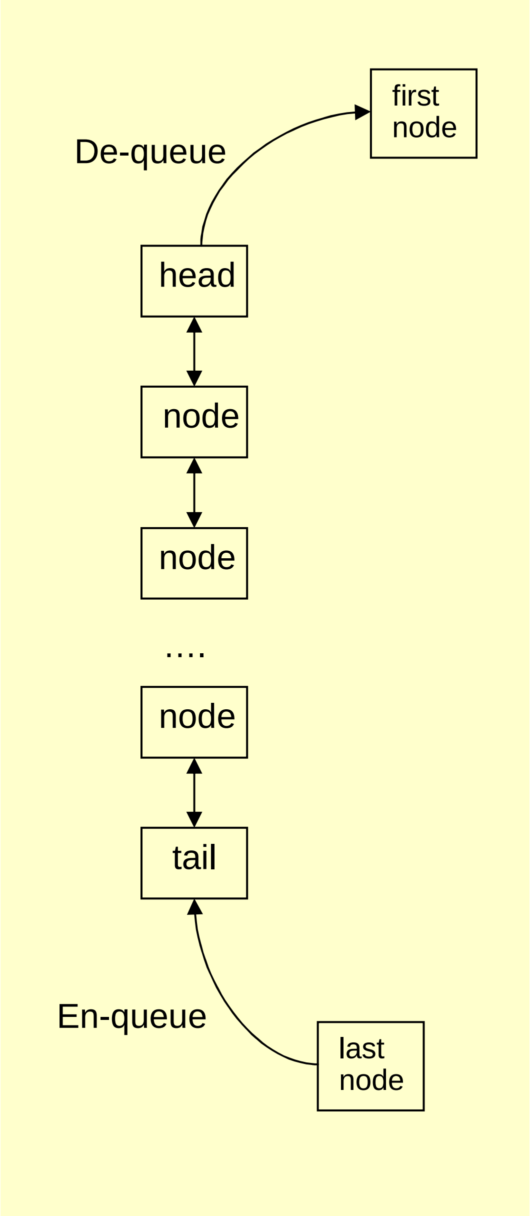
Transactional Execution: Concurrent Linked Queue

- ***~2x improved scalability of juc.ConcurrentLinkedQueue***
- ***Unbound Thread-Safe LinkedQueue***
  - First-in-first-out (FIFO)
    - Insert elements into tail (en-queue)
    - Poll elements from head (de-queue)
  - No explicit locking required
- ***Example usage: a multi-threaded work queue***
  - Tasks are inserted into a concurrent linked queue as multiple worker threads poll work from it concurrently



🟦 **New TX-base implementation**

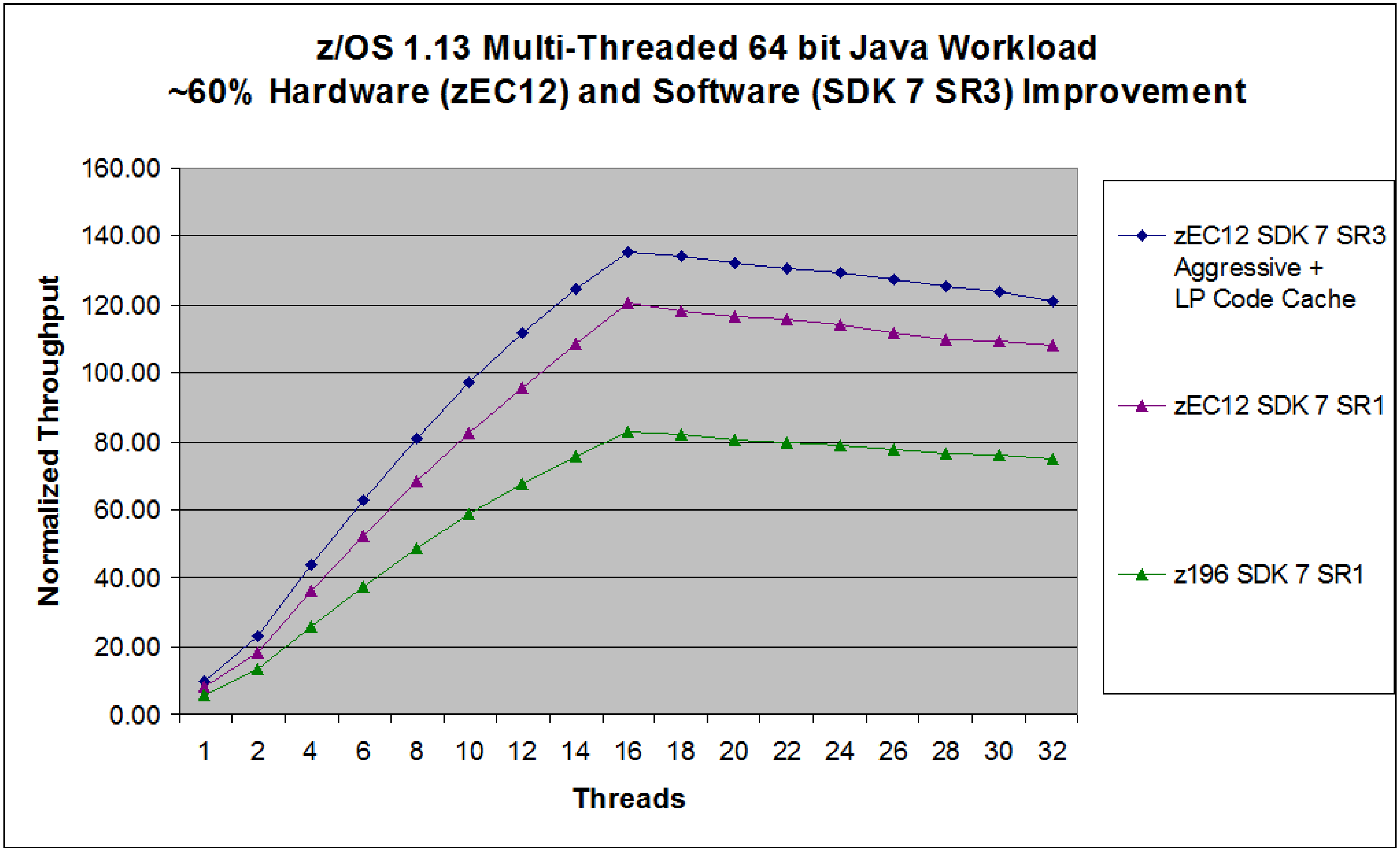
🟨 **Traditional CAS-base implementation**



(Controlled measurement environment, results may vary)

