GreenTea JUG

IBM

# What is IBM Java ?

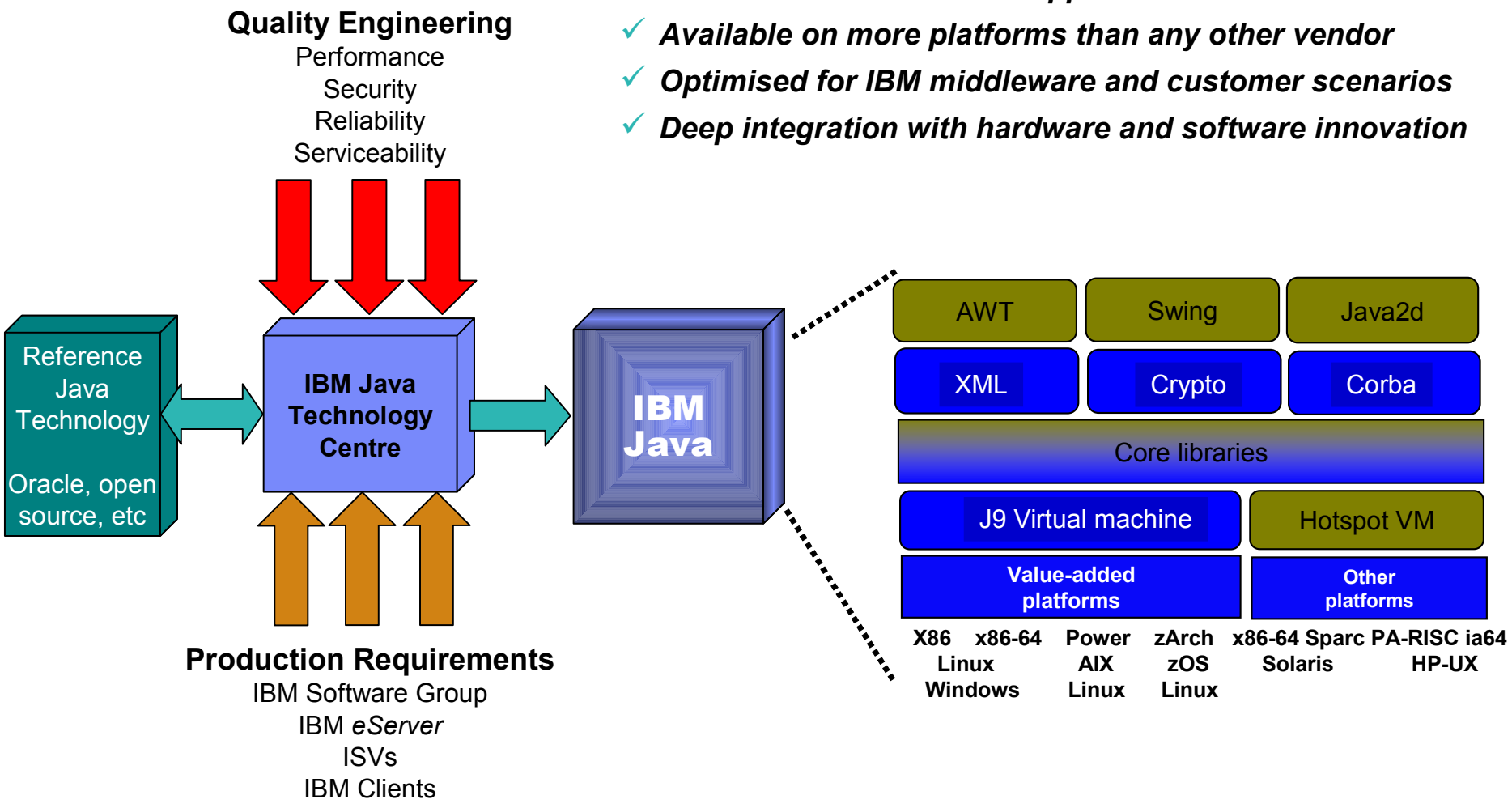
**Tim Ellison**
**Senior Technical Staff Member**
**IBM United Kingdom Ltd.**

Introduction to

# IBM Java SE

# IBM's approach to Java SE technology

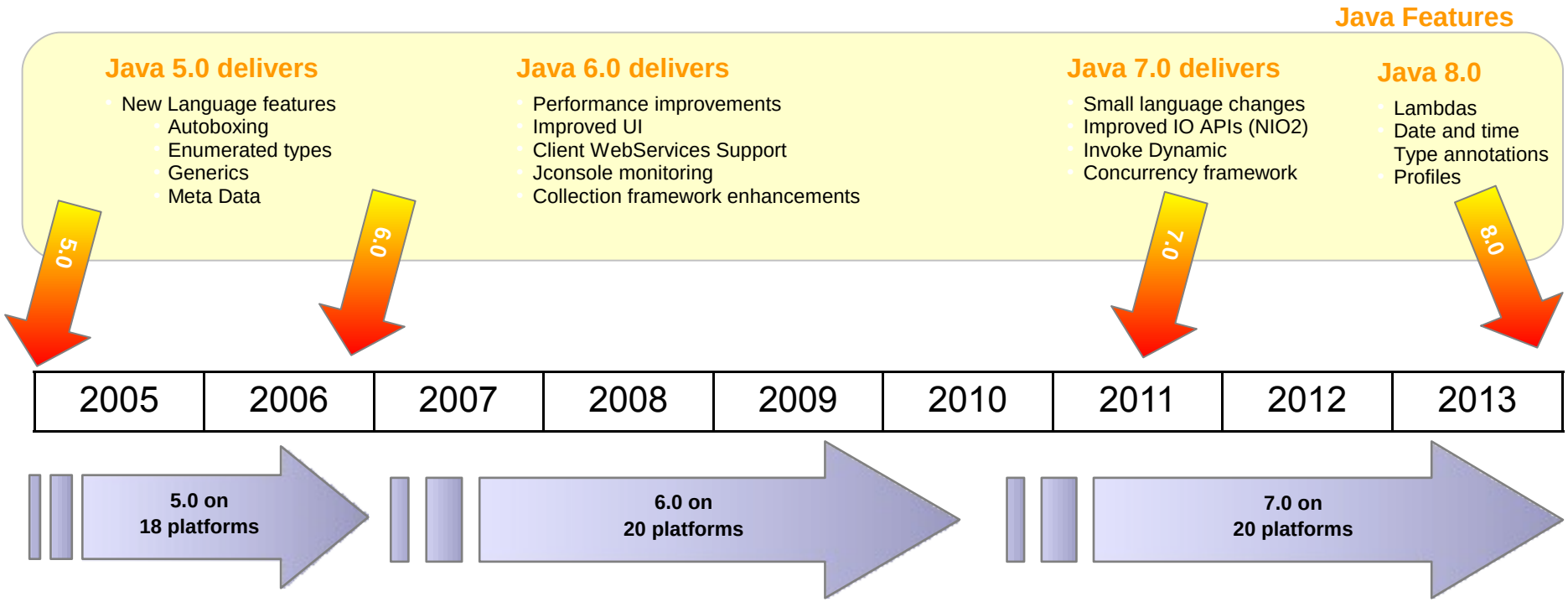
- ✓ ***World class service and support***
- ✓ ***Available on more platforms than any other vendor***
- ✓ ***Optimised for IBM middleware and customer scenarios***
- ✓ ***Deep integration with hardware and software innovation***



