What's New in IBM Java 8 SE?

Tim Ellison – Hursley labs.

About the Speaker

- Technical staff based in the Java Technology Centre, Hursley UK
- Working on various runtime technologies for >20 years
- Experience of open source communities
- Currently focused on class library design and delivery
- Overall technical lead for IBM Java 8 SE

tim_ellison@uk.ibm.com

@tpellison

Agenda

Over the next ~60 minutes, I hope to...

- Introduce you to IBM Java SE
  - Explain how and why IBM Java is different to other Java runtime offerings
  - Outline our goals, and strategy to achieve them in Java 8
- Briefly describe the standard Java 8 features
  - Show how Java 8 SE was defined
  - Give you an introduction to the key new features in standard Java 8
- Look at the new IBM features in a bit more detail
  - Show you how IBM Java is addressing your problems
  - Share our ideas and opportunities for shaping the future of Java
- Answer your questions

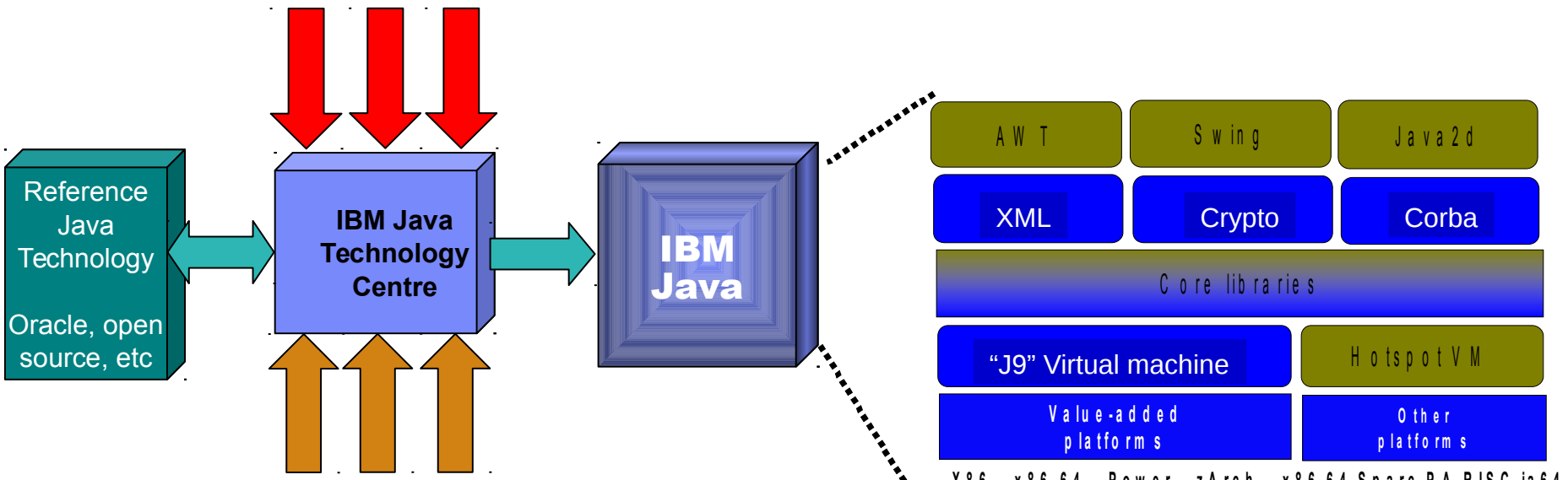
Introduction to

IBM Java SE

IBM's approach to Java SE technology

Quality Engineering
Performance
Security
Reliability
Scalability
Serviceability

- ✓ ***World class service and support***
- ✓ ***Available on more platforms than any other vendor***
- ✓ ***Optimised for IBM middleware and customer scenarios***
- ✓ ***Deep integration with hardware and software innovation***



Production Requirements
IBM Software Group
IBM *hardware*
ISVs
IBM Clients

IBM invests in Java technology to make it ready for the most demanding business applications

- **Performance**
  - Performance is key for all Java customers
  - IBM has decades of experience in performance engineering and cares deeply about creating high performance, scalable solutions
  - We leverage this experience and close relationships with hardware, operating system and middleware designers to drive best in class performance across our supported platforms
- **Security**
  - IBM is a key contributor to Java and XML security standards
  - We offer FIPS certified JCE and JSSE providers and broad hardware crypto support
- **Reliability**
  - Java is used in mission-critical applications
  - IBM has carefully redesigned the JVM, the engine at the heart of the Java runtime, for high reliability
- **Serviceability**
  - In the event of failure, it is critical that problems can be found and isolated quickly
  - IBM focuses on trace and logging capabilities, first failure data capture, debugging and performance interfaces and tools to ensure rapid problem resolution
- **Scalability**
  - Highly configurable runtimes for a variety of application profiles
  - Pluggable interfaces with different implementations to match target requirements
  - New class library technology available underpinning appropriate specification APIs

A Bit of History: Java SE

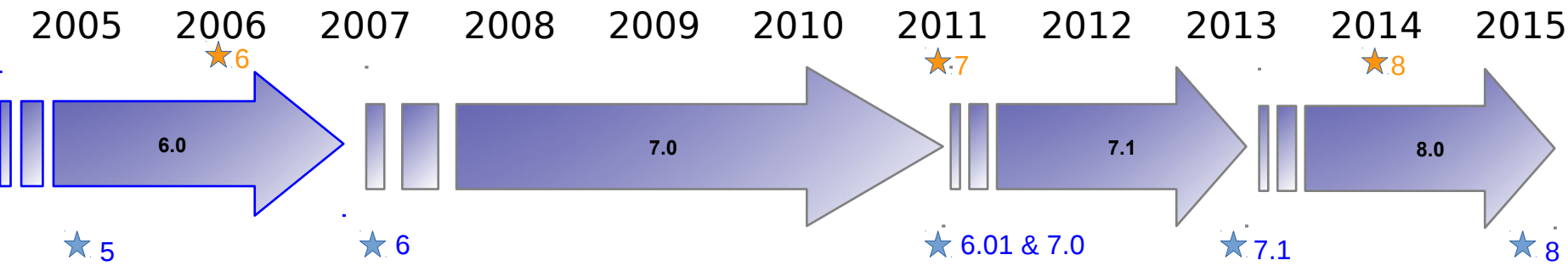
Standard Java Features

Java 5.0
- New Language features
  - Autoboxing
  - Enumerated types
  - Generics
  - Meta Data

Java 6.0
- Performance improvements
- Improved UI
- Client WebServices Support
- Jconsole monitoring
- Collection framework enhancements

Java 7.0
- Small language changes
- Improved IO APIs (NIO2)
- Invoke Dynamic
- Concurrency framework

Java 8.0
- Lambdas
- Date and time
- Type annotations
- Profiles



Additional IBM Java Features

IBM Java 5.0
- Improved performance
  - Generational Garbage Collector
  - Shared classes support
  - New JIT technology
- First Failure Data Capture
- Configurable Trace
- Full Speed Debug
- Hot Code Replace
- Common runtime technology
  - ME, SE, EE

IBM Java 6.0
- Improvements in
  - Platform coverage
  - Performance
  - Serviceability tooling
- New Functionality
  - IBM WebSphere Real-Time V1.0
  - z10 Exploitation
  - DFP exploitation for BigDecimal
  - Large Pages

IBM Java 6.0.1 & 7.0
- Improvements in
  - Start up performance
  - Throughput performance
  - New Balanced GC
  - New feature in serviceability tooling
  - Soft Realtime evaluation
  - Performance exploitation of POWER7
- z196™ Exploitation
  - OOO Pipeline
  - 70+ New Instructions
- JZOS/Security Enhancements

IBM Java 7.1
- 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

IBM Java SE platform coverage

IBM supported platforms	Linux	AIX	Windows	z/OS	Solaris	HP-UX
Intel 32-bit	X		X		X	
AMD 64-bit	X		X		X	
PowerPC 32-bit BE	X	X				
PowerPC 64-bit BE	X	X				
PowerPC 64-bit LE	X					
z System 31-bit	X			X		
z System 64-bit	X			X		
Itanium 32-bit						X
Itanium 64-bit						X
Sparc 32-bit					X	
Sparc 64-bit					X	

For full details of supported platforms visit `http://www.ibm.com/developerworks/java/jdk/docs.html`

IBM Java Service Schedule

- IBM continues to offer quarterly service releases and APAR deliveries of Java 7, 6 and 5.
  - Ensures security fixes will be delivered rapidly to the field across all platforms.

- Key dates
  - Java 5
    - GA 2005
    - went out of currency in September 2013.
    - only receives customer and security fixes.
    - goes out of service in September 2015 (zOS Sept 2013).
  - Java 6
    - GA 2007
    - will go out of currency in September 2015
    - receives platform & OS updates, as well as customer and security fixes.
    - goes out of service in September 2017 (zOS to be announced)
  - Java 7
    - GA 7.0 2011, GA 7.1 2013
    - will go out of currency in September 2017
    - receives enhancements, platform & OS updates, as well as customer and security fixes.
    - goes out of service in September 2019 (zOS to be announced)
  - Java 8
    - GA February 2015
    - Receives enhancements, platform & OS updates, as well as customer and security fixes.

ref: http://www.ibm.com/developerworks/java/jdk/lifecycle/

Java 8 – Standard Features

Java 8 SE Standards Structure

- Java Specification Request (JSR) 337
  - Java Specification Request that pulls together the set of changes proposed for Java SE
  - Umbrella document describing the themes of the release and operating rules

- Release drivers
  - Lambda expressions
    - Java language changes to support multi-core programming
    - Corresponding changes to collections APIs to exploit parallelisation
  - Virtual extension methods
    - Language constructs designed for library evolution
    - Enhancements to existing interfaces to provide new functionality

- Component JSR specifications and Java Enhancement Proposals (JEPs)
  - A list of ~55 significant items delivered as part of Java 8 GA
  - JSRs are developed in conjunction with Java 8, and incorporated
  - Each JEP is at least two weeks platform development work

- Features, bug fixes, and security patches
  - Hundreds of smaller pieces of work that don't warrant a JEP

Anonymous Inner Classes

- Currently anonymous inner classes are used for passing context (poorly)

```
final State myState = ...
model.addEventListener(new Listener() {
  public void eventCallback(Event e) {
    if (myState.isActive() && e.isInteresting()) {
       ...
    };
  }
});
```

- Bulky syntax and confusion surrounding the meaning of names and "`this`"
- Inflexible class-loading and instance-creation semantics, often leading to 'class leaking'
- Inability to capture non-final local variables

- Often used with:
  - `java.lang.Runnable`
  - `java.security.PrivilegedAction`
  - `java.io.FileFilter`
  - `java.beans.PropertyChangeListener`
  - ...etc

JSR 335 – Lambda expressions

- Lambda expressions in Java 8 have a simple syntax
  - Think of them as “anonymous methods”
  - No need for the class definition infrastructure

```
() -> Integer.SIZE;

(int x, int y) -> x + y

(String s, int x) -> { x+=2; System.out.println(s); return x;}
```

- No new level of lexical scoping, so variable names and 'this' are identical to enclosing environment
- The Java 8 compiler will allow references to 'effectively final' variables even if they are not marked final
  - compiler data flow determines that the value is not being modified by the lambda expression

```
State myState = …
Listener ear = (Event e) -> {
  if (myState.isActive() && e.isInteresting()) {
     ...
  };
};
model.addEventListener(ear);
```

Lambdas enable localization of operations

- Lambdas allow the control flow for operations on data to reside near the data
- e.g. internal iteration
  - New methods on collections that accept Lambda expressions as operations on them

Java 7 syntax

```
for (MyType element : myCollection) {
  element.reset();
};
```

Java 8 syntax

```
myCollection.forEach(element -> {element.reset();});
```

  - Allows the data collections to decide how to iterate over elements
    - Laziness, out-of-order execution, parallelism

Lambdas enable data stream operations

- The operations on data structures can now be pipelined into a stream
- Streams can re-order and optimize lambda operations based on the characteristics of the underlying data stream
  - `ORDERED`, `DISTINCT`, `SORTED`, `SIZED`, `NONNULL`, `IMMUTABLE`, `CONCURRENT`, and `SUBSIZED`.