z/OS Java SDK 7:16-Way Performance

64-bit Java Multi-threaded Benchmark on 16-Way



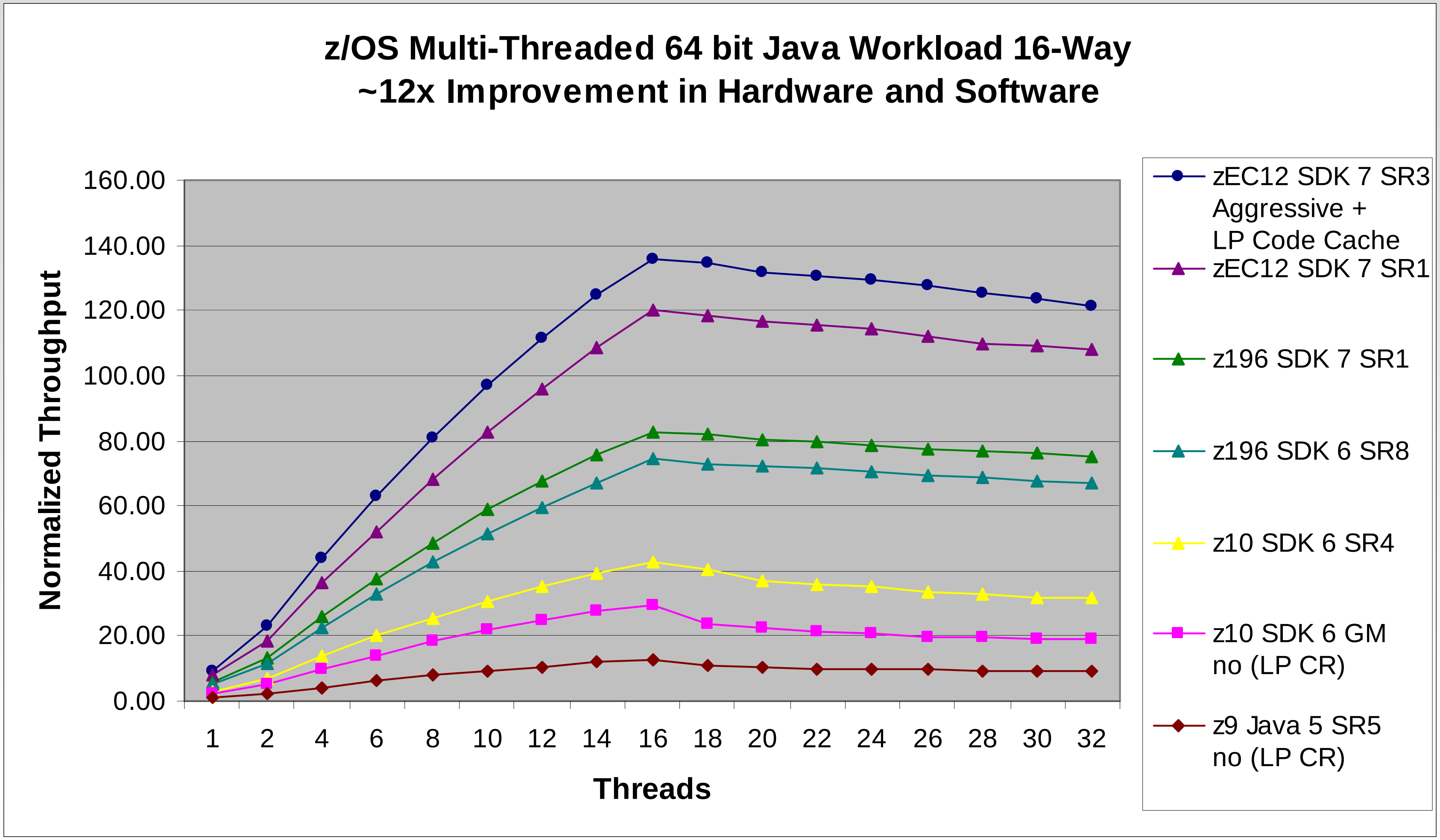
Aggregate 60% improvement from zEC12 and Java7SR3

- **zEC12 offers a ~45% improvement over z196 running the Java Multi-Threaded Benchmark**
- **Java7SR3 offers an additional ~13% improvement** **(-Xaggressive + Flash Express pageable 1Meg large pages)**

(Controlled measurement environment, results may vary)

z/OS Java SDK 7: 16-Way Performance

Aggregate HW and SDK Improvement z9 Java 5 SR5 to zEC12 Java7SR3



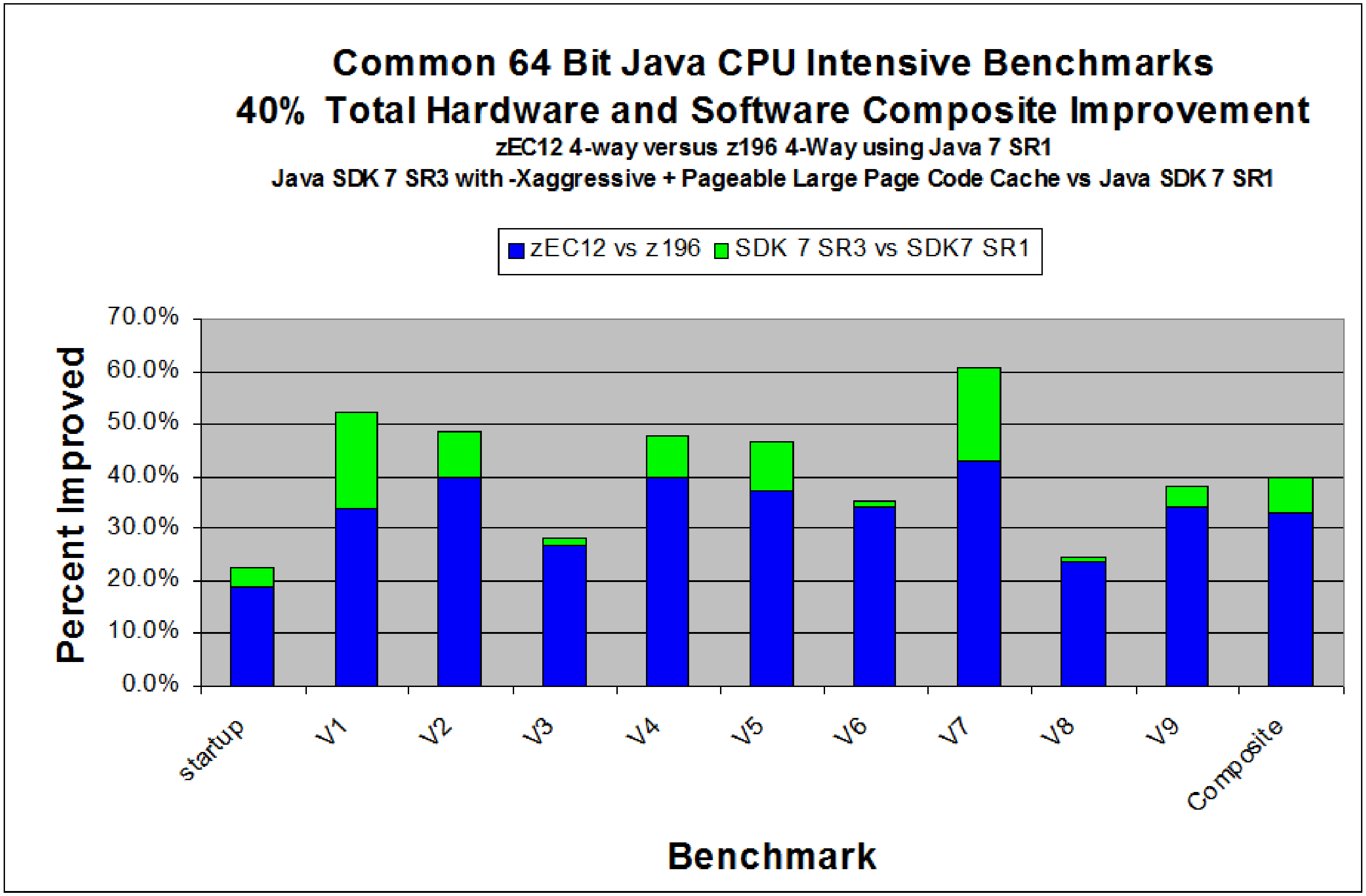
~12x aggregate hardware and software improvement comparing Java5SR5 on z9 to Java7SR3 on zEC12

LP=Large Pages for Java heap CR= Java compressed references

Java7SR3 using -Xaggressive + Flash Express pageable 1Meg large pages

(Controlled measurement environment, results may vary)