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

- **Performance**
  - Performance is key for 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
  - Pluggable interfaces with different implementations to match target
  - Class library independence

# Bits of history




## IBM Java 5.0 features

- 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 features

- Improvements in
  - Platform coverage
  - Performance
  - Serviceability tooling
- New Functionality
  - IBM WebSphere Real-Time V1.0

## IBM Java 7.0 features

- Improvements in
  - Start up performance
  - Throughput performance
  - New Balanced GC
  - New feature in serviceability tooling
  - Soft Realtime evaluation
  - Performance exploitation of POWER7 and z196

**IBM Java Features**

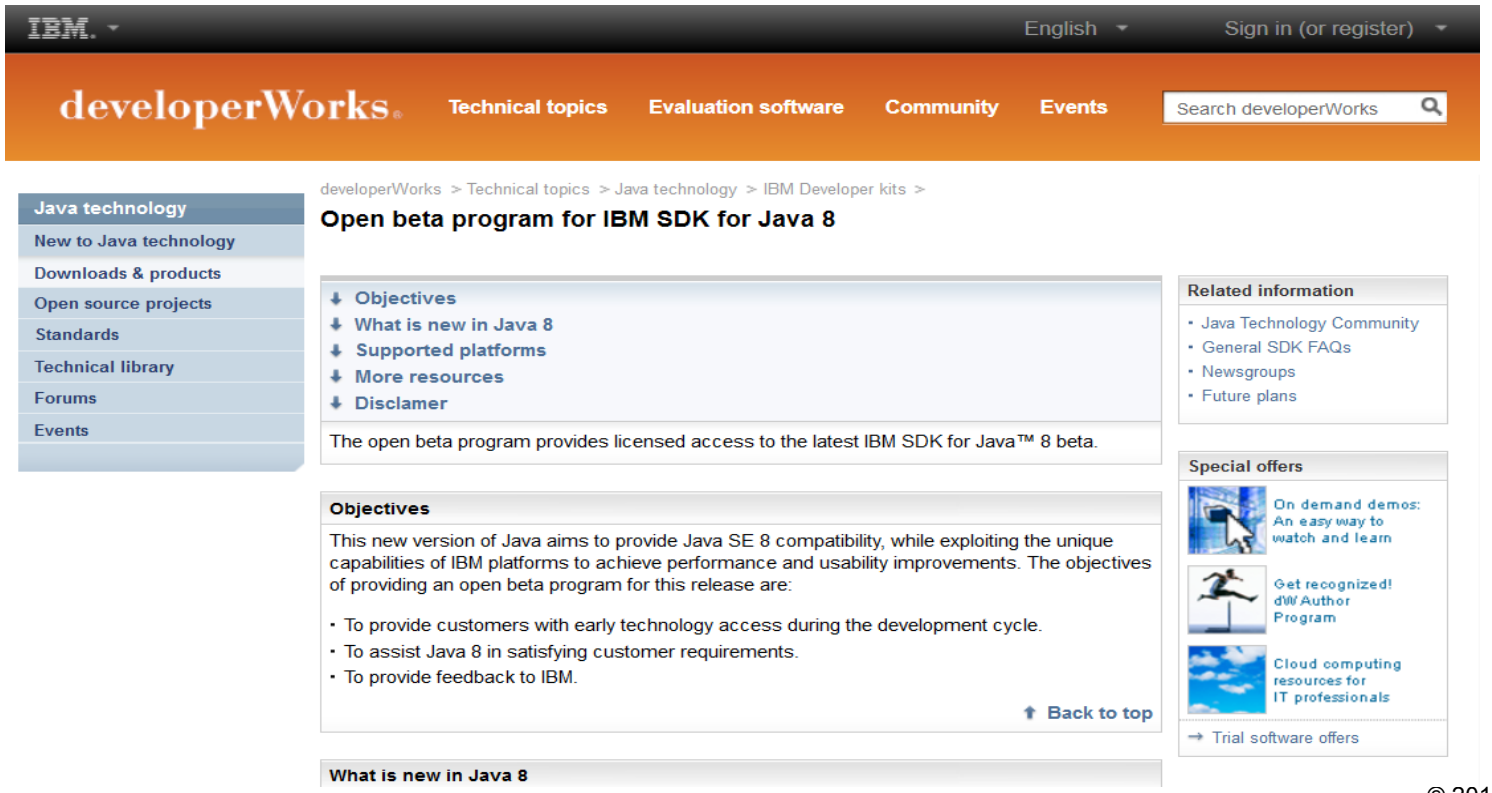
A closer look at

# IBM unique features

# IBM Java innovations – available now! (beta)

- Continuously delivering beta builds of IBM SDK for Java 8
  - Linux, AIX, zOS on a variety of hardware
- Open beta programme is an opportunity to feedback into development and influence the direction we take