~ Stream Pattern in Java 8 ~

- Ask your collection / IO channel / function for a stream, describe operations, gather results.
  - Intermediate operations on streams produce new streams
  - Terminal operations produce results

```
Stream<MyType> stream = myCollection.stream()
     .filter(element -> element.length() == 0)
     .forEach(element -> { element.reset(); });

Set<MyType> emptyTypes = stream.into(new HashSet());
```

- Intermediate operations can be lazy, terminal operations will be eager

Virtual extension methods

- Recognize that Lambda and stream operations are useful on existing collection types
- Need some way to extend well established data structures while retaining compatibility
- Option 1: Creating parallel hierarchy of similar structures
  - Bulky class library with constant need to juggle types
- Option 2: Adding a new method to an existing interface
  - Binary compatible, but disenfranchises implementers
- Option 3: Enhance language to provide default implementations in interfaces
  - Interface declarations contain code, or references to code, to run if classes do not provide an implementation

```
public interface Set<T> extends Collection<T> {
     public boolean add(E e);
     public void clear();
     ...
     public void forEach(Block<T> blk)
          default Collections.<T>setForEach;
}
```

JSR 308: Annotations on Java Types

- Extending the scope of annotations as introduced in Java 6
  - Annotate type usage, not just type declaration
  - Carried in class files for robust development time checking
- Allows for pluggable extensions to Java language type checking
  - Strengthen and refine the built-in type system
  - Type annotations can be written before any type, e.g. `@NonNull String`

~ Expected usage? ~

- Software quality and security
  - Null pointer errors, side effects on immutable data, race conditions, information leakage, non-internationalized strings, etc.
- Checkers framework use additional information
  - Non-prescriptive use of annotations allows for varied tooling
  - Expect to see variety of coding tools use annotations for developer feedback

```
List<@NonNull String> strings;

myGraph = (@Immutable Graph) tmpGraph;

class UnmodifiableList<T> implements @Readonly List<@Readonly T> { ... }

@Tainted String entry;
```

JSR 310: Date and Time API

- A new, modern, date and time API for Java
- Current date and time types are split across multiple packages
  - `java.util, java.sql, java.text`, etc.
- API could be improved in a number of ways...
  - `java.util.Date` is actually a timestamp!
  - Based on years from 1900 onwards
  - Calendar instances cannot be converted to simple date formatted strings
  - etc.
- JSR-310 is a top to bottom review of the date and time handling in Java
  - Based upon relevant standards, including ISO-8601, CLDR, and BCP47
  - Types represent point in time, duration, and localization

```
java.time
     main API for dates, times, instants, and durations
java.time.calendar
     Support for Hijrah, Japanese, Minguo, Thai Buddest calendar systems
java.time.format
     Provides classes to print and parse dates and times
java.time.temporal
     Expands on the base package for more powerful use cases
java.time.zone
     Support for time-zones and their rules
```

More standard Java 8 features, at a glance...

- Language
  - Access to Parameter Names at Runtime
  - Add Javadoc to javax.tools (JSR 199 MR)
  - Annotations on Java Types (JSR 308)
  - Generalized Target-Type Inference (JSR 335)
  - Lambda Expressions & Virtual Extension Methods (JSR 269 MR, 335)
  - Repeating Annotations (JSR 269 MR, 337)
- Core Libraries
  - Base64 Encoding & Decoding
  - Bulk Data Operations for Collections (JSR 335)
  - Concurrency Updates
  - Date & Time API (JSR 310)
  - Enhance Core Libraries with Lambda (JSR 335)
  - JDBC 4.2 (JSR 114 MR, 221 MR)
  - Parallel Array Sorting
- I18n
  - BCP 47 Locale Matching
  - Unicode 6.2

OpenJDK

http://openjdk.java.net/

- Security
  - Configurable Secure Random-Number Generation
  - Enhance the Certificate Revocation-Checking API
  - Limited doPrivileged
  - NSA Suite B Cryptographic Algorithms
  - TLS Server Name Indication (SNI) Extension
- Platform
  - Compact Profiles
  - Prepare for Modularization (JSR 160 MR, 173 MR, 206 MR, 337)

Java Community Process

Java 8 – IBM unique features

Java 8 – IBM unique features

Hardware exploitation

IBM z13 and IBM Java 8 – designed together

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 Integrated Information Processor (zIIP), and Integrated Facility for Linux (IFL)*

Single Instruction Multiple Data (SIMD)

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

Cryptographic Function (CPACF)

- **Transparently accelerate IBMJCE security provider**
- **Block ciphering, Secure Hashing and Public Key Cryptography**

New Instructions

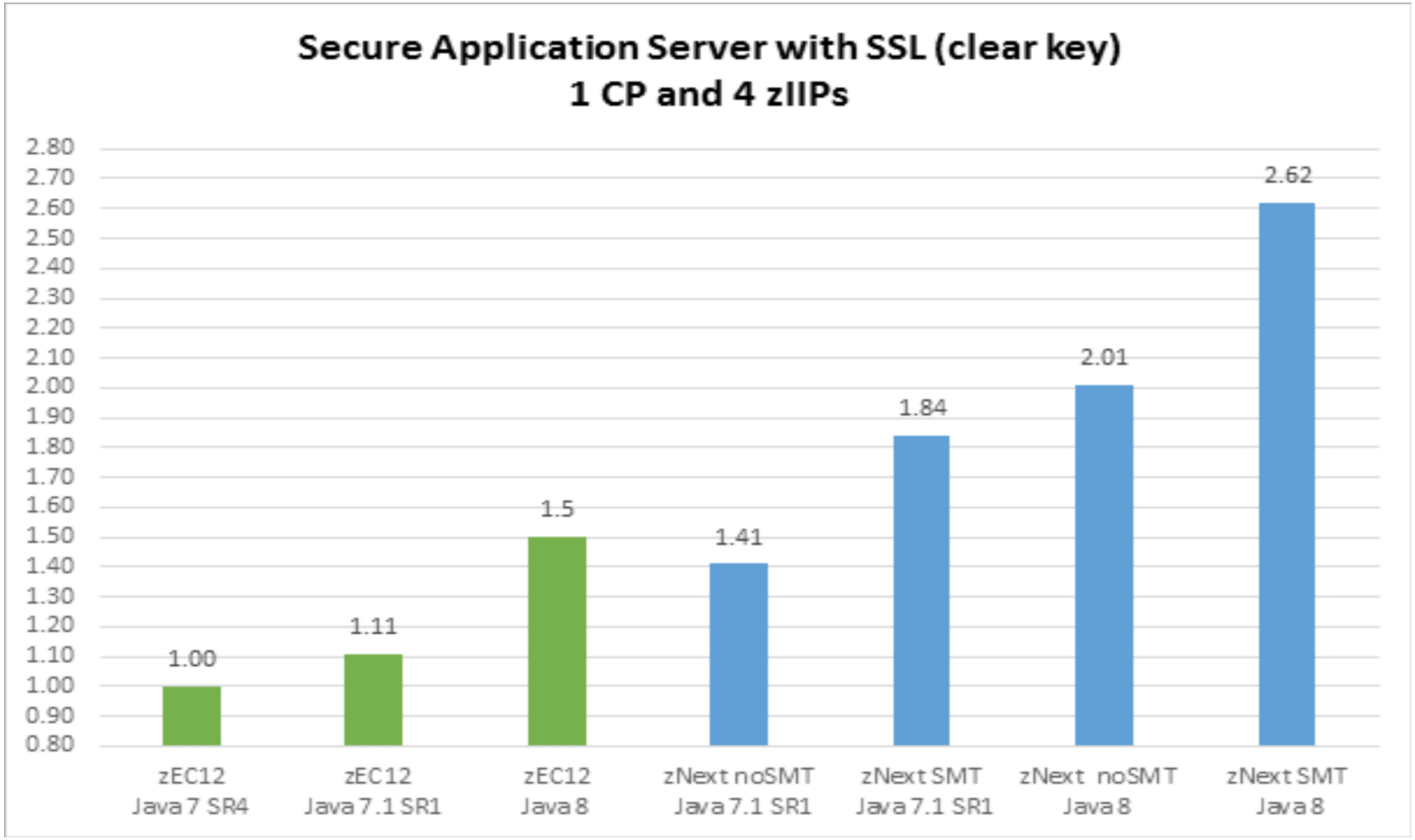
Up to 50% improvement for generic applications

No application code changes!