z/OS Java SDK 7: CPU-Intensive Benchmark



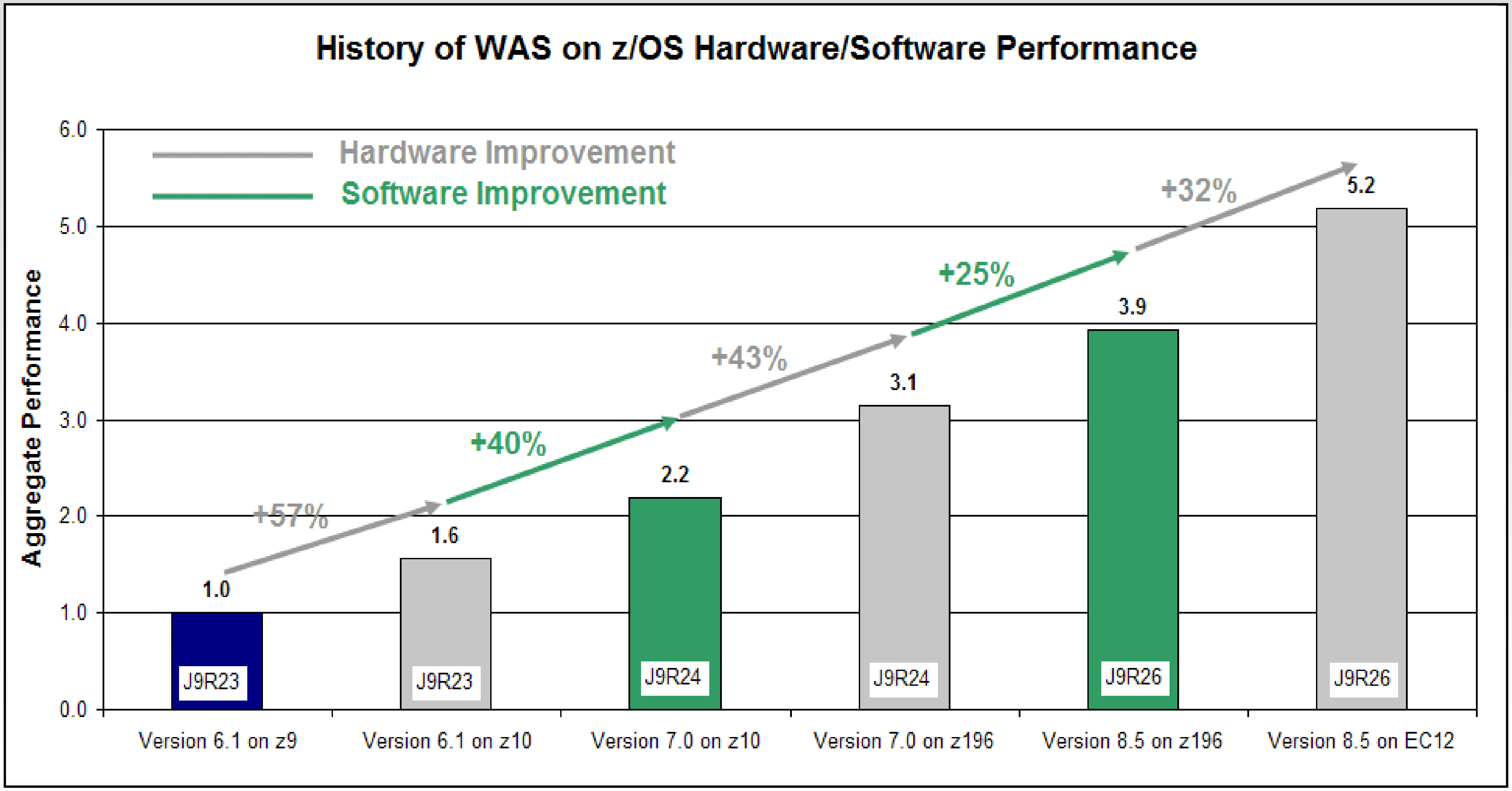
zEC12 and Java7SR3 offer a ~40% composite improvement over z196 running the CPU Intensive benchmark

- **zEC12 offers a ~33% improvement over z196 running the CPU-Intensive Benchmarks**
- **Java7SR3 offers an additional ~5% improvement (-Xaggressive + Flash Express pageable 1Meg large pages)**

(Controlled measurement environment, results may vary)

WAS on z/OS –

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

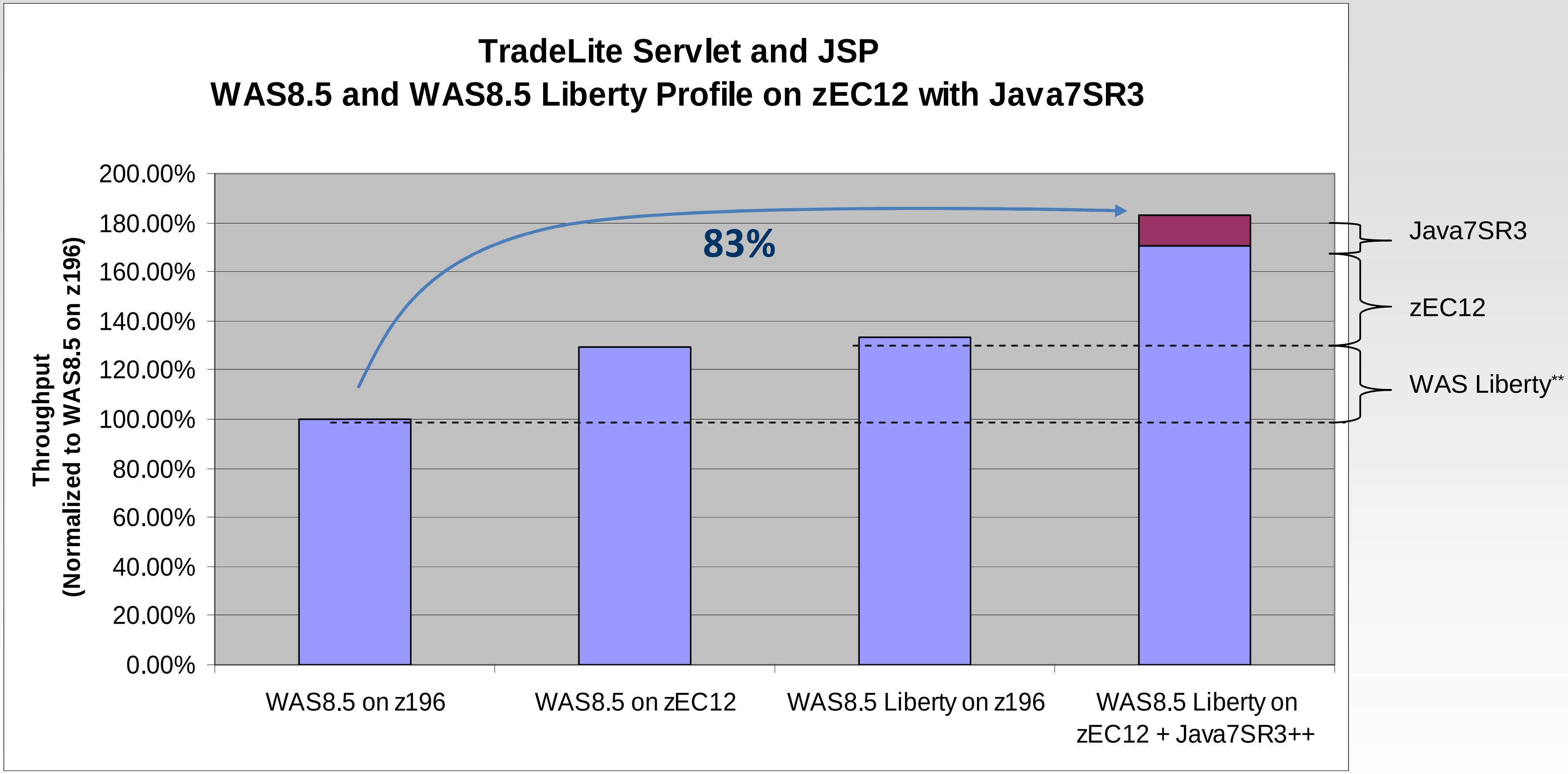


~5x aggregate hardware and software improvement comparing WAS 6.1 Java5 on z9 to WAS 8.5 Java7SR1 on zEC12

(Controlled measurement environment, results may vary)