**http://www.ibm.com/developerworks/java/jdk/beta/**



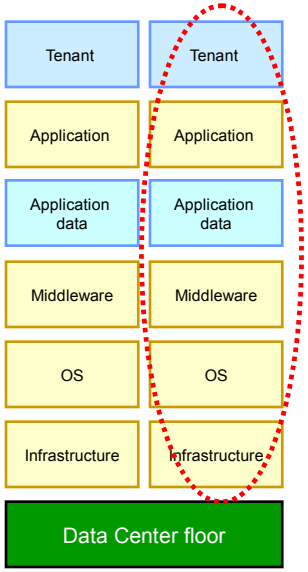
# IBM Innovation

- Cloud and virtualization
- Data access performance and interoperability
- Diagnostics and service tooling
- Performance

# Increasing levels of JDK support for virtualization

Application specific

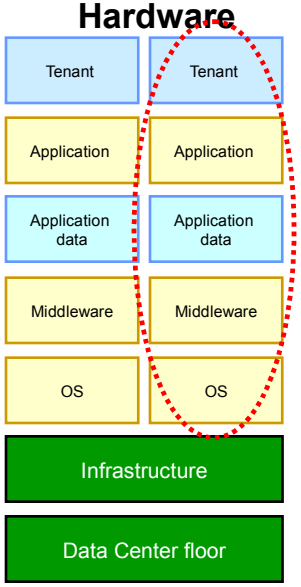
## No Sharing




*Sharing servers storage, networks in a data center*

**Common approach**
- Data isolation
- Resource management

## Shared Hardware

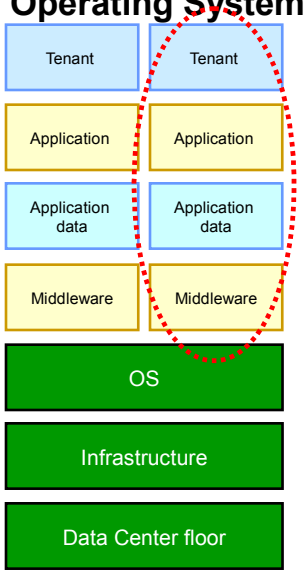



*Hypervisors (e.g. KVM, VMWare) are used to virtualize the hardware*

**Class Sharing (read-only)**
- Packaged in each O/S
- Depends on hypervisor page deduplication

**Dynamic Heap Size**
- React to LPAR resize
- Adjustable via JMX

## Shared Operating System

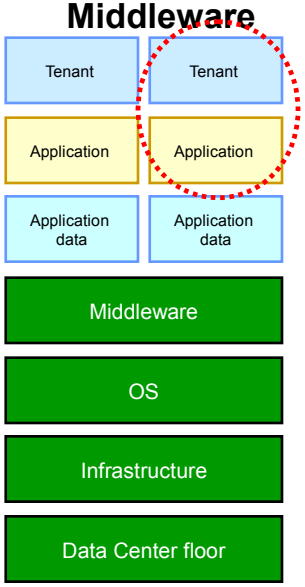



*Multiple copies of middleware in a single operating system*

**Class Sharing (r/w)**
- Share 10's of MB / JVM
- Share data & JIT code

**Extended JMX Beans**
- Hypervisor awareness
- Access O/S perf. stats
- Right-size, react to △'s

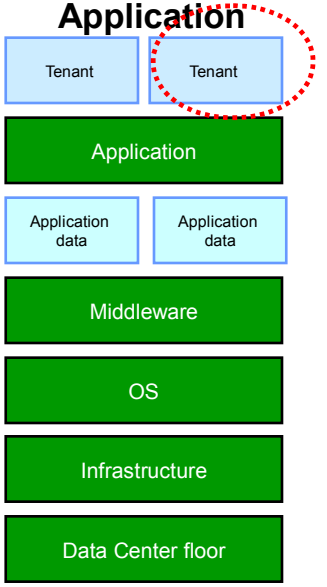
## Shared Middleware




*Multiple applications sharing the same middleware*

**Multitenant JDK**
- Multiple apps (Liberty) share a process
- Resource management and throttling
- Transparent via –Xmt command-line
- Up to 5x less memory, 2x faster startup