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

The Result: Java Application Server Performance

Combined effect of moving from zEC12 Java 7 to z13 Java 8



2.62x improvement in throughput with IBM Java 8 and IBM z13

(Controlled measurement environment, results may vary)

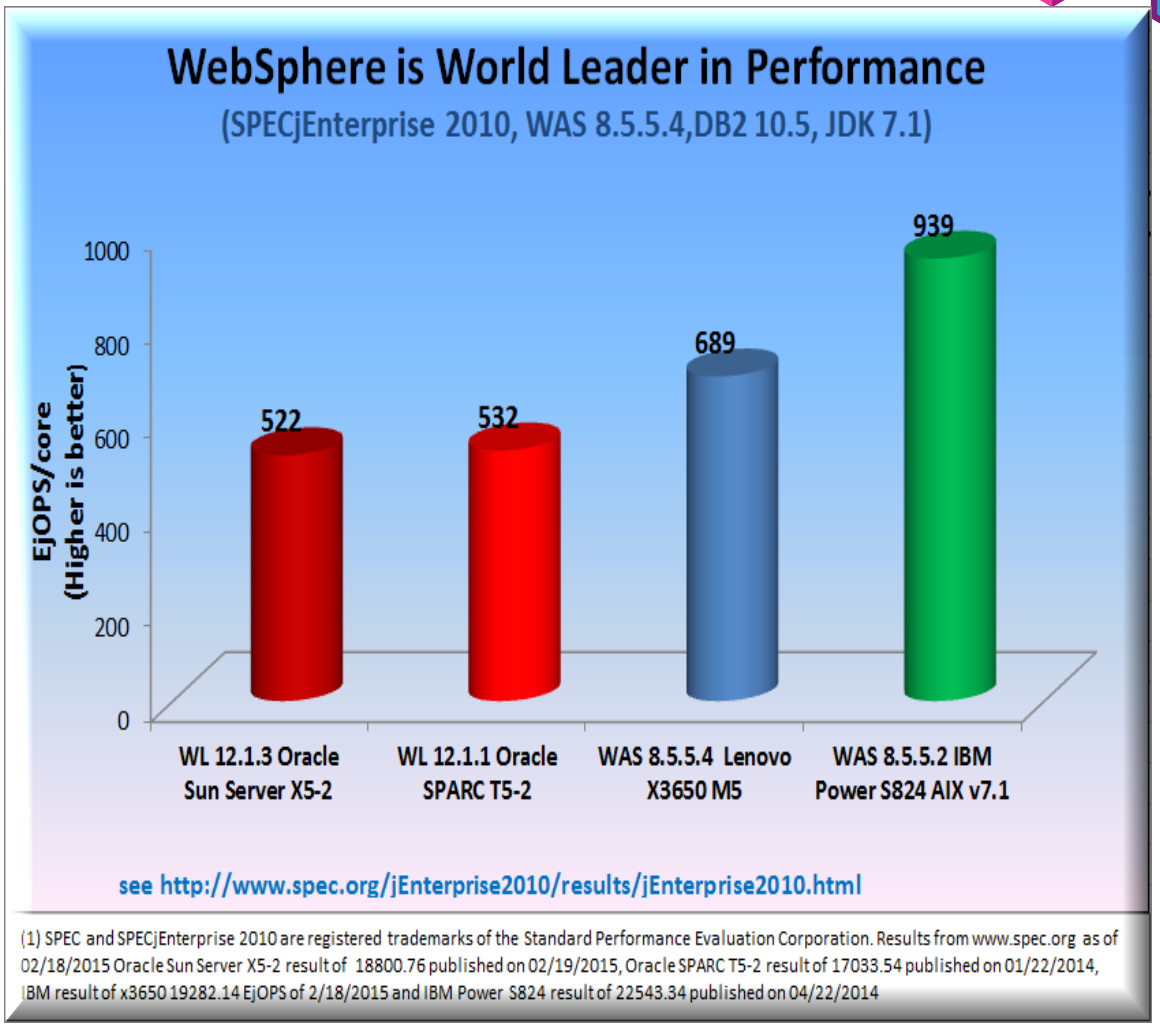
IBM POWER Architecture and IBM Java 8

- Focus on new support built into POWER 7 and POWER 8 hardware
  - Transactional Memory (seeing 2x on select concurrent classes)
  - On-Core AES crypto 2.5 x faster, 30-40% faster than vector multimedia extension (VMX)
  - Simultaneous Multi-threading SMT8 exploit, 20% improvement over SMT4
  - General instruction set improvements – DirectMove, Vector load/store
  - JIT improvements
    - New prefetching capabilities
    - Extended divide instructions
    - Conversion between integer and float
    - Bit permutation and popcount instructions
    - BCD assist - Exploited through Java BigDecimal
- IBM Java on POWER Little Endian (LE) configurations
- IBM Java support for co-processors (FPGA, GPU, Security, etc)

Websphere Application Server on POWER8

IBM hardware + IBM Software → Unbeatable System Performance

- Exploit Significant Parallelism Offered by Power 8
- Exploit Transactional Memory
- Improve Per Core Performance
- Reduce Virtualization Overhead with PowerVM
- Exploit Faster Networking and Storage Capabilities
- Improve Security Workload Performance
- Exploit Larger Cache including L4 Cache

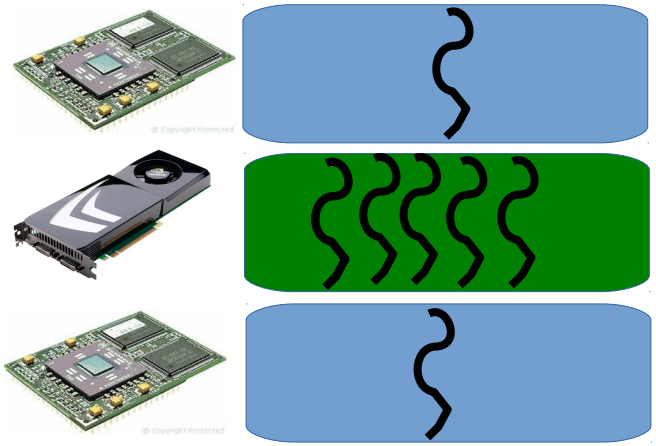


Java 8 – IBM unique features

GPU off loading

IBM Power 8 – now with GPU acceleration

- GPU devices plug into the host PCIe bus to provide massive arrays of co-processors



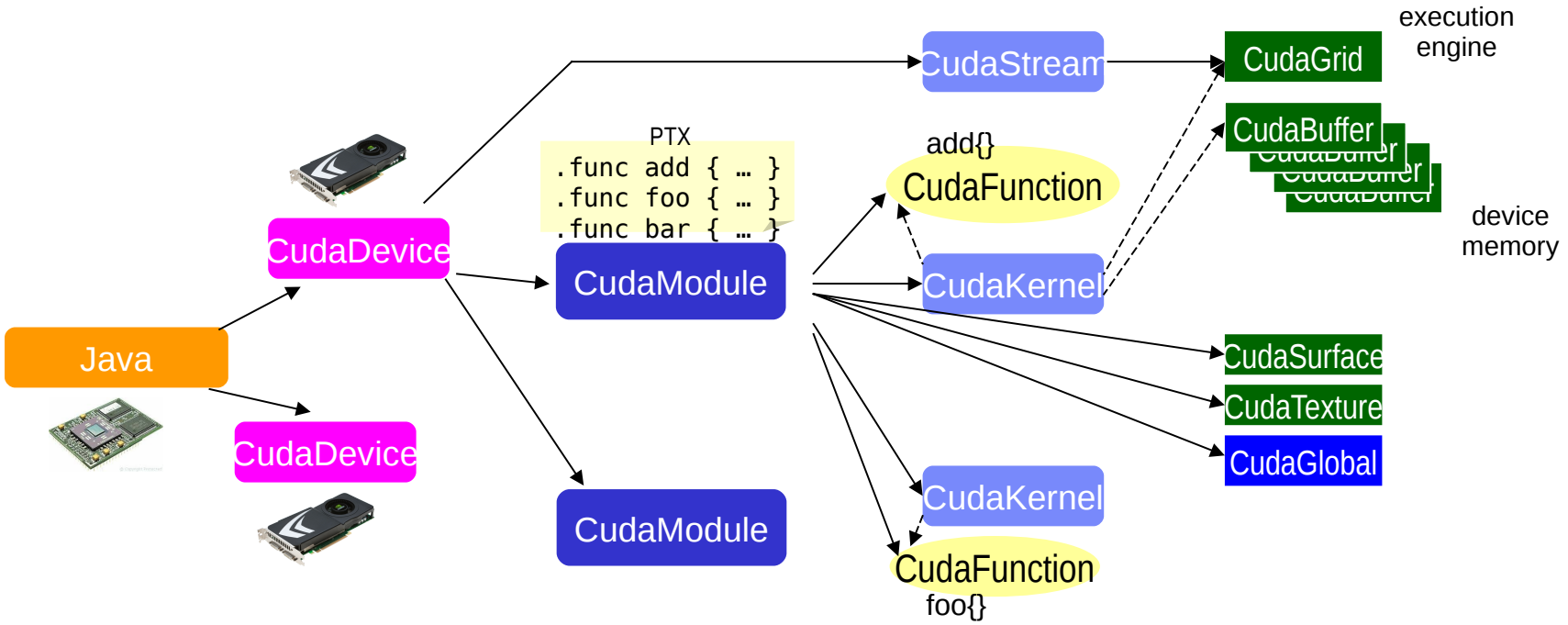
- Typical scenario for heterogeneous programming:
  - Host computer with CPU(s) and GPU(s) installed on PCIe bus
  - Programmer identifies parallelizable, compute intensive routine, and codes to GPU
  - Flow of data and control passes between CPU host and GPU device under control of host device

Particularly suited to scientific and numerical analysis problems (e.g. linear algebra). We have focused on Nvidia CUDA as the programming model for exploiting GPUs.

Three-tiers of exploitation in Java:

- CUDA4J : a low-level interface to the GPU for applications that want direct control from Java, enabling reuse of kernels from Java, faster time to market
- Java SE library exploitation : backing standard Java APIs with a GPU implementation for improved performance (sort, etc)
- Dynamic workload off-loading : identifying patterns in application code that will benefit from parallelisation directly in the JIT

Fundamental types in CUDA4J



CudaLinker — Used to combine multiple cubin/fatbin/PTXs into single module

CudaEvent — Device events

Corresponds to a HW feature in GPU

Relationship for generating an instance

Relationship as an argument

Limitations and considerations

- Allows developers to code explicitly for the GPU
  - These are new APIs that give close control of the device
  - Uses familiar concepts and paradigms for GPU experts
  - Convenience and productivity improvements from language
  - Fundamental building blocks for higher level algorithms
- Requires the developer to identify suitable GPU workloads
  - Re-code routines to operate on data in parallel
  - Minimize branching flow of control in kernels
- Amortizing overhead of moving work to GPU
  - Time taken to copy data between host and device over PCIe
  - Overhead of switching flow of control from CPU to GPU

Flickr: Gareth

GPU-enabling standard Java SE APIs

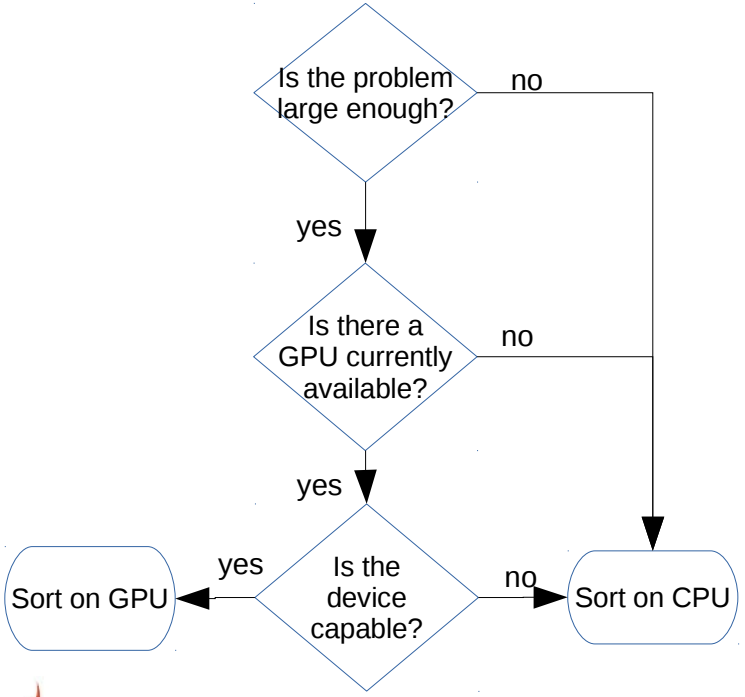
- Natural question after seeing the good speed-ups using explicit programming …
- What areas of the standard Java API implementation are suitable for off-loading onto GPU?