WAS on z/OS

Servlets and JSPs with the Liberty Profile



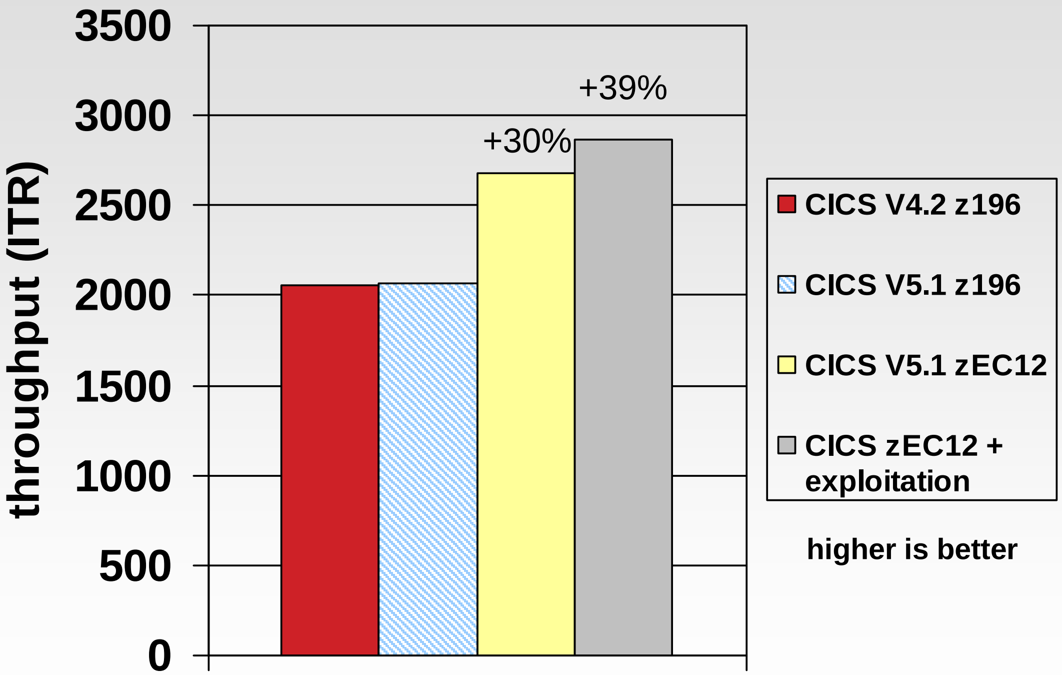
- WAS8.5 Liberty on zEC12 using Java7SR3 vs WAS8.5 on z196 running TradeLite demonstrates a 83% improvement to Servlet and JSP throughput.
- WAS8.5 Liberty offers up to 5x start-up time reduction vs. WAS8.5 (<5 seconds)
- WAS8.5 Liberty offers reduced real-storage requirements up to 81% vs. WAS8.5 (80M versus 420M)

(Controlled measurement environment, results may vary)

JCICS with Java7SR3 and zEC12

More than a third of CICS customers are using JCICS

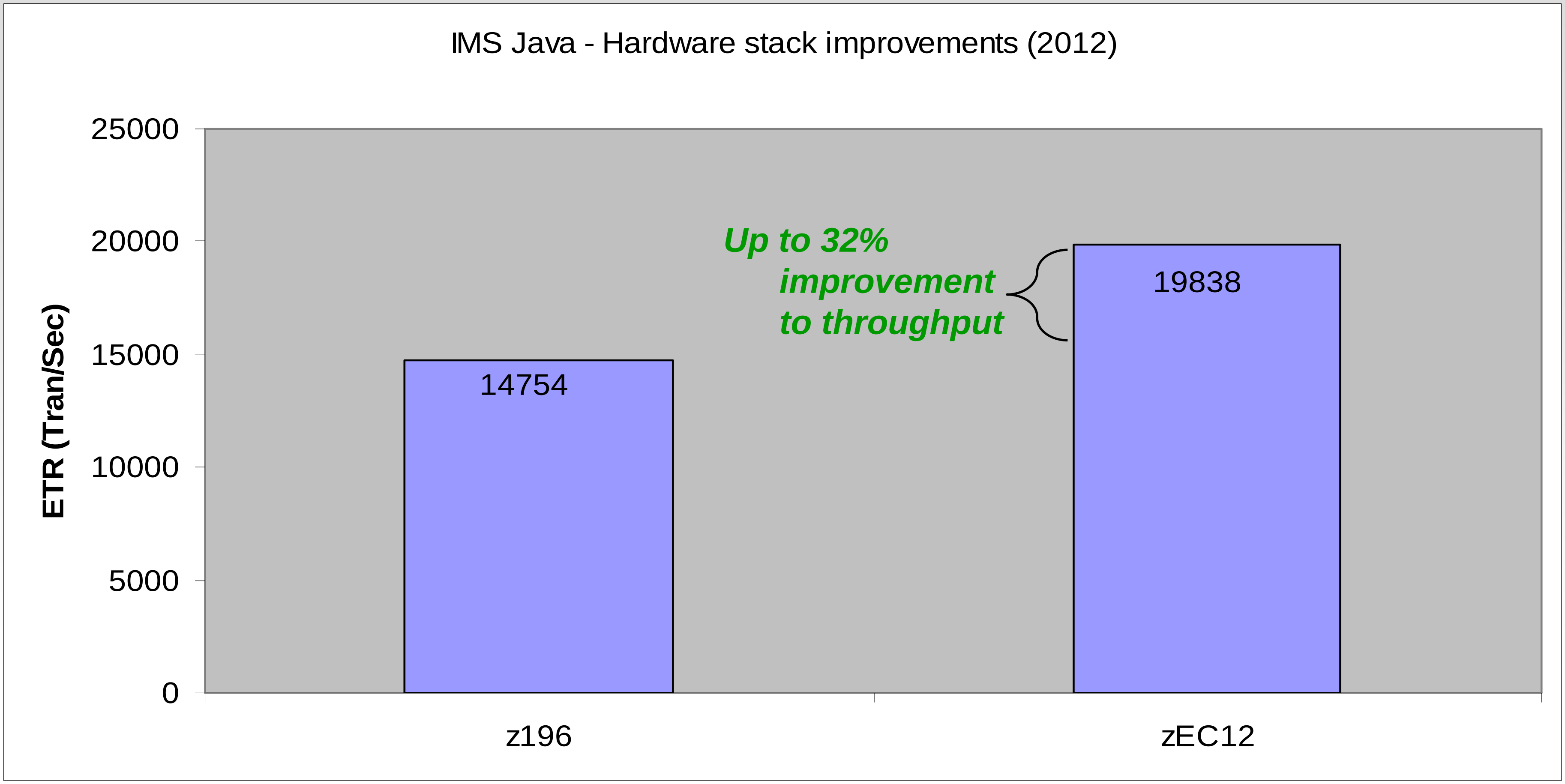
- Using complex Java workload – Axis2 webservice
- Equivalent throughput using CICS V5.1 on z196 compared to CICS V4.2
- 30% improvement in throughput using CICS V5.1 on zEC12 compared to CICS V4.2 on z196
- 39% improvement in throughput using CICS V5.1 with Java 7 zEC12 exploitation compared to CICS V4.2 on z196



higher is better

IMS JMP Region with Java7SR3 and zEC12

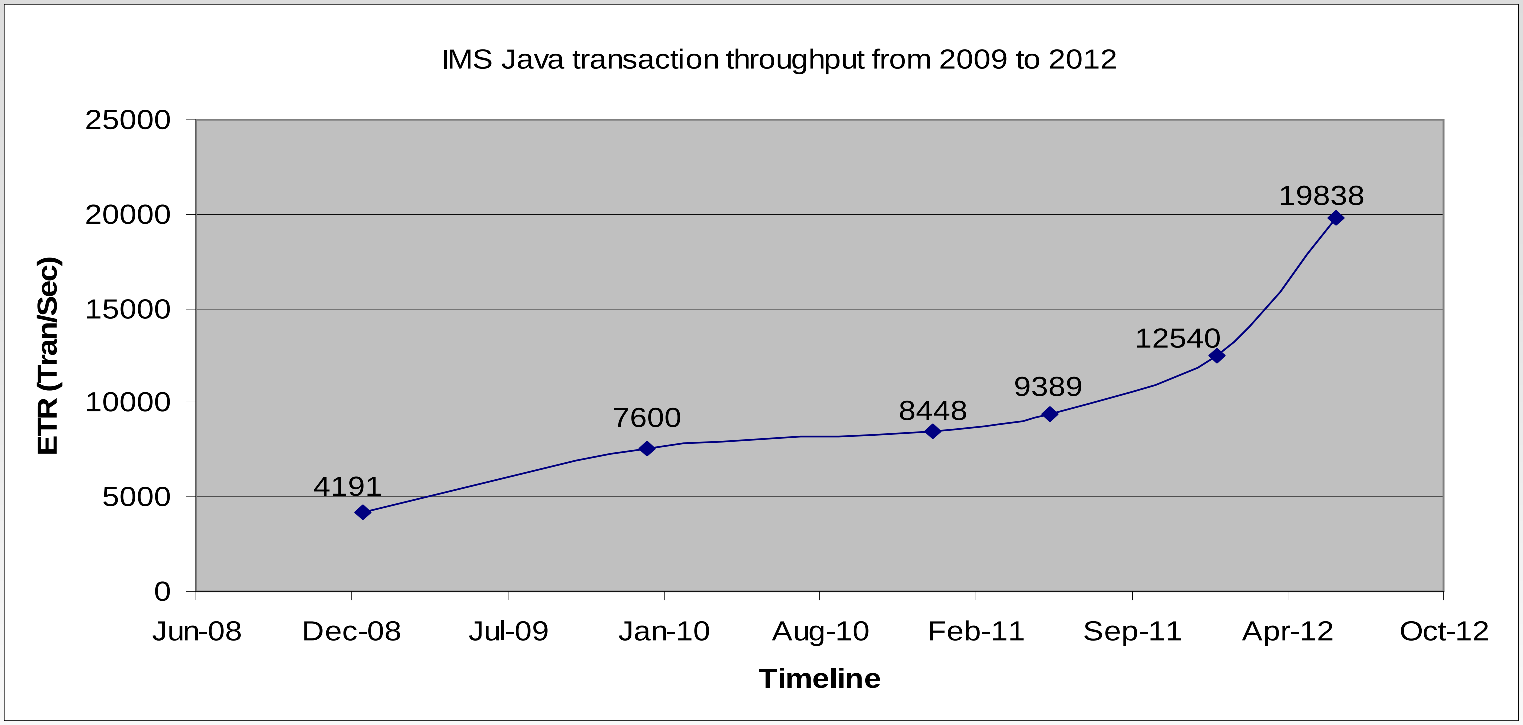
More than 20% of top IMS customers are using IMS-Java