## Shared Application



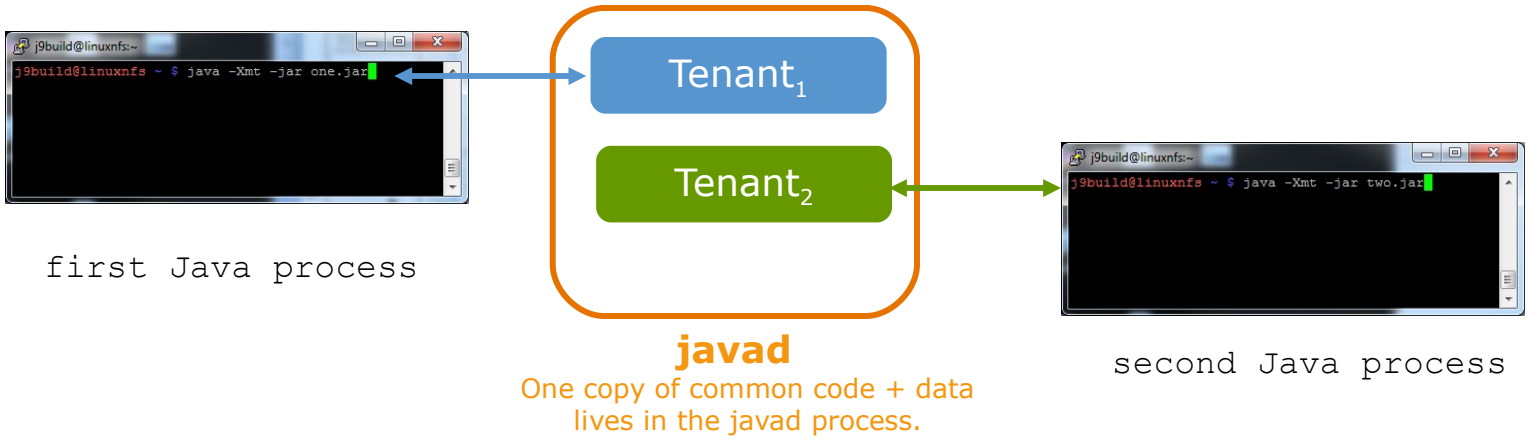

*Sharing the same application*

**Tenant API**
- Java API for tenant management
- Fine grained static field isolation (@TenantScope)

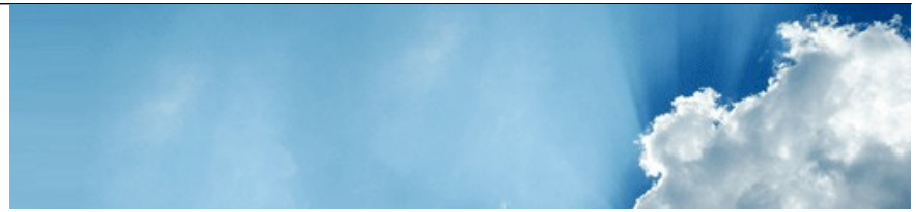
# IBM Java SDK: Multi-tenancy

- Multitenancy is all about reducing duplication by transparently sharing JVM components
  - Multiple applications (tenants) run with a single GC, single JIT, shared heap objects
  - Tenants are isolated from one another
  - plus: JVM-enforced resource constraints to meter and limit tenant resource consumption

- Enable multitenancy with a single flag: **-Xmt** (**m**ulti**t**enancy)
  - -Xmt tells the VM to look for and share a "javad" service
  - *no application changes* required



# IBM Java SDK: Cloud support

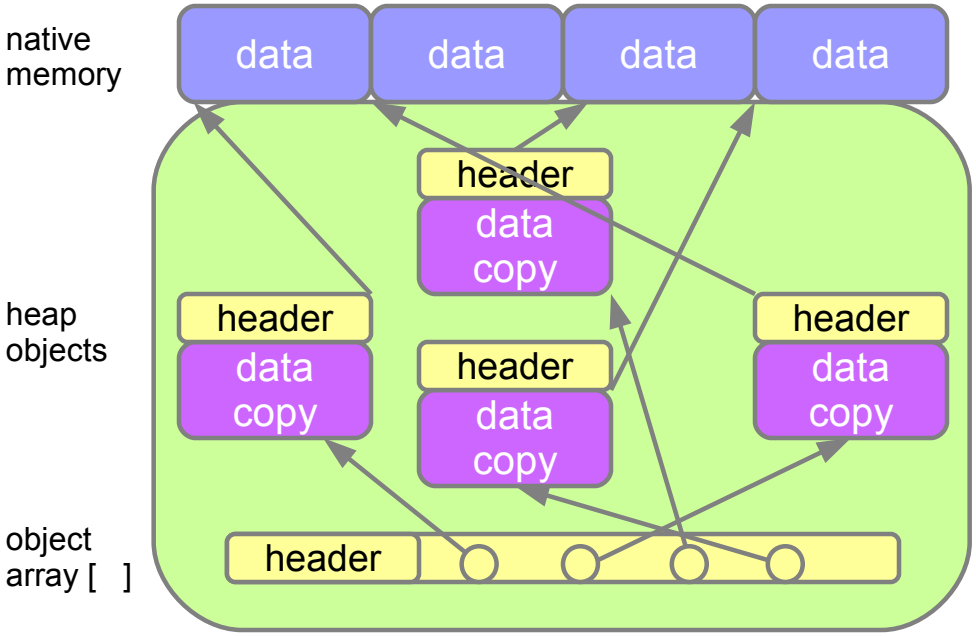


- **-Xtune:virtualized includes a 'deep idle' mode for the JIT**
  - Reduces background JIT activity when the application is idle by ~85%

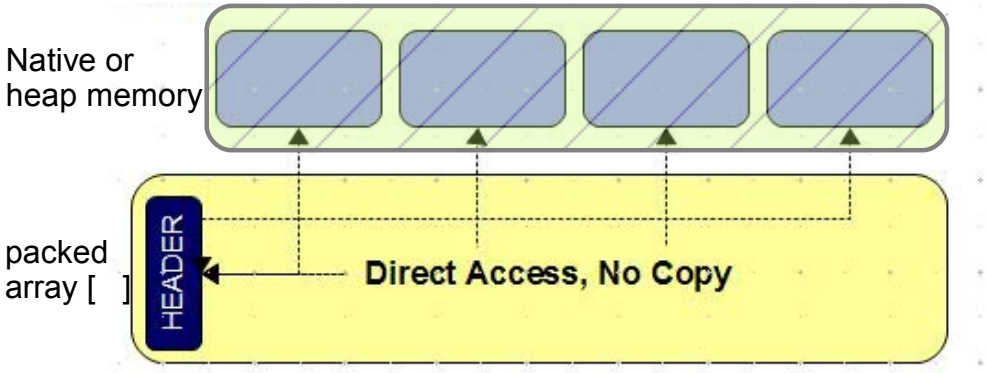
- **Improved OperatingSystemMxBean**
  - New operating system queries supported to allow applications to adjust to current load conditions as dynamic situation changes
  - New API includes:
    - processCpuLoad()
    - getFreeSwapSpaceSize()
    - getTotalSwapSpaceSize()