- We picked

  `java.util.Arrays.sort(int[] a)` and friends
  - GPU modules exist that do efficient sorting

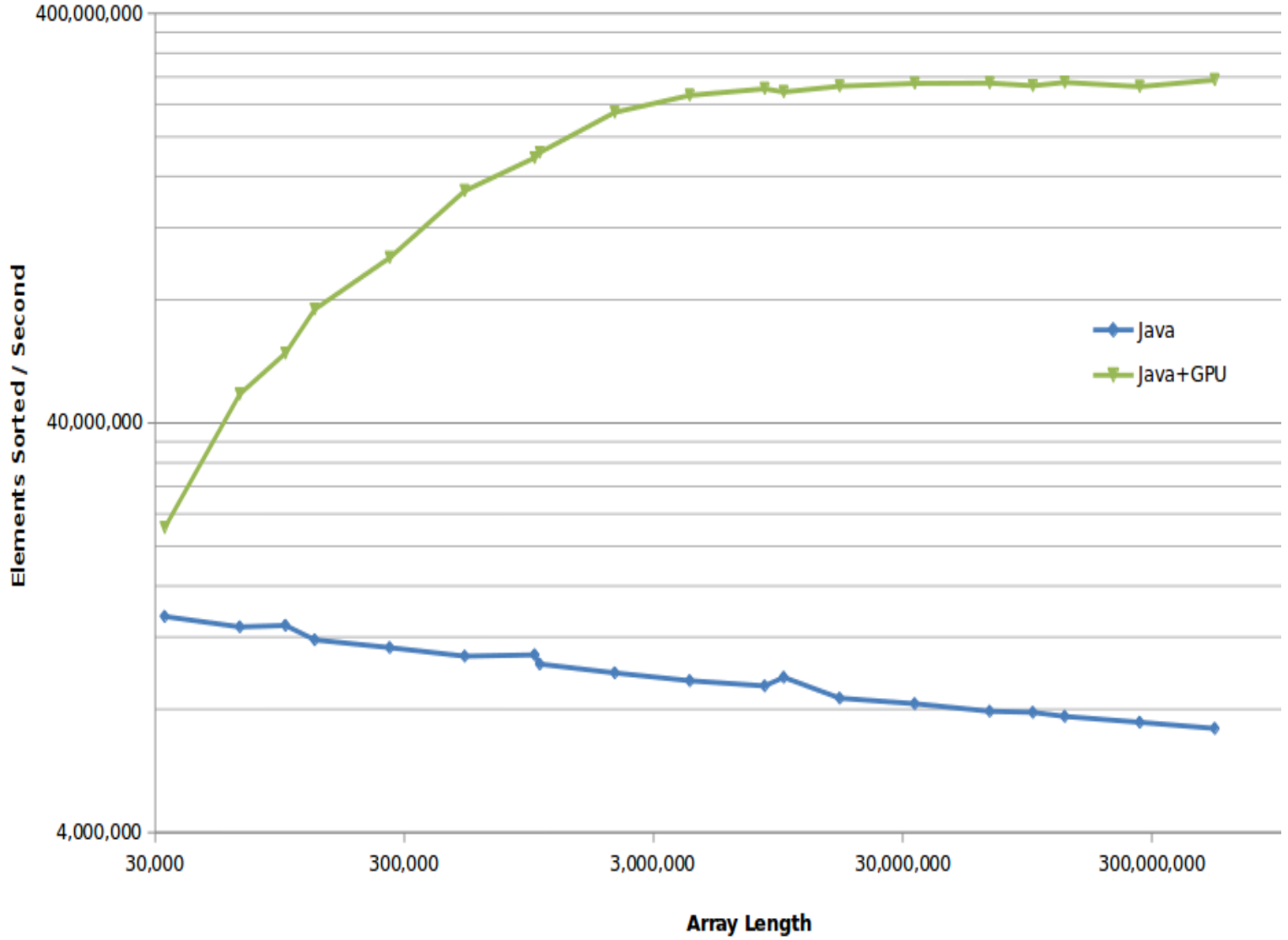
We employ heuristics that determine if the work should be off-loaded to the GPU.

- Overhead of moving data to GPU, invoking kernel, and returning results means small sorts (<~20k elements) are faster on the CPU.
- Host may have multiple GPUs. Are any available for the task?
- Is there space for conducting the sort on the device?



IBM Developer Kits for Java
ibm.com/java/jdk

GPU-enabled array sort method



IBM Power 8 with Nvidia K40m GPU

Beyond specific APIs – Java 8 streams

- Streams allow developers to express computation as aggregate parallel operations on data
- For example:

```
IntStream.range(0, N).parallel().forEach(i -> c[i] = a[i] + b[i]);
```

creates a stream whose operations can be executed in parallel

- What if we could recognize the terminal operation and conduct it on the GPU?
  - ✔ Reuses standard Java idioms, so no code changes required
  - ✔ No knowledge of GPU programming model required by the application developer
  - ✖ But no low-level manipulation of the device – the Java implementation has the controls
  - ✔ Future smarts introduced into the JIT do not require application code changes

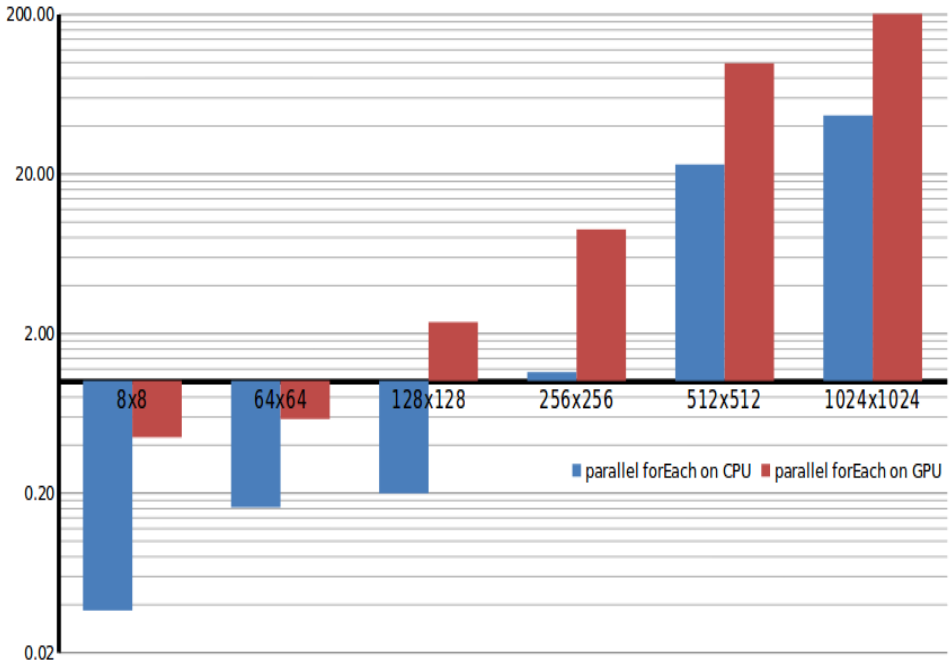
JIT / GPU optimization of Lambda expression

JIT recognized Java code for matrix multiplication using Java 8 parallel stream

```
public void multiply() {
  IntStream.range(0, COLS*COLS).parallel().forEach(
    id -> {
      int i = id / COLS;
      int j = id % COLS;
      int sum = 0;

      for (int k=0; k<COLS; k++) {
         sum += left[i*COLS + k] * right[k*COLS + j]
      }
      output[i*COLS + j] = sum;
  });
}
```

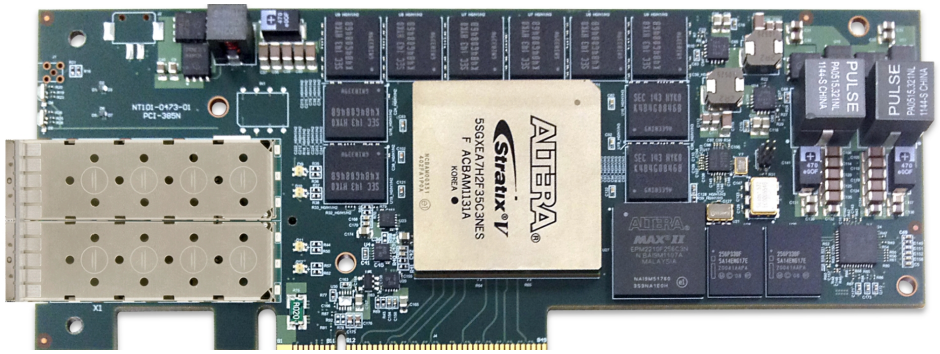
Speed-up factor when run on a GPU enabled host



IBM Power 8 with Nvidia K40m GPU

Java 8 – IBM unique features

ZIP compression off-loading



zEnterprise Data Compression (zEDC)

What is it?

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

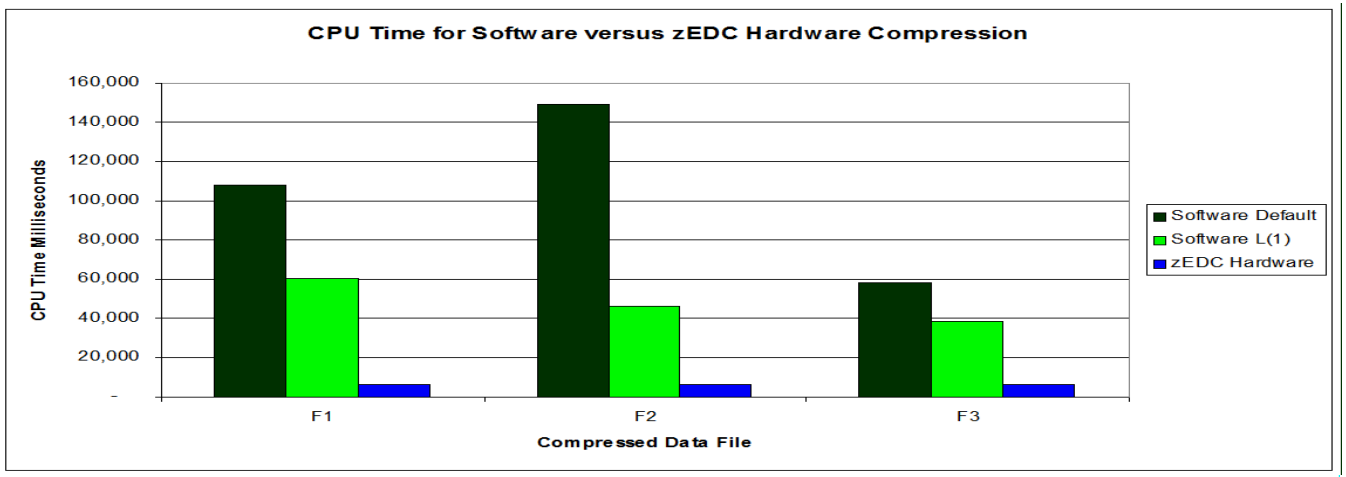
Using IBM Java 7R1 :

