





CON7357

Why Should I Switch to Java SE 7?

Dave Keenan, Java Performance Architect, Oracle
Staffan Friberg, JVM Product Manager, Oracle

MAKE THE
FUTURE
JAVA



The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Agenda

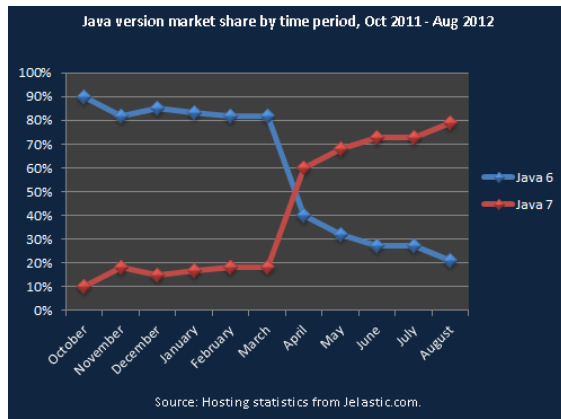
- Mature and Recommended
- New Features
- Performance Benefits
- Backwards Compatibility
- Summary

Mature and Recommended

Deployed and Used

Fast and increasing uptake

- 23% of the respondents to ZeroTurnaround's study use JDK 7^[1]
 - Less than 6 months after Java SE 7 was released!
- Jelastc's latest report show 79% of the deployments in their Java PaaS use JDK 7^[2]



1) <http://zeroturnaround.com/labs/developer-productivity-report-2012-java-tools-tech-devs-and-data>

2) <http://blog.jelastc.com/2012/09/12/software-stack-market-share-august-2012>



Mature

One year since GA

- Tried and tested with a huge set of applications and libraries
 - Java and OpenJDK communities
 - Java ISVs
 - Oracle's Java products
- 7 update releases
 - Bugs and security fixes
 - Performance enhancements
 - New features



Certified and Recommended

- Certified and supported by all major
 - IDEs and development tools
 - Java EE application servers
- Recommended JDK
 - New deployments – both server and client applications
- Auto-update coming soon
 - Already default JRE on java.com



Certified Platforms

Oracle JDK 7

- Windows – x86 and x64
- Linux – x86, x64 and **ARM v6 & v7**
- Solaris – x86, x64, SPARC and SPARC v9
- **Mac OS X – x64**

<http://www.oracle.com/technetwork/java/javase/config-417990.html>

Support

Free Public Updates

- Java SE 6
 - Public updates of Java SE 6 ending in February 2013
 - Updates available via support contract
- Java SE 7
 - Public updates available until at least July 2014
- Java SE public updates
 - a) 3 years after it has been made generally available
 - b) 1 year after a subsequent major release
 - c) 6 months after a subsequent major release has become the default JRE

<http://www.oracle.com/technetwork/java/eol-135779.html>



Java SE 7 Implementations

- OpenJDK
 - Most of our development work is done in OpenJDK
- Java SE Licensees
 - SAP, HP, Fujitsu
- IBM
 - New JVM features
 - Balanced GC
 - Optimized class libraries



New Features

Development

Focusing Resources

- Feature and performance development is targeted for
 - JDK 7 update releases
 - JDK 8
- JDK 6 not completely frozen
 - Security and bug fixes are always back-ported

Serviceability Features

JRockit / HotSpot Convergence

- Java Mission Control
 - Monitor, manage, profile
- Java Flight Recorder
 - Profiling, problem analysis, debugging
 - Partial implementation in JDK 7
 - Oracle Fusion Middleware probes only

Java Diagnostic Commands

JDK Introspection

- Complete tool chain
 - Framework for easy implementation and integration
 - Command line tool, jcmd
- jcmd
 - List running Java processes
- jcmd <pid> help
 - List all available commands, currently ~15 different commands

Diagnostic Commands

Class Histogram

```
jcmd <pid> GC.class_histogram
```

num	#instances	#bytes	class name

1:	5466	9460256	[I
2:	54844	7491712	<constMethodKlass>
3:	54844	7474144	<methodKlass>
4:	4722	5887584	<constantPoolKlass>
5:	4722	4133992	<instanceKlassKlass>
6:	4091	3663584	<constantPoolCacheKlass>
7:	27380	2352496	[C
8:	12229	2181656	[B
9:	27066	649584	java.lang.String
10:	5082	622520	java.lang.Class
...			



G1 – Garbage First

Recommended Use Cases

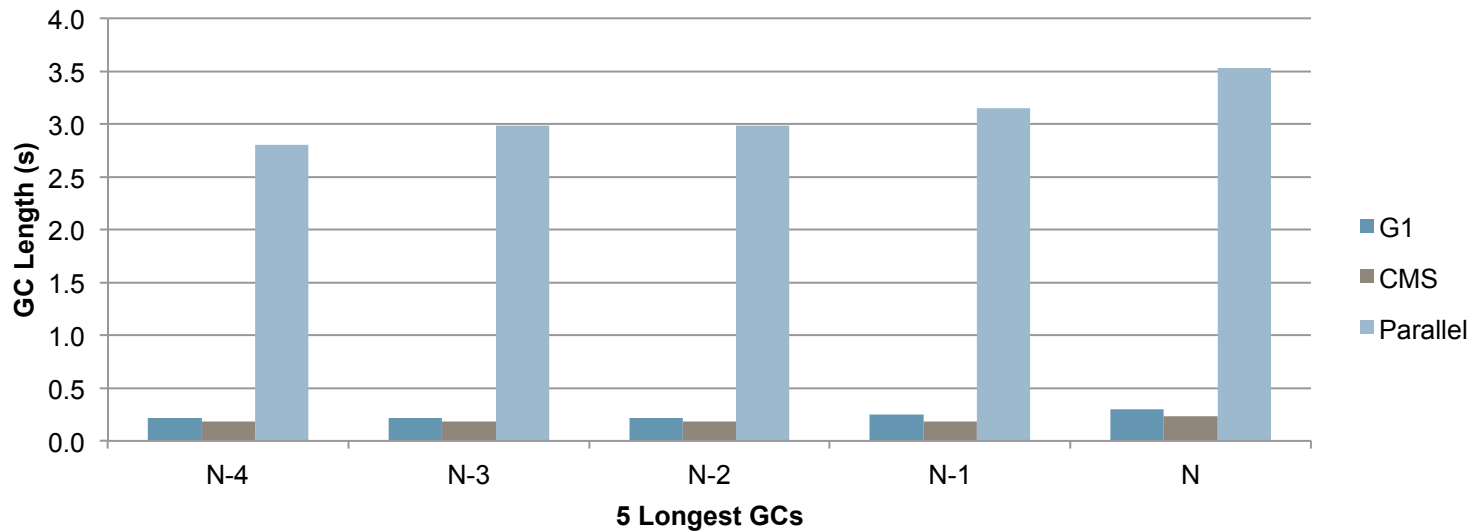
- Supported for production use since 7u4
- Large heaps with GC latency requirements
 - Typically ~6GB or larger heaps
 - Stable and predictable GC latencies below 0.5 seconds
- Applications that have
 - more than 50% live data on the heap
 - varied object allocation rate
 - undesired long GC or compaction pauses (greater than 0.5s)



G1 – Garbage First

Performance