- **-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

# IBM Java SDK: Packed objects support

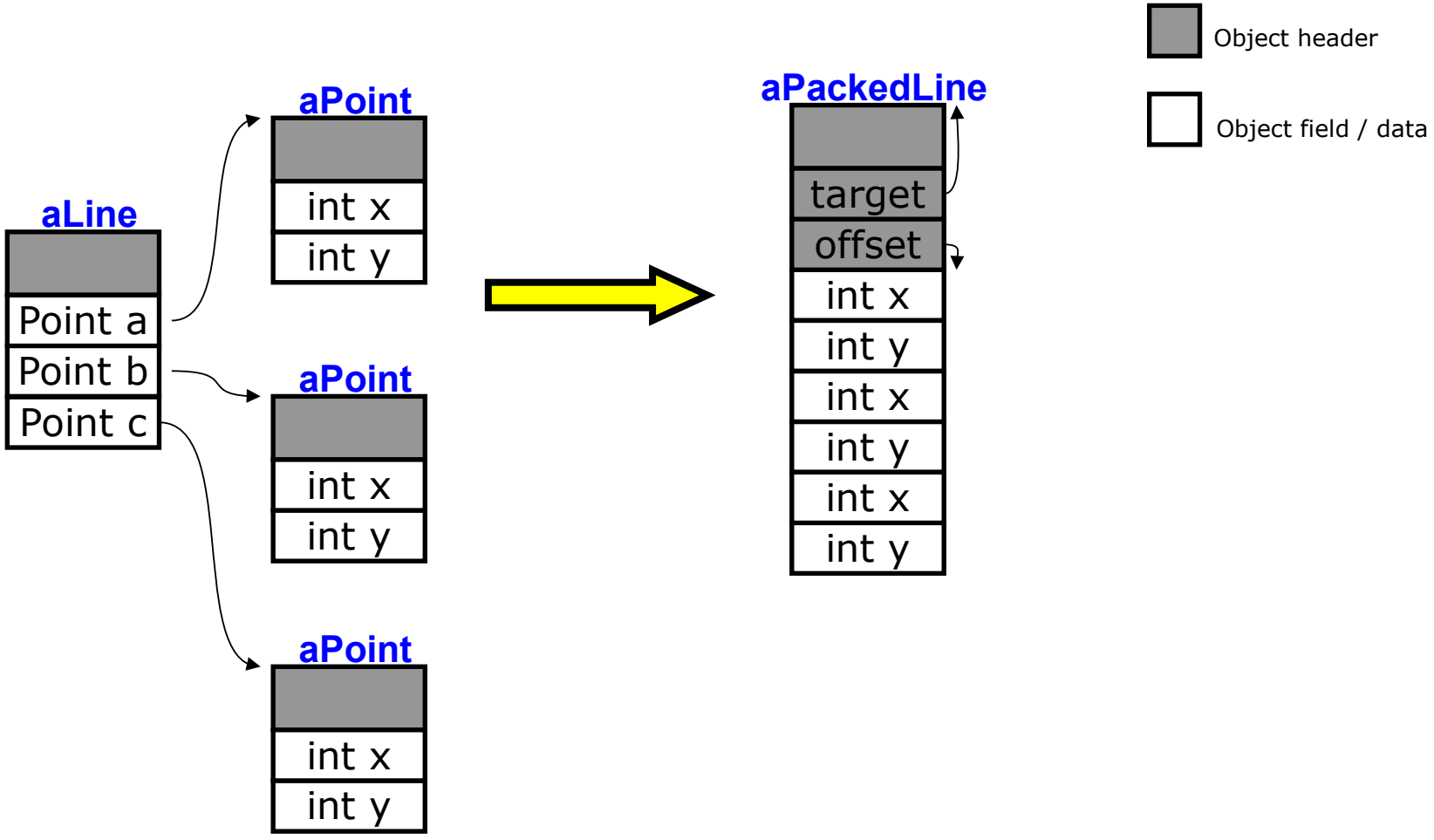



- Java requires memory to be in Java "object" form to be accessed directly
- External data needs to be read into Java heap format to use – conversion is expensive
- Memory bloat occurs due to data copies and headers
- Natural object representation looses data locality properties