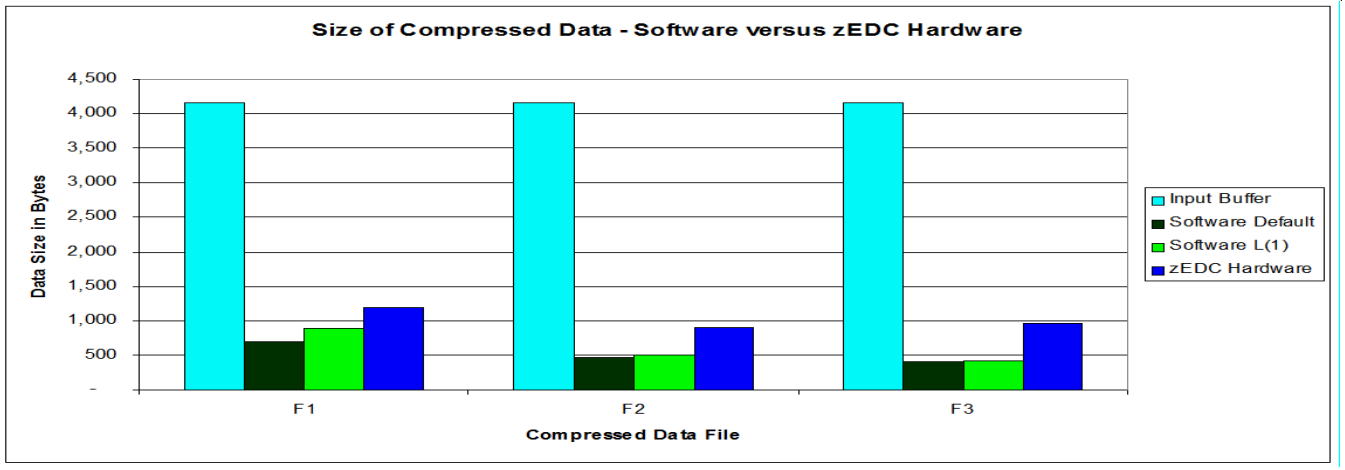
Java applications compress files using java.util.zip.GZIPOutputStream class

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

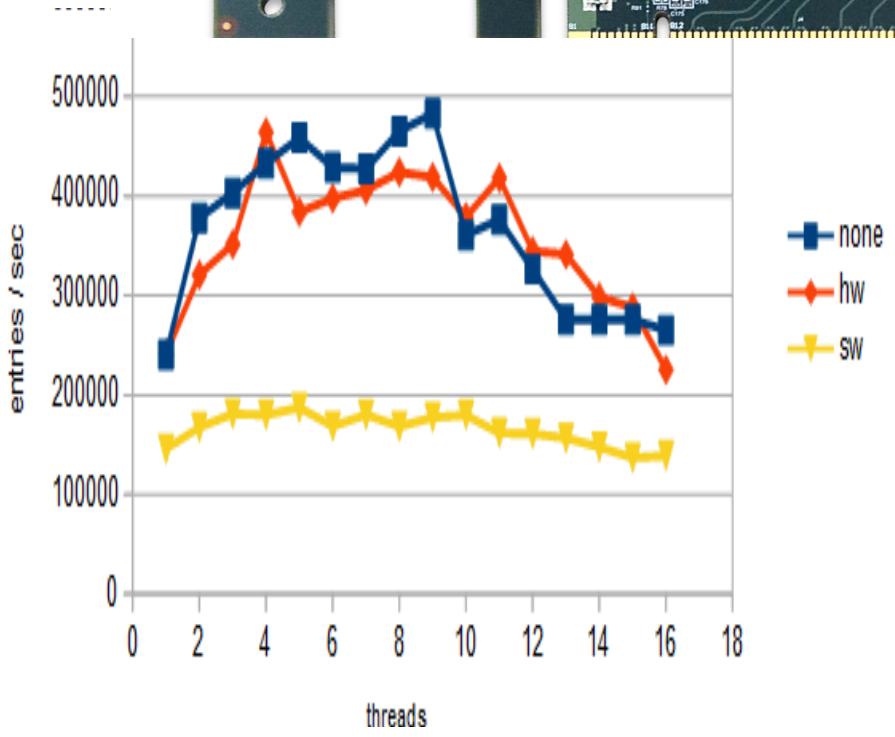
Up to 74% reduction in Elapsed time (not shown)

Compression ratio up-to ~5x





PowerPC : Field-Programmable Gate Arrays (FPGAs)



Hardware-backed Java compression APIs

Work sent to custom card firmware rather than the CPU.

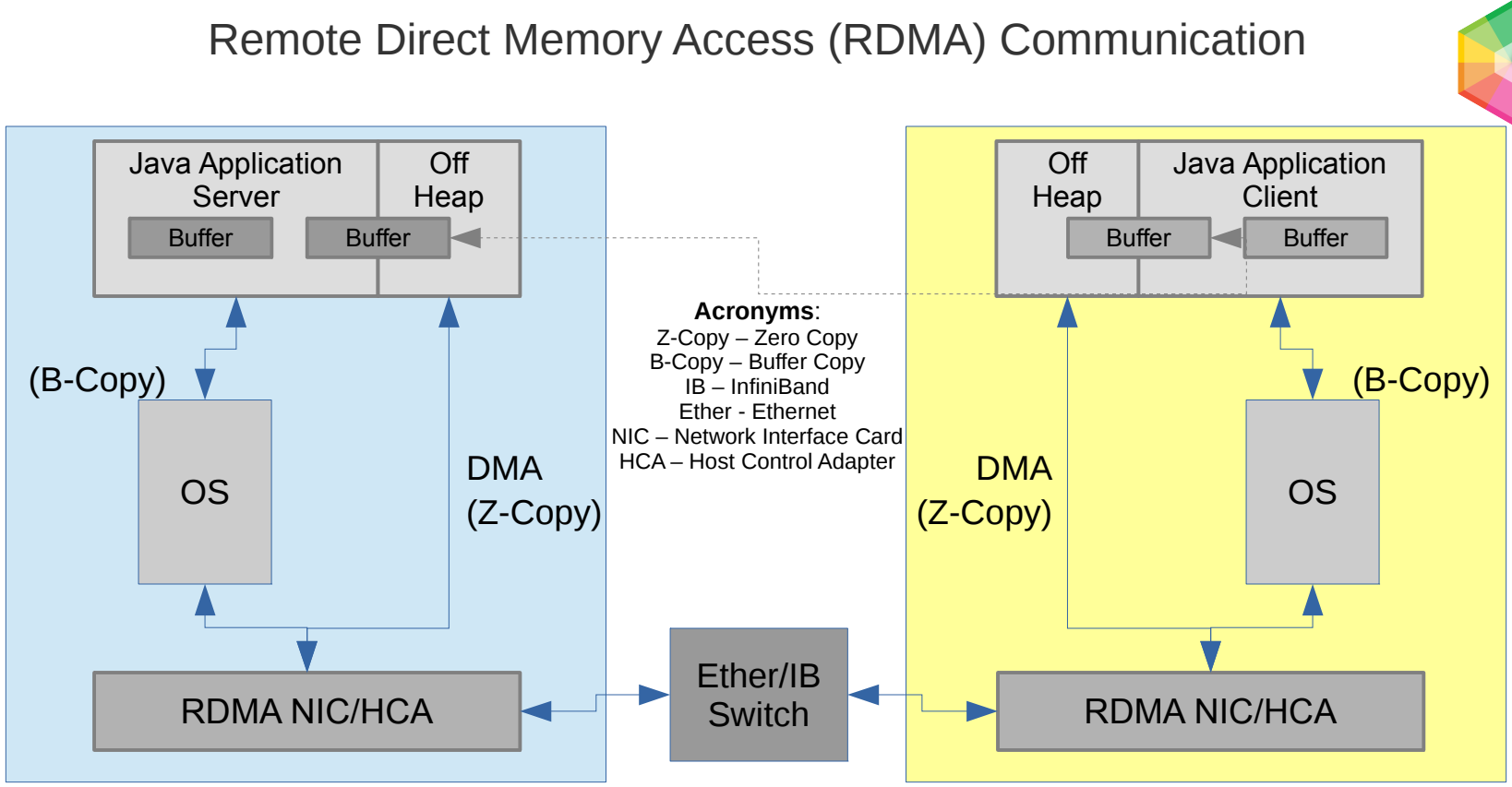
Not only does it run faster on the card, but it frees up the CPU to do other things.

Benchmarking Liberty Power Linux shows high performance extensible log (HPEL) engine writes *compressed* logs as quick as *uncompressed*

Java 8 – IBM unique features

Remote Direct Memory Access

What is RDMA?

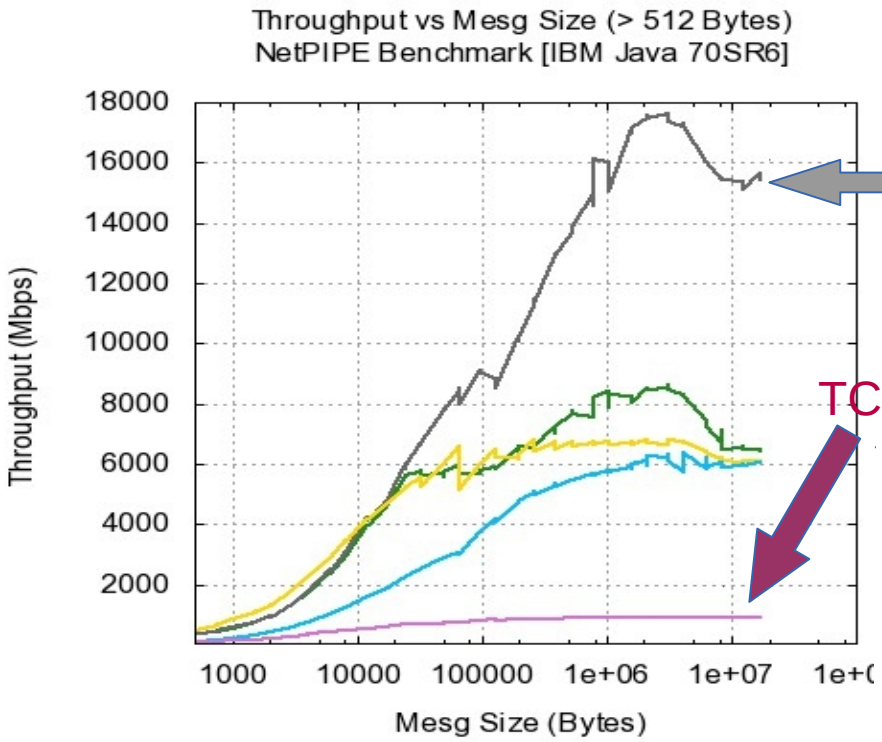


- Low-latency, high-throughput networking
  - Direct 'application to application' memory pointer exchange between remote hosts
  - Off-load network processing to RDMA NIC/HCA – OS/Kernel Bypass (zero-copy)