(Controlled measurement environment, results may vary)

IMS JMP region performance

Aggregate SDK, software and hardware improvements



Over 4x aggregate throughput improvement from 2009 to 2012 due to the following enhancements

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

(Controlled measurement environment, results may vary)

Java8-Beta Program

- **Provides Java SE 8 compatibility, while exploiting the unique capabilities of IBM platforms to achieve performance and usability improvements**
  - To provide early technology access during the development cycle
  - To assist Java 8 in satisfying customer requirements
  - To provide feedback to IBM

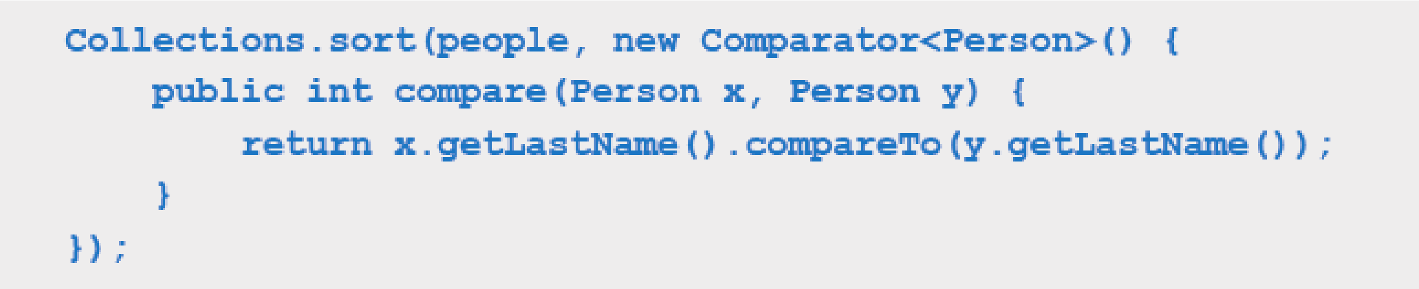
- **New in IBM SDK, Java Technology Edition, Version 8:**
  - Compatibility with the new Java SE 8
  - Leveraging new IBM hardware (e.g. IBM zEnterprise EC12)
  - Improved performance for workload optimized runtimes, which delivers better application throughput without changes to application code
  - Enhanced support for Cloud & Multi-tenancy environments
  - Improved efficiency of manipulating native data records/types directly from Java code

- **Managed and Open Beta**
  - http://www.ibm.com/developerworks/java/jdk/beta/index.html

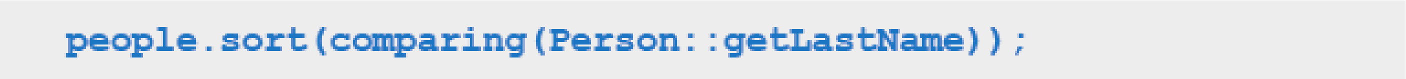
Java8: Language Innovation -- Lambdas

New syntax to allow concise code snippets and expression

- Useful for sending code to java.lang.concurrent
- On the path to enabling more parallelisms