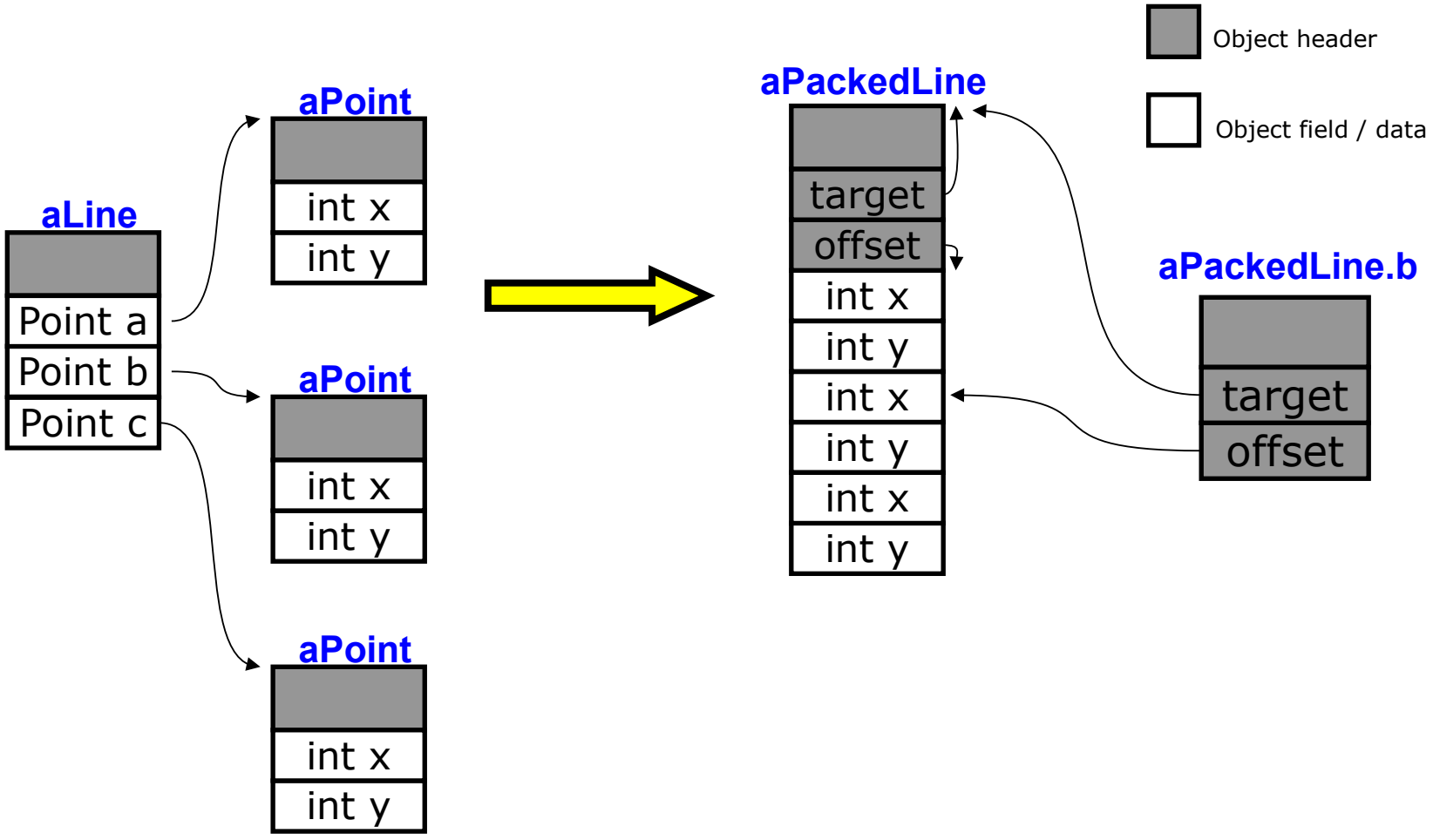



- PackedObjects enables direct access to data in arbitrary formats without the redundant copying; no conversion
- PackedObjects data can be in native memory or Java heap space