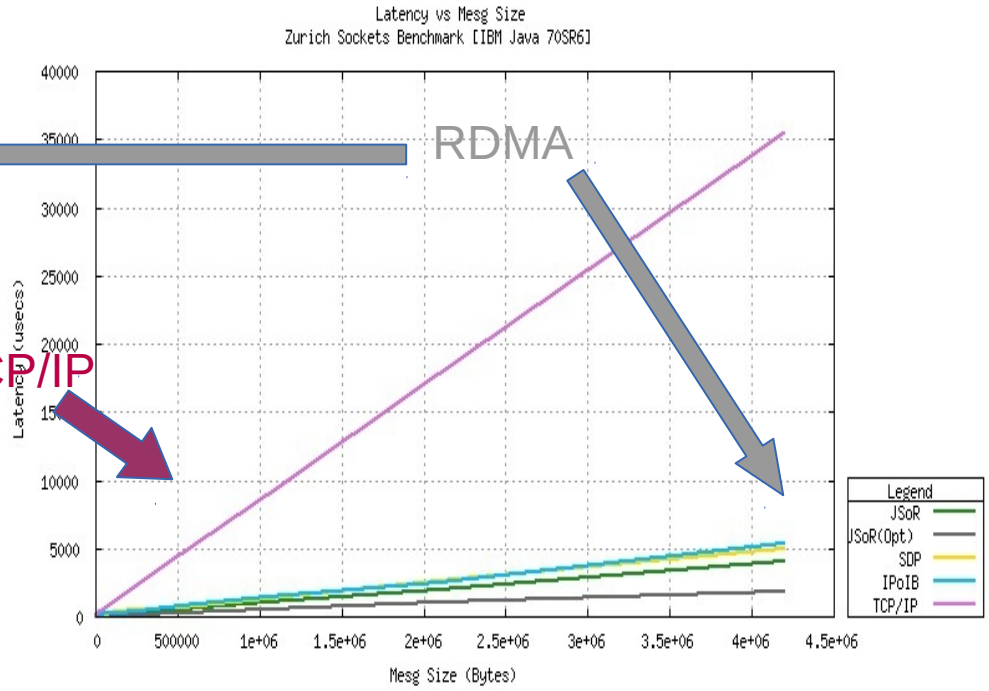
Remote Direct Memory Access (RDMA)

- Protocols for high-performance network fabrics – 10/40/56 Gbps
- Transparent availability over java.net.Socket APIs
- Enables data caches, workloads, even virtual images to be hardware transient

Throughput



Latency



RDMA

TCP/IP

Java 8 – IBM unique features

Data Access Accelerators &
Packed Objects

Data Access Accelerator (DAA)

Data-centric tasks such as big data, analytics and inter-language communication require optimal performance for accessing and operating on native format data records and types from Java. Prefer to avoid object creation, data copying, abstraction, boxing etc

DAA provides a Java library for bare-bones data conversion, arithmetic etc.

Provides native-oriented operations directly on Java byte arrays

Orchestrated with JIT for deep platform optimization

- No intermediate Java objects created when recognized by the IBM JIT
- Avoid expensive Java object instantiation by allowing in-place operations

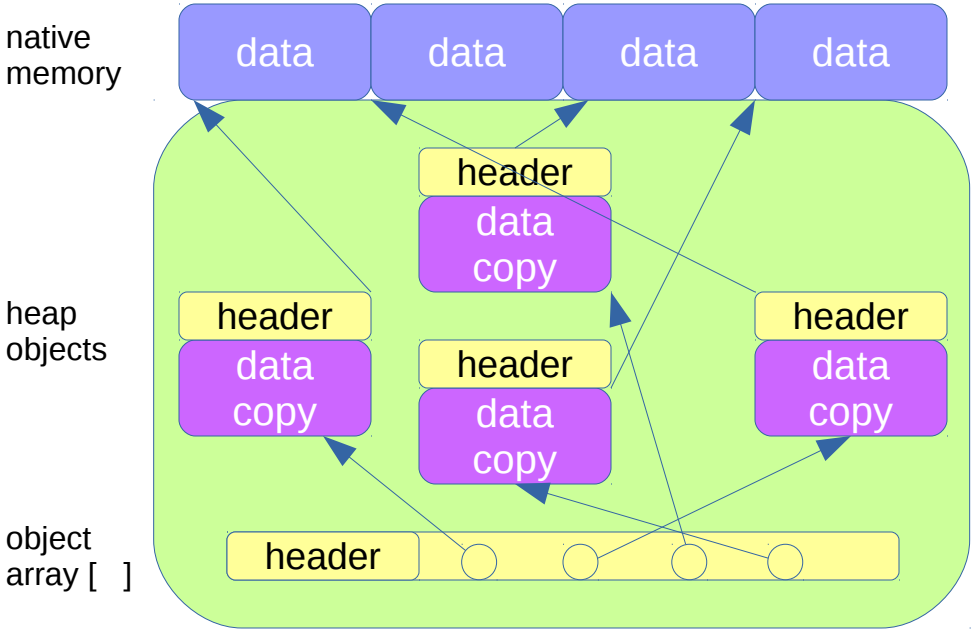
Benefits:

Expose hardware acceleration in a platform and JVM-neutral manner (2 – 100x speed-up)

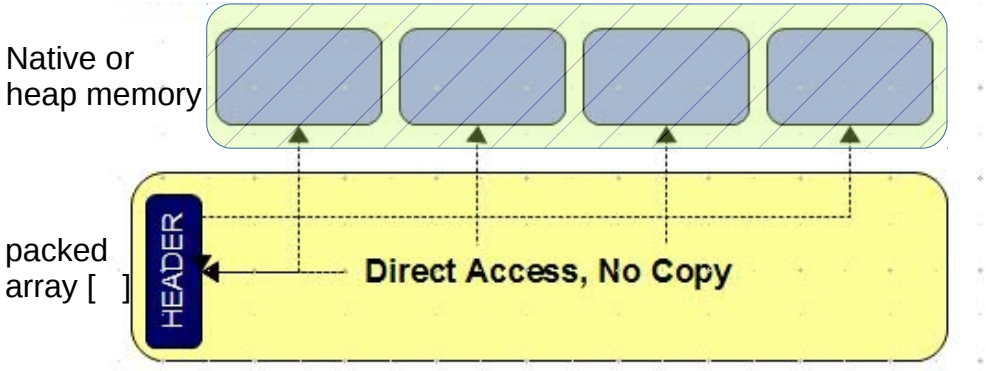
Can provide significant speed-up to record parsing frameworks

Can provide significant speed-up for data marshaling and inter-language communication

IBM Java SDK: Packed objects support

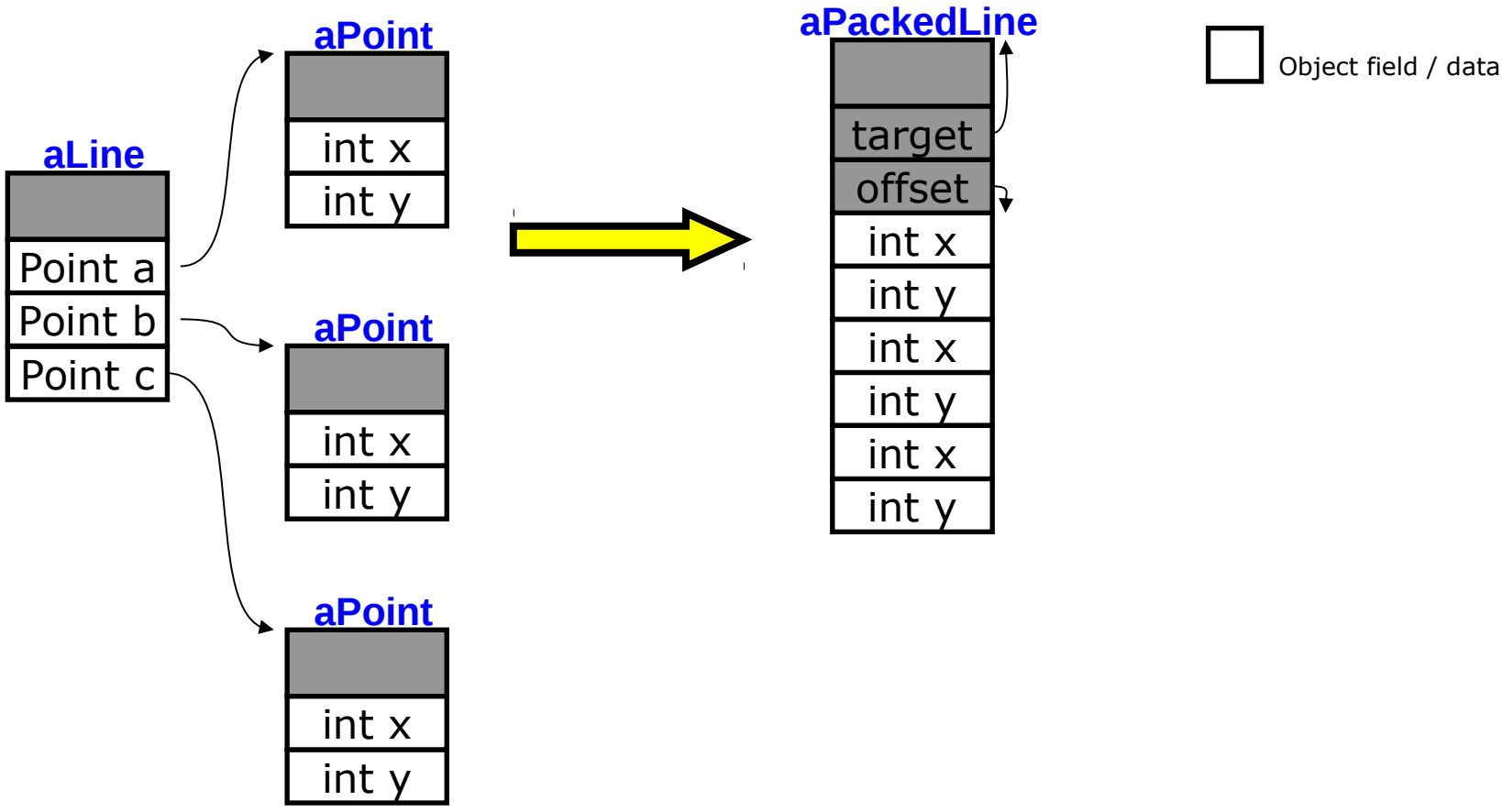


- Java requires memory to be in Java "object" form to be accessed directly
- External data needs to be read into Java heap format to use – conversion is expensive
- Memory bloat occurs due to data copies and headers
- Natural object representation looses data locality properties