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



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

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

Java8: Data Access Accelerator

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[]);
}
```

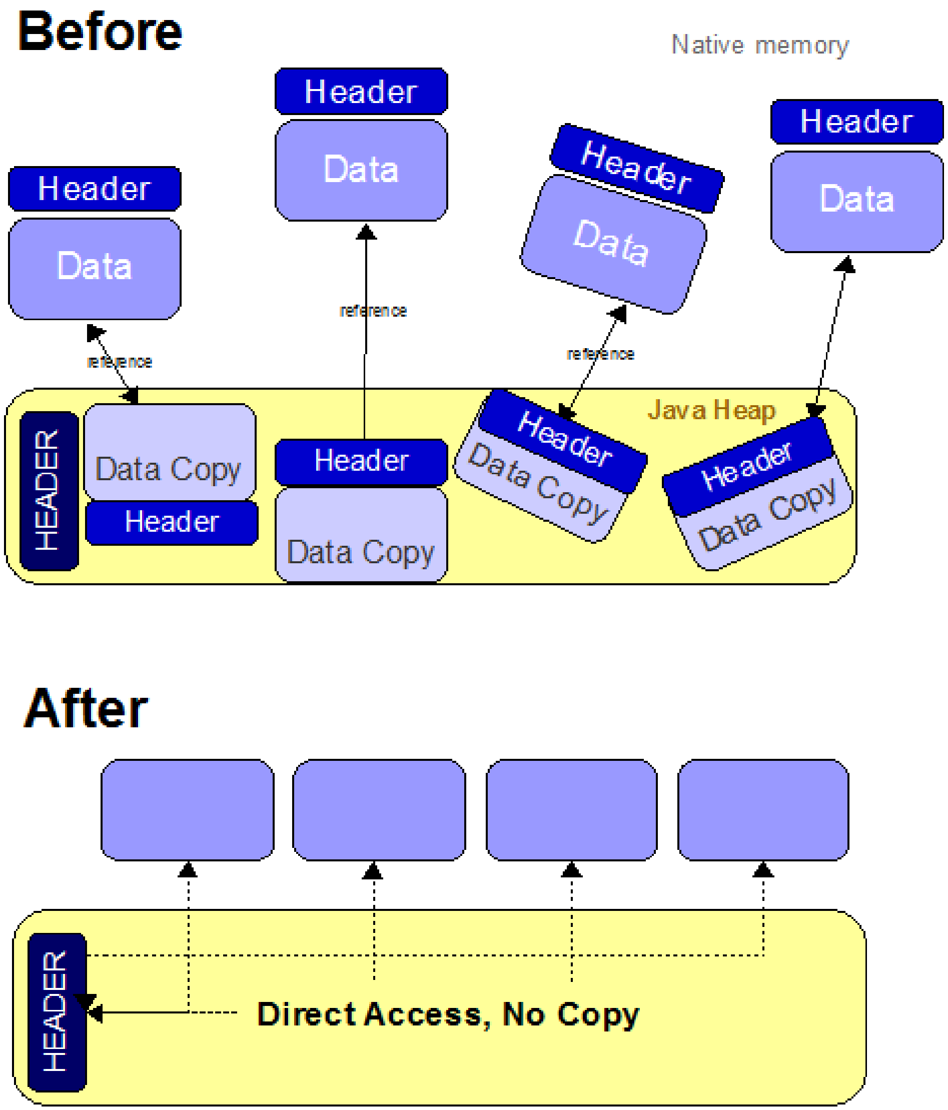
Looking Ahead: PackedObjects with IBM Java

PackedObjects

Experimental feature in the IBM JVM. Introduces a new Java type that implements an explicit object model which tightly packs fields allowing for natural and efficient direct mapping of structured data.

Goals

- Allow for explicit source-level representation of structured data in Java
- Improve serialization and I/O performance
- Allow direct access to "native" (off-heap) data

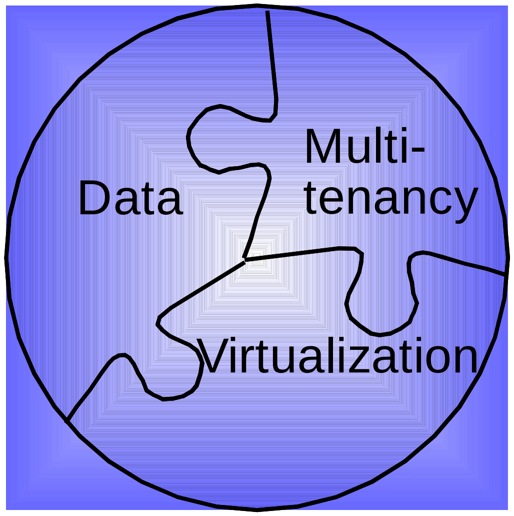


http://www.slideshare.net/mmitran/ibm-java-packed-objects-mmit-20121120
http://duimovich.blogspot.ca/2012/11/packed-objects-in-java.html

Timelines and deliveries are subject to change.

Looking Ahead: Cloud with IBM Java

- **Multi-tenancy support will allow multiple applications to run in a single shared JVM for high-density deployments.**
  - *Win*: Footprint reduction enabled by sharing runtime and JVM artifacts while enforcing resource consumption quotas
  - *Platform Coverage*: 64-bit, balanced GC policy only
  - *Ergonomics*: Single new command-line flag (**-Xmt** = **m**ulti **t**enancy)



- **Runtime Adjustable Heap Size (-Xsoftmx)**
  - JMX beans allow for dynamically adjusting heap size
  - Allows users to take advantage of hot-add of memory

- **Hypervisor, Virtual Guest, and Extended-OS JMX Beans**
  - Allows applications to detect and identify the installed hypervisor and query attributes of LPAR
  - Provides richer access to operating system performance statistics

Timelines and deliveries are subject to change.

Looking Ahead: Cloud

- **Multi-tenancy support will allow multiple applications to run in a single shared JVM for high-density deployments.**
  - *Win*: Footprint reduction enabled by sharing runtime and JVM artifacts while enforcing resource consumption quotas