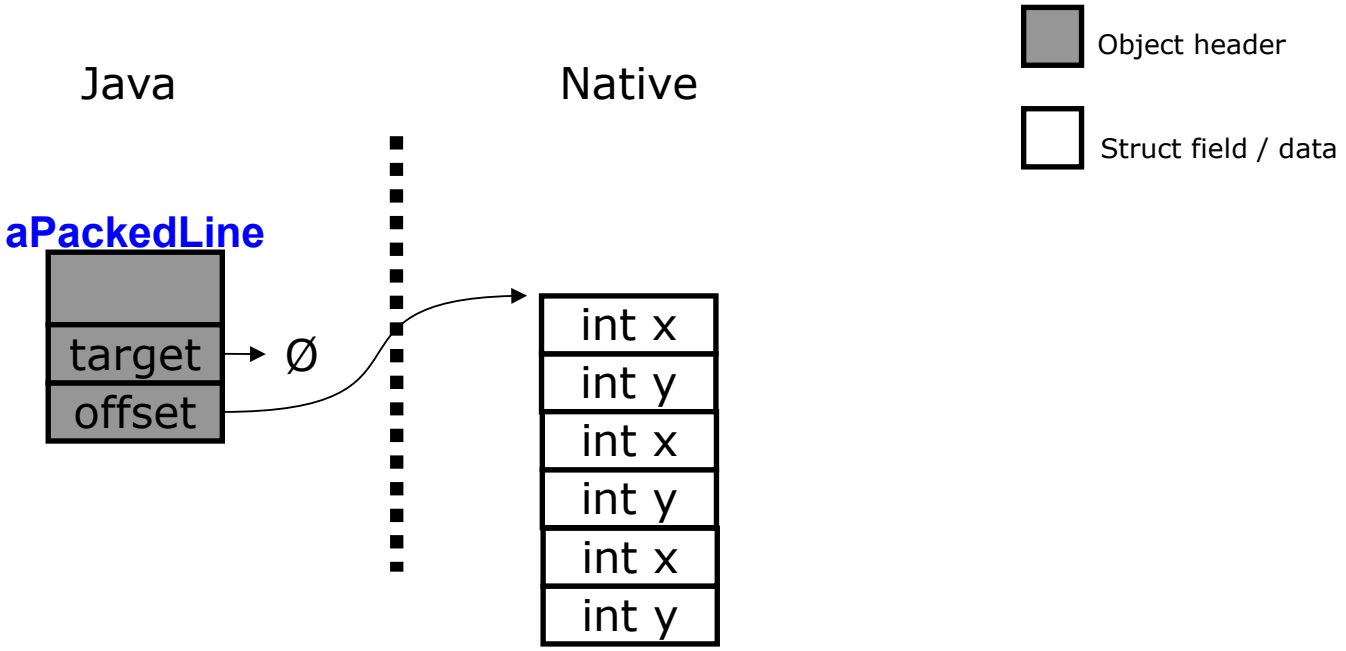
# Packed Objects: Heap referenced data



# Packed Objects: Heap referenced data



# Packed Objects: In Practice with Native Access



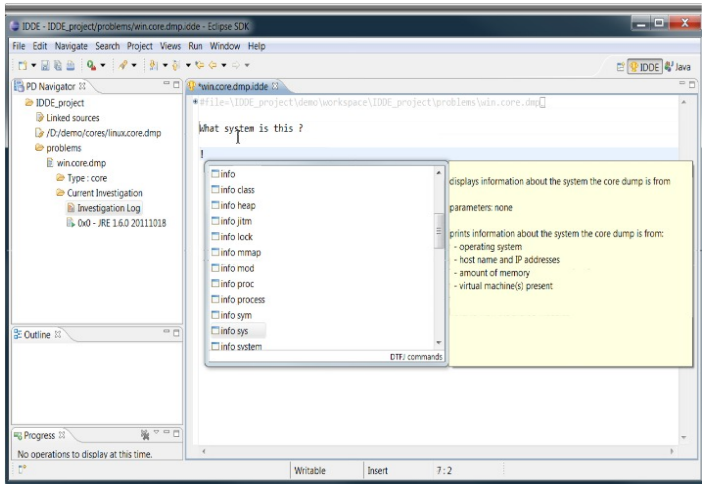

```
@Packed
final class PackedPoint extends PackedObject {
    int x;
    int y;
}
```

# IBM Java SDK: 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:

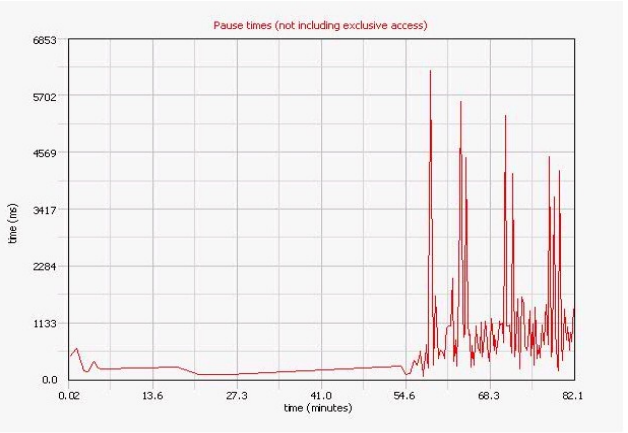



**Interactive Diagnostic Data Explorer**

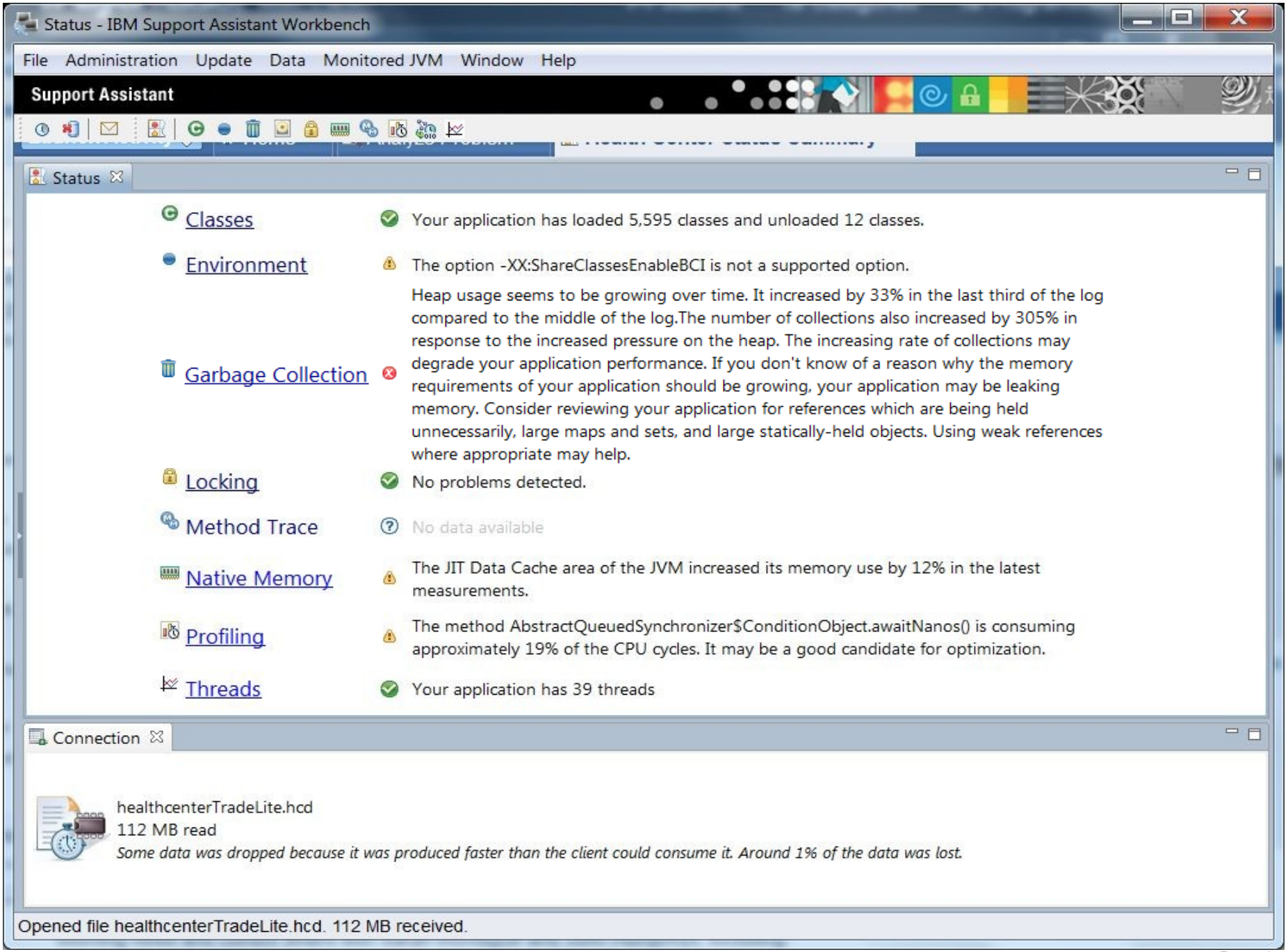
**Garbage Collection and Memory Visualizer**