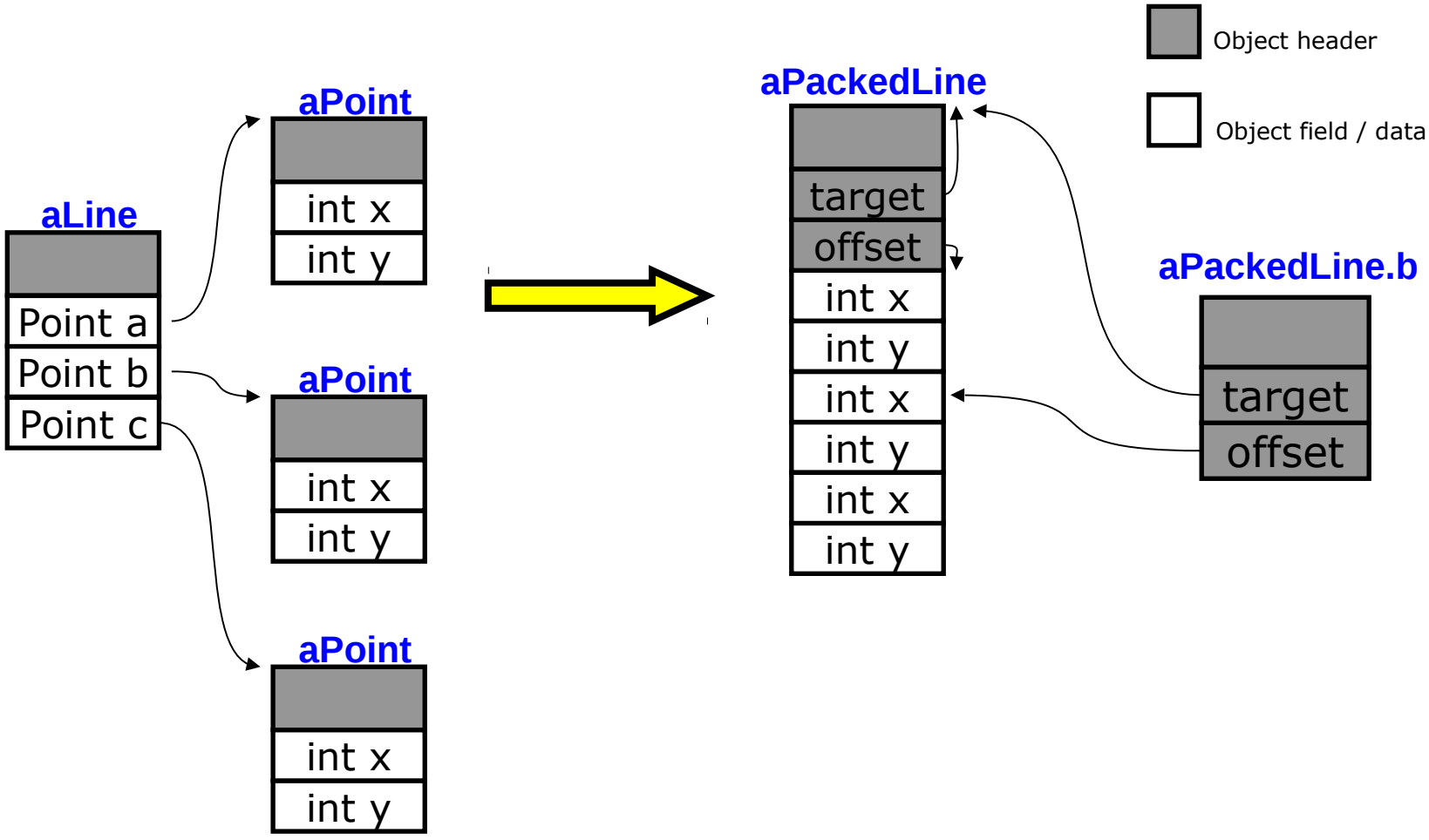


- PackedObjects enables direct access to data in arbitrary formats without the redundant copying; no conversion
- PackedObjects data can be in native memory or Java heap space

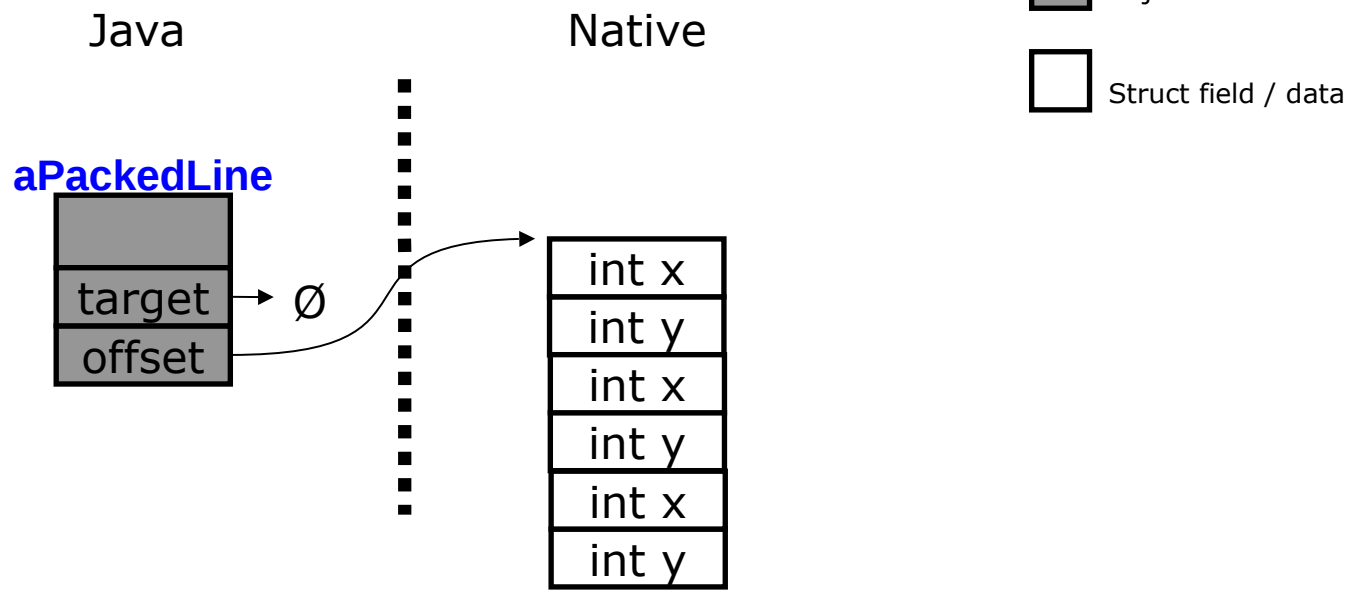
Packed Objects: Heap referenced data



Packed Objects: Heap referenced data



Packed Objects: In Practice with Native Access

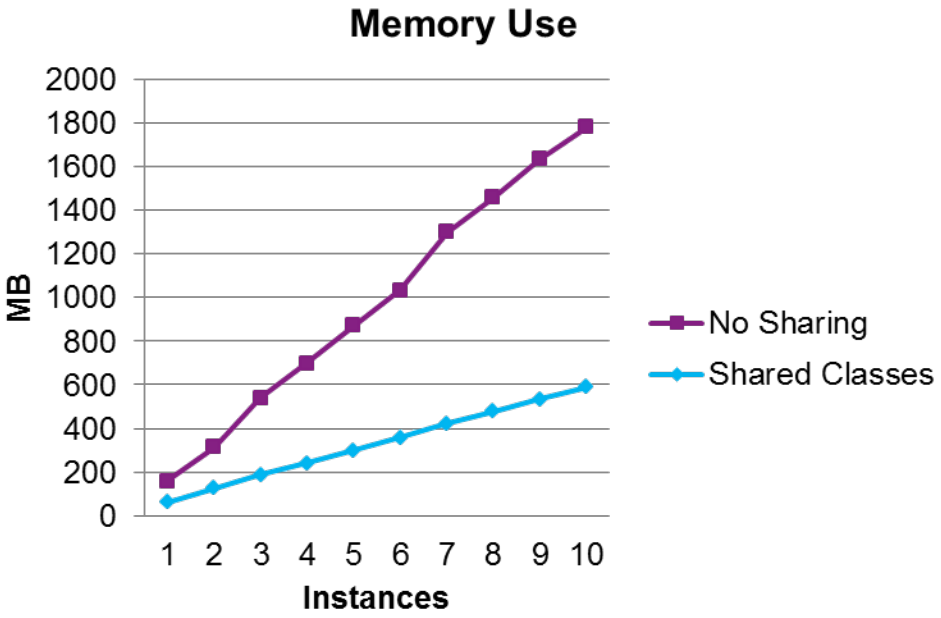


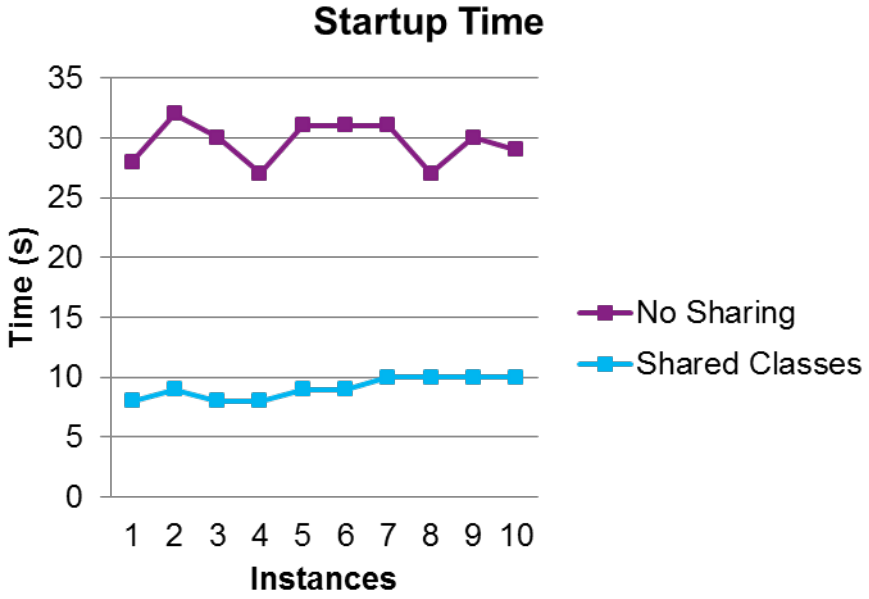

```
@Packed
final class PackedPoint extends PackedObject {
    int x;
    int y;
}
```

Java 8 – IBM unique features

Cloud enhancements

Docker and IBM Java's shared classes, faster startup + higher density





10 instances of an app server in Docker containers
~2x better density, ~2x faster startup

docker

IBM Java SDK: Cloud support

- **-Xtune:virtualized includes a 'deep idle' mode for the JIT**
  - Reduces background JIT activity when the application is idle by ~85%