  - *Platform Coverage*: 64-bit, balanced GC policy only
  - *Ergonomics*: Single new command-line flag (**-Xmt** = **m**ulti **t**enancy)

- **Runtime Adjustable Heap Size (-Xsoftmx)**
  - JMX beans allow for dynamically adjusting heap size
  - Allows users to take advantage of hot-add of memory

- **Hypervisor, Virtual Guest, and Extended-OS JMX Beans**
  - Allows applications to detect and identify the installed hypervisor and query attributes of LPAR
  - Provides richer access to operating system performance statistics

Timelines and deliveries are subject to change.

Thank
You

Liberty and traditional profile capabilities

There are functional differences between traditional WAS and the Liberty profile – Liberty provides a useful subset of traditional WAS

Liberty Profile

Bean validation
Blueprint
Java API for RESTful Web Services
Java Database Connectivity (JDBC)
Java Naming and Directory Interface (JNDI)
Java Persistence API (JPA)
Java Server Faces (JSF)
Java Server Pages (JSP)
JMX
Monitoring
OSGi JPA
Remote connector
Secure Sockets Layer (SSL)
Security
Servlet
Session Persistence
Transaction
Web application bundle (WAB)
z/OS Security (SAF)
z/OS Transactions (RRS)
z/OS Workload Management

Traditional WAS Profile

Everything Liberty has…

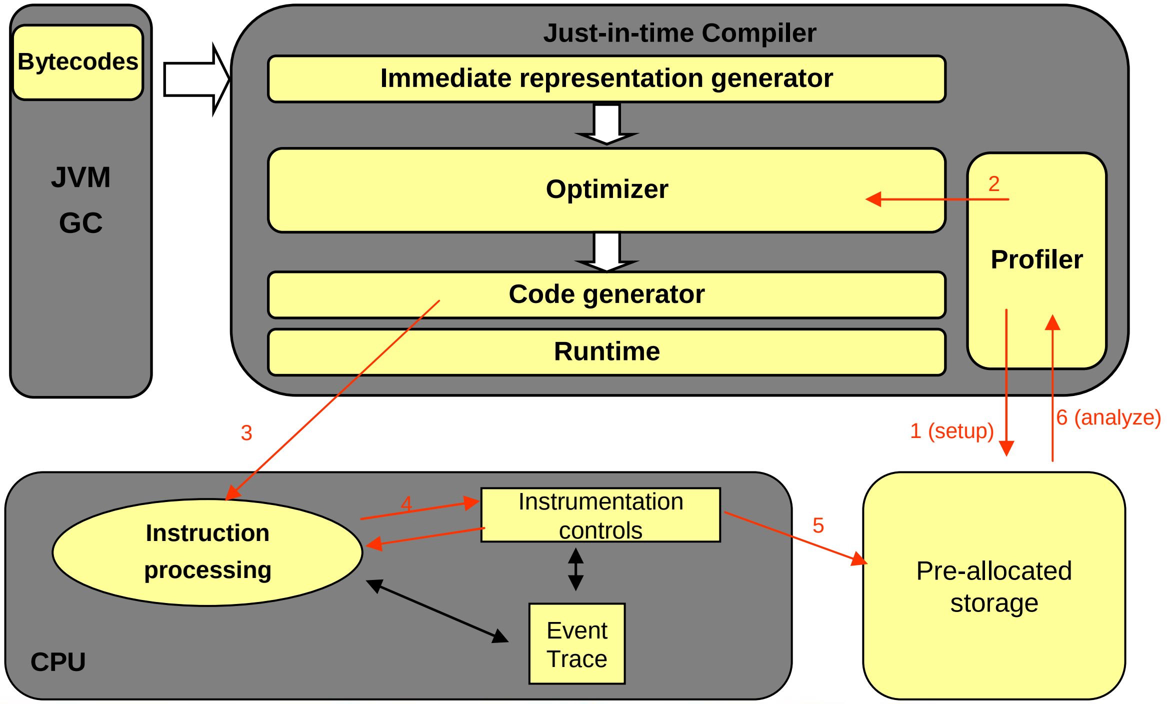
+

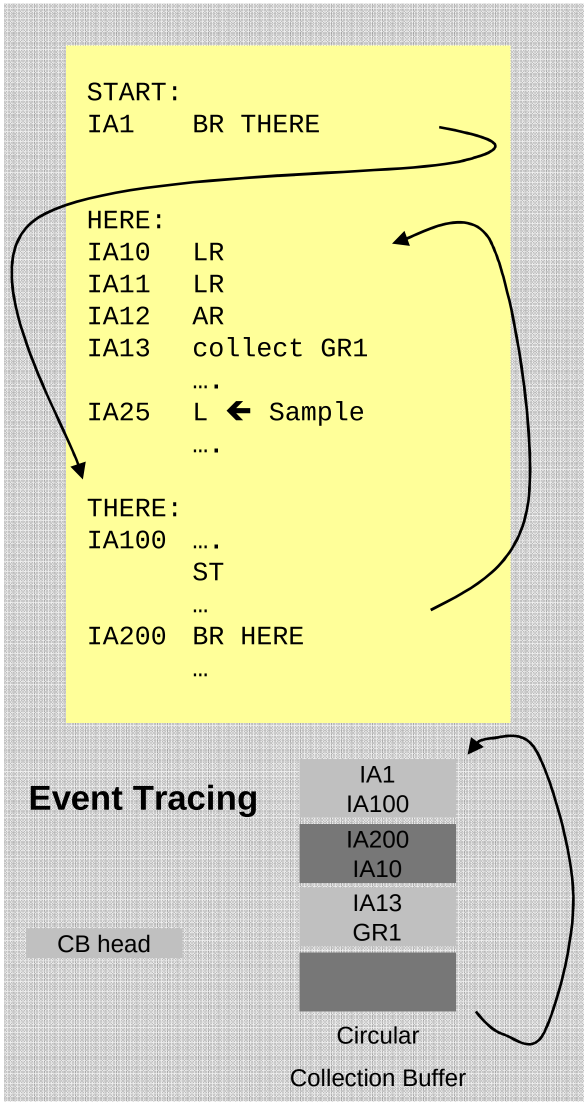
Enterprise Java Beans (EJBs)
Messaging (JMS)
Web Services
Service Component Arch (SCA)
Java Connector Architecture (JCA)
Clustering
WebSphere Optimized Local Adapters
Administrative Console
WSADMIN scripting
Multi-JVM Server Model

And much more …

Runtime Instrumentation

- ***Low overhead profiling with hardware support***
  - Instruction samples by time, count or explicit marking
- ***Sample reports include hard-to-get information:***
  - Event traces, e.g. taken branch trace
  - “costly” events of interest, e.g. cache miss information
  - GR value profiling
- ***Enables better “self-tuning” opportunities***





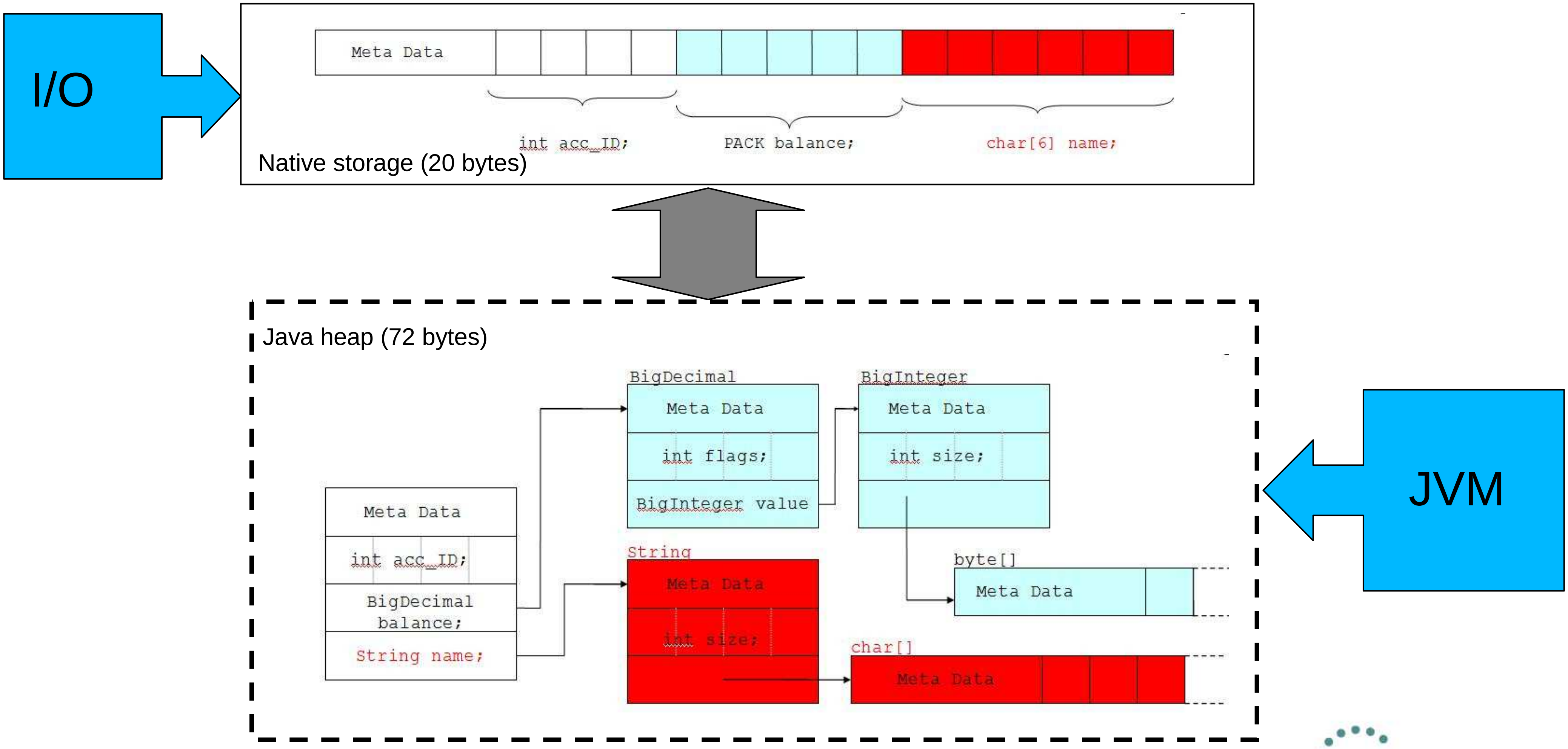
```
START:
IA1    BR THERE