**Memory Analyser**

**Health Centre**




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

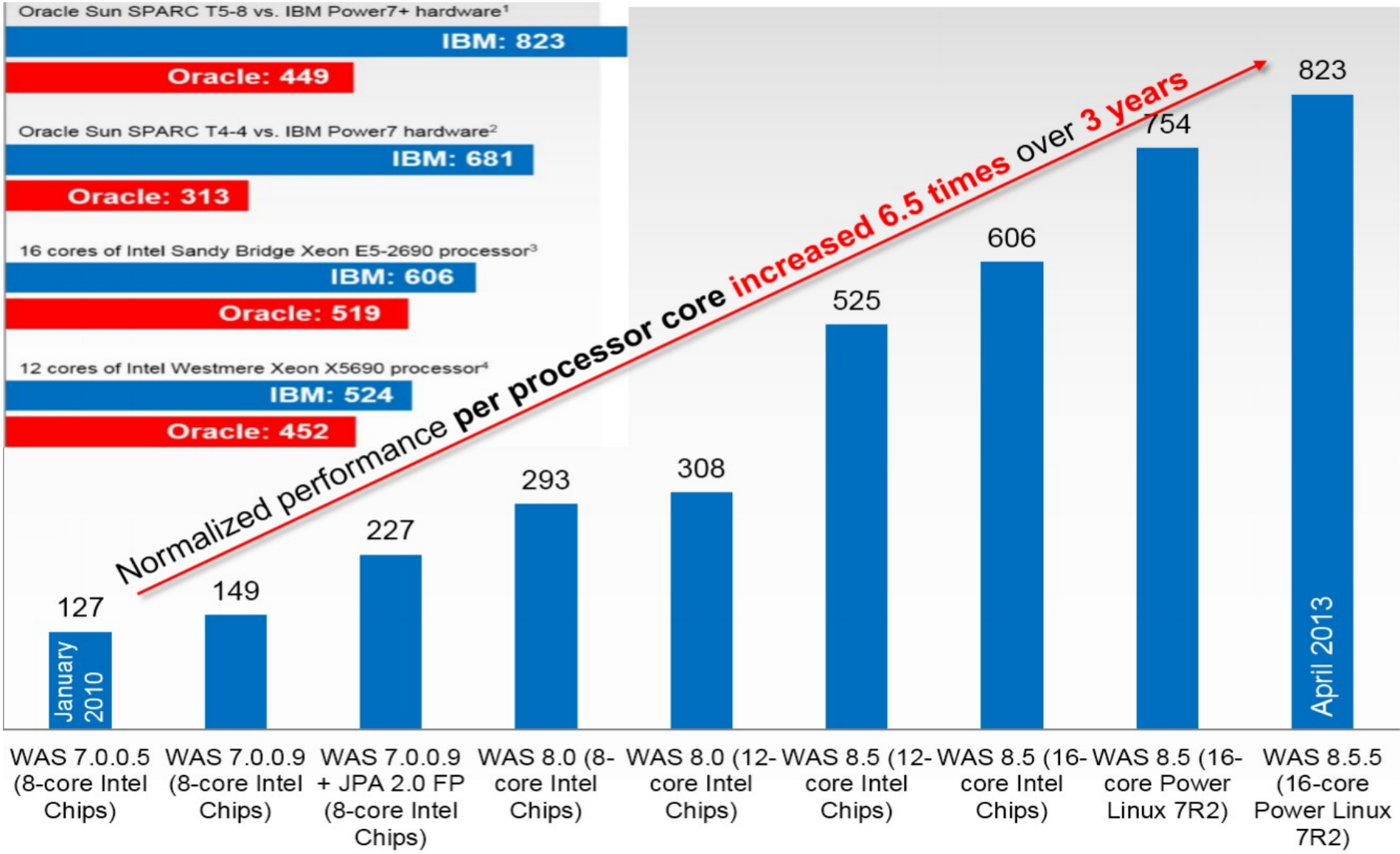

Status - IBM Support Assistant Workbench
File Administration Update Data Monitored JVM Window Help
Support Assistant
Status
Classes
Your application has loaded 5,595 classes and unloaded 12 classes.
Environment
The option -XX:ShareClassesEnableBCI is not a supported option.
Garbage Collection
Heap usage 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.

# IBM Java SDK: Performance

- IBM Java team works closely with the hardware designers and middleware architects
  - Defining new capabilities for Java across System x, POWER, System z, …
- We can influence, and are influenced, by hardware and software to optimize the IBM stack for customer workloads
  - Pure Application Systems, Liberty, WebSphere Application Sever, key benchmarks (e.g. SPEC), …


- Expect to see...
  - Huge improvements to scripting performance
  - GC scalability improvements on large systems (>= 64 processors)
  - Deeper platform exploitation:
    - Large page enablement by default for platforms that support it, including selectable page sizes
    - Large page support on z/OS, 1m pageable and 2Gb
    - Compressed references on by default (32-bit like performance and footprint on 64-bit workloads for heaps ~25Gb and under)
  - Faster debugging and class file re-transformation
  - Reduced memory footprint when re-transforming classes in hot deploy and debug

# IBM Java SDK: Powering WebSphere Performance




1 As per SPEC Published Data as of 4/26/2013: http://www.spec.org/jEnterprise2010/results/jEnterprise2010.html

# IBM Java SDK: Packed Objects Usage

```
// non-packed class
abstract class Event extends PackedObject {
        public EventHeader header;
}

packed class ExposeEvent extends Event {
        // packed inherited field "header" is nested

        // a nested p-array field
        public Point[[2]] extent;

        ExposeEvent(short id, Point[[]] init_extent) {
                // assignment-by-value
                extent[[0]] := init_extent[[0]];
                extent[[1]] := init_extent[[1]];

                // invoke nested field's constructor
                header.new(EXPOSE, id);
        }
}
```