- **Improved OperatingSystemMxBean**
  - New operating system queries supported to allow applications to adjust to current load conditions as dynamic situation changes
  - New API includes:
    - processCpuLoad()
    - getFreeSwapSpaceSize()
    - getTotalSwapSpaceSize()

- **-Xsoftmx everywhere**
  - Allows runtime modification of JVM heap size programmatically, can be used to take advantage of hypervisor hot-add memory, or to reduce heap size in idle programs.
  - Generalization of an AIX DLPAR feature

Java 8 – IBM unique features

Monitoring and Management Tools

Enhanced Monitoring and Diagnostics

- **New Features**
  - Enhanced com.ibm.jvm.Dump API
  - Additional information in javacore dumps
  - Improved JIT diagnostics
  - Improved performance of SDK method trace
  - New JVMTI extensions for subscribing to tracepoints
  - Improved SDK tracepoints

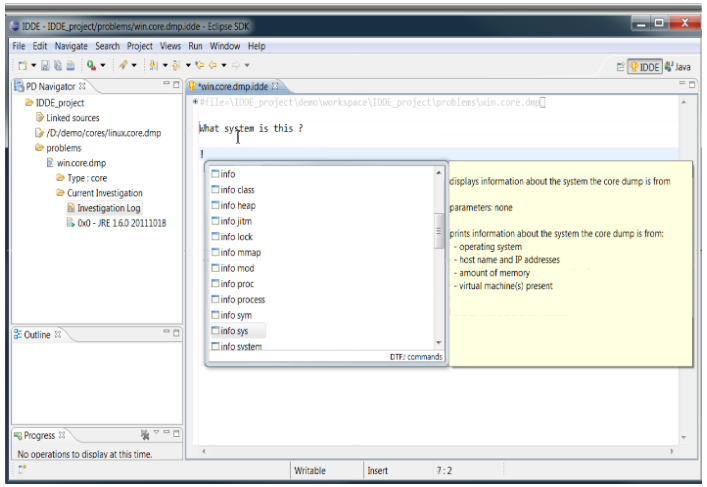
- **Customer benefits**
  - Improved API enables Java applications to capture dump diagnostics easily
  - Additional content of environment and thread sections in the javacore dump aids first failure diagnosis
  - New JIT dump allows more rapid investigation and first failure diagnosis of JIT problems
  - Increased application throughput when tracing enables a powerful SDK facility for investigating Java application flow
  - Improved facility for integration of SDK trace with other software components
  - Extended tracepoint coverage improves SDK serviceability

Runtime Monitoring and Management Tools

Tools and documentation for application monitoring and problem diagnosis.

- **Free unified suite of tools** to understand different aspects of Java applications.
- **Lightweight, low performance overhead** monitoring and diagnostics.
- Provide more than **visualizations** – also provide observations and **recommendations**.

Tools in the IBM Monitoring and Diagnostic Tools Portfolio:

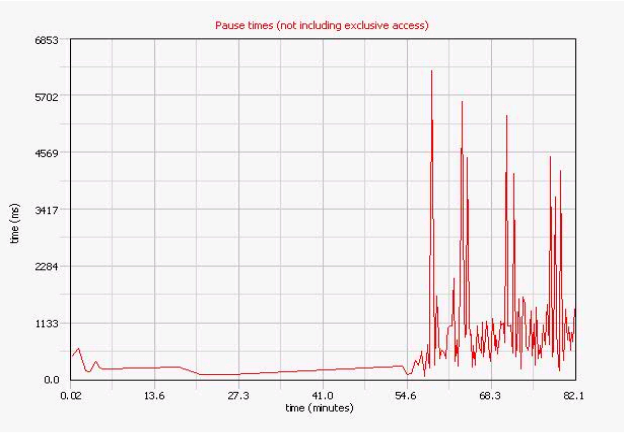


Interactive Diagnostic Data Explorer

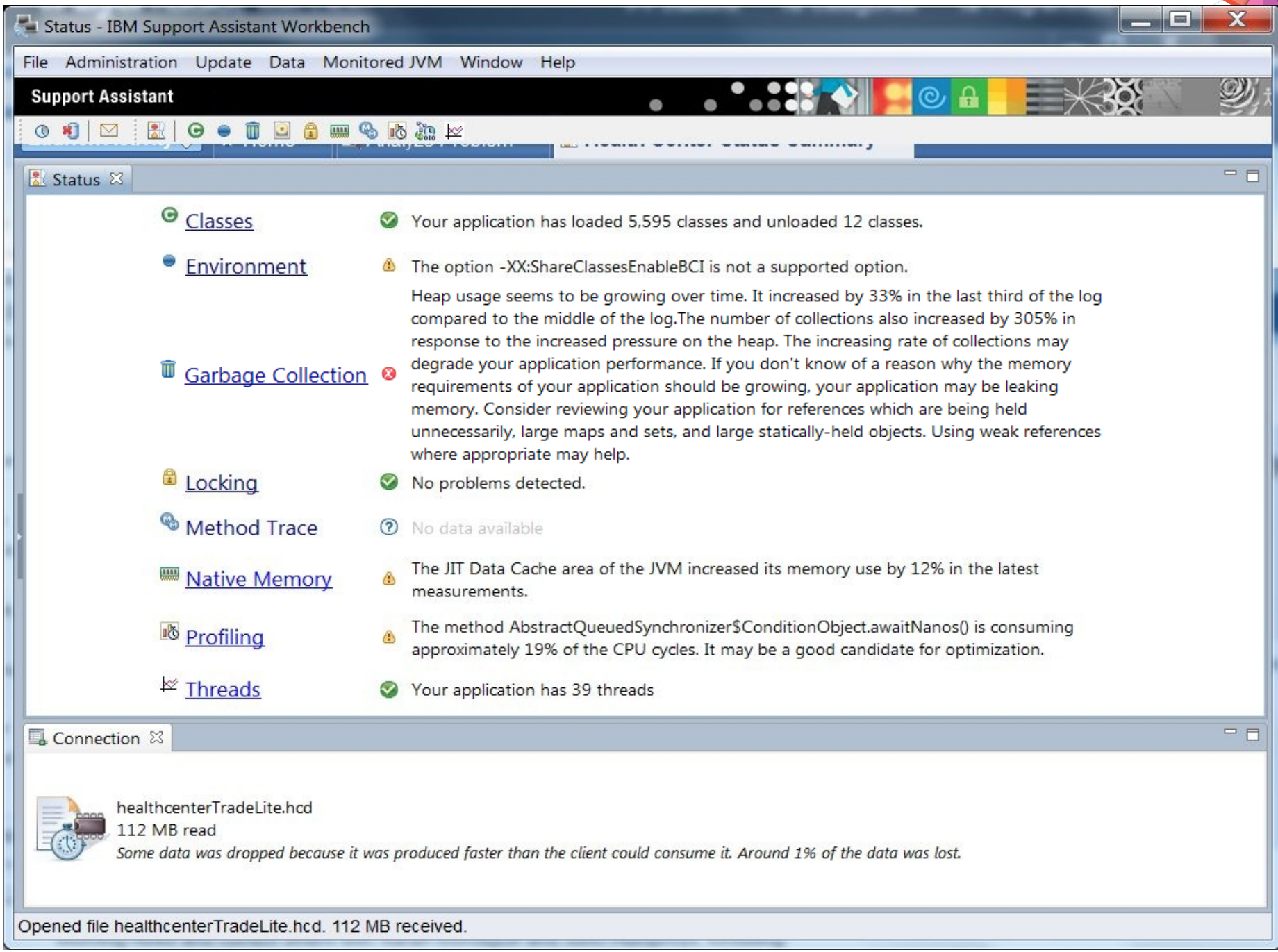
Garbage Collection and Memory Visualizer

Memory Analyser

Health Centre



For More Information Visit:
http://www.ibm.com/developerworks/java/jdk/tools/index.html


Status - IBM Support Assistant Workbench
File Administration Update Data Monitored JVM Window Help
Support Assistant
Status
Classes
Your application has loaded 5,595 classes and unloaded 12 classes.
Environment
The option -XX:ShareClassesEnableBCI is not a supported option.
Garbage Collection
Heap usage seems to be growing over time. It increased by 33% in the last third of the log compared to the middle of the log.The number of collections also increased by 305% in response to the increased pressure on the heap. The increasing rate of collections may degrade your application performance. If you don't know of a reason why the memory requirements of your application should be growing, your application may be leaking memory. Consider reviewing your application for references which are being held unnecessarily, large maps and sets, and large statically-held objects. Using weak references where appropriate may help.
Locking
No problems detected.
Method Trace
No data available
Native Memory
The JIT Data Cache area of the JVM increased its memory use by 12% in the latest measurements.
Profiling
The method AbstractQueuedSynchronizer$ConditionObject.awaitNanos() is consuming approximately 19% of the CPU cycles. It may be a good candidate for optimization.
Threads
Your application has 39 threads
Connection
healthcenterTradeLite.hcd
112 MB read
Some data was dropped because it was produced faster than the client could consume it. Around 1% of the data was lost.
Opened file healthcenterTradeLite.hcd. 112 MB received.

Additional things to note...

- Diagnostics Collector removed
  - Use IBM Support Assistant Data collector instead
- Java serial communications API no longer available in IBM Java 8
- Legacy operating system support removed
  - Windows XP, Server 2003
  - Linux RHEL 5, SLES 10, Ubuntu 8.04 & 10.04, Asianux Server 3

References

- **Get Products and Technologies:**
  - IBM Java Runtimes and SDKs:
    - https://www.ibm.com/developerworks/java/jdk/
  - IBM Monitoring and Diagnostic Tools for Java:
    - https://www.ibm.com/developerworks/java/jdk/tools/

- **Learn:**
  - IBM Java InfoCenter:
    - http://www.ibm.com/support/knowledgecenter/SSYKE2/welcome_javasdk_family.html

- **Discuss:**
  - IBM Java Runtimes and SDKs Forum:
    - http://www.ibm.com/developerworks/forums/forum.jspa?forumID=367&start=0

Notices and Disclaimers

U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IN NO EVENT SHALL IBM BE LIABLE FOR ANY DAMAGE ARISING FROM THE USE OF THIS INFORMATION, INCLUDING BUT NOT LIMITED TO, LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF PROFIT OR LOSS OF OPPORTUNITY. IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer is in compliance with any law.

Notices and Disclaimers (con't)

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of 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. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. IBM EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

- IBM, the IBM logo, ibm.com, Bluemix, Blueworks Live, CICS, Clearcase, DOORS®, Enterprise Document Management System™, Global Business Services ®, Global Technology Services ®, Information on Demand, ILOG, Maximo®, MQIntegrator®, MQSeries®, Netcool®, OMEGAMON, OpenPower, PureAnalytics™, PureApplication®, pureCluster™, PureCoverage®, PureData®, PureExperience®, PureFlex®, pureQuery®, pureScale®, PureSystems®, QRadar®, Rational®, Rhapsody®, SoDA, SPSS, StoredIQ, Tivoli®, Trusteer®, urban{code}®, Watson, WebSphere®, Worklight®, X-Force® and System z® Z/OS, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.