HERE:
IA10   LR
IA11   LR
IA12   AR
IA13   collect GR1
       ….
IA25   L ← Sample
       ….

THERE:
IA100  ….
       ST
       …
IA200  BR HERE
       …
```

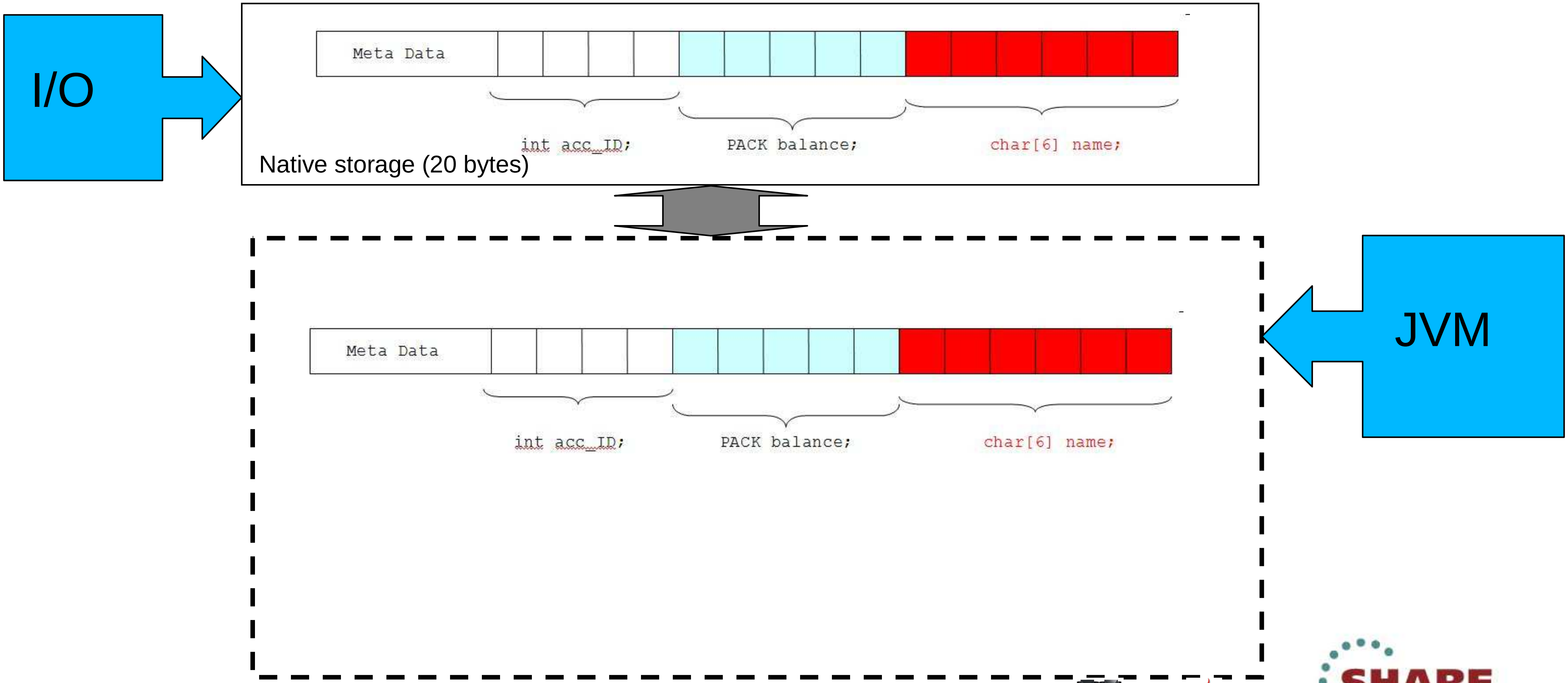
Speak to me in 'Java'

- Java only speaks 'Java'…
  - Data typically must be copied/re-formatted onto/off Java heap
  - Costly in path-length and footprint



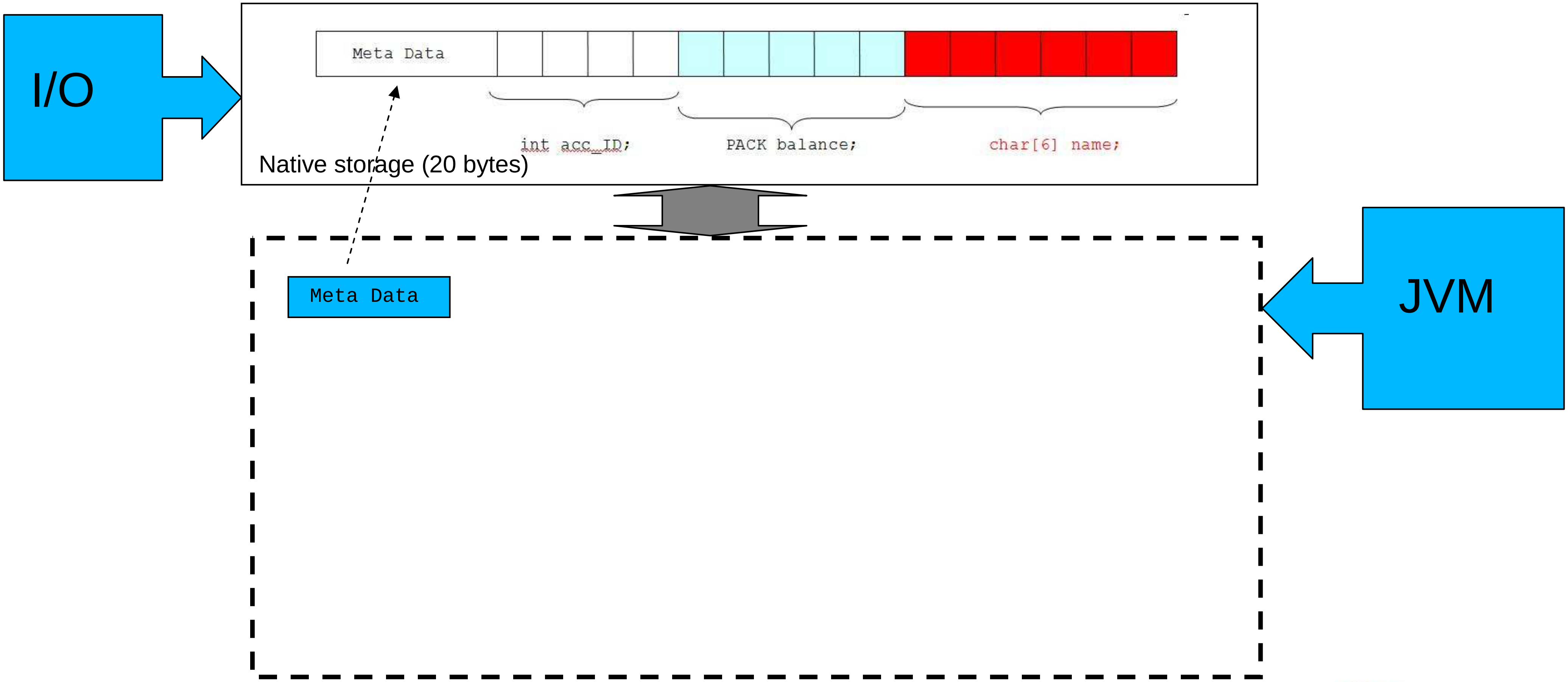
On-Heap PackedObject

- Allows controlled layout of storage of data structures on the Java heap
  - Reduces footprint of data on Java heap
  - No (de)serialization required

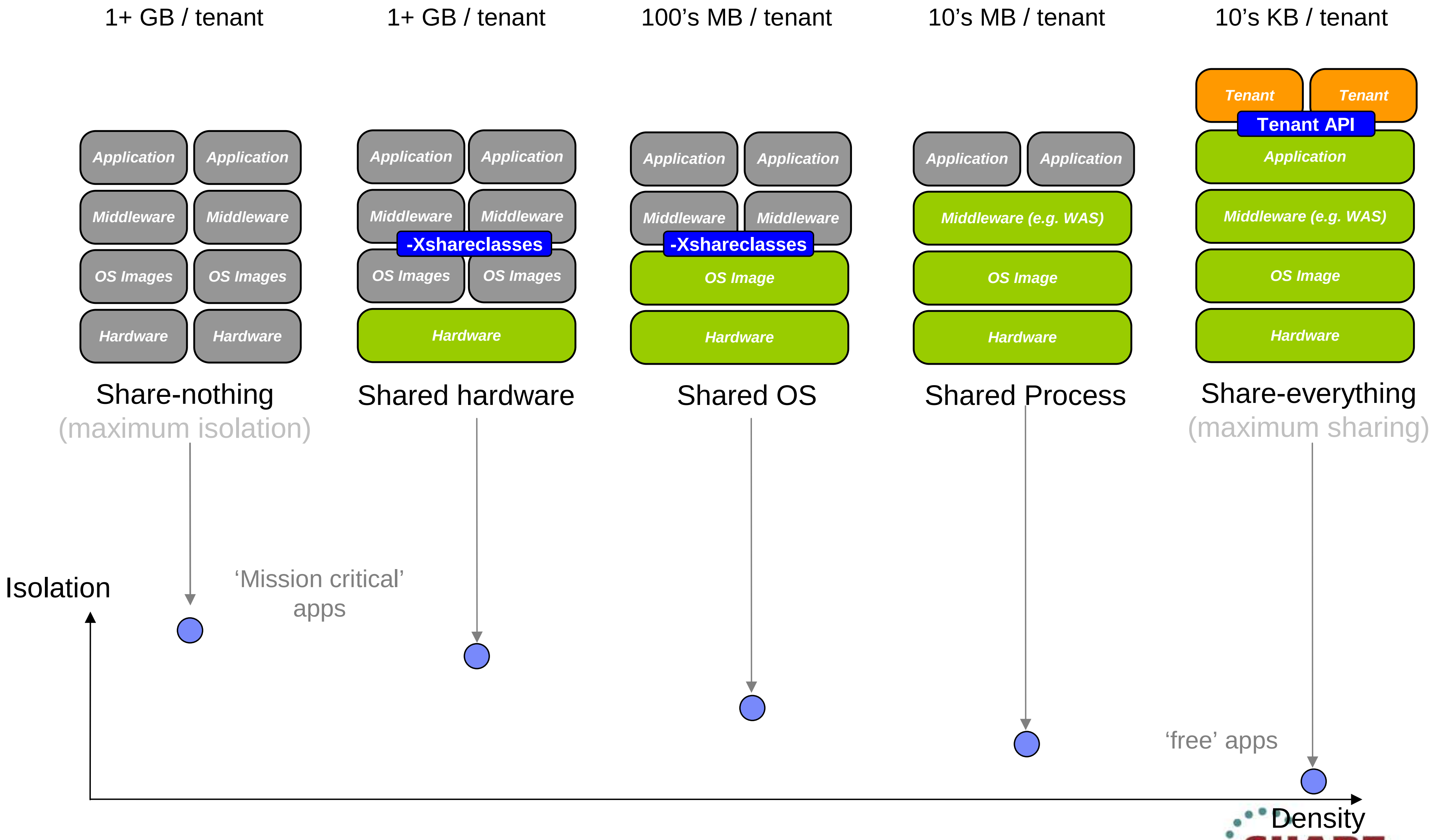


Off-Heap PackedObject

- Enable Java to talk directly to the native data structure
  - Avoid overhead of data copy onto/off Java heap
  - No (de)serialization required



Multitenancy: Isolation and Density



1+ GB / tenant

Application | Application

Middleware | Middleware

OS Images | OS Images

Hardware | Hardware

Share-nothing

(maximum isolation)

1+ GB / tenant

Application | Application

Middleware | Middleware

-Xshareclasses

OS Images | OS Images

Hardware

Shared hardware

100's MB / tenant

Application | Application

Middleware | Middleware

-Xshareclasses

OS Image

Hardware

Shared OS

10's MB / tenant

Application | Application

Middleware (e.g. WAS)

OS Image

Hardware

Shared Process

10's KB / tenant

Tenant | Tenant

Tenant API

Application

Middleware (e.g. WAS)

OS Image

Hardware

Share-everything

(maximum sharing)

Isolation

'Mission critical' apps

'free' apps

Density

Timelines and deliveries are subject to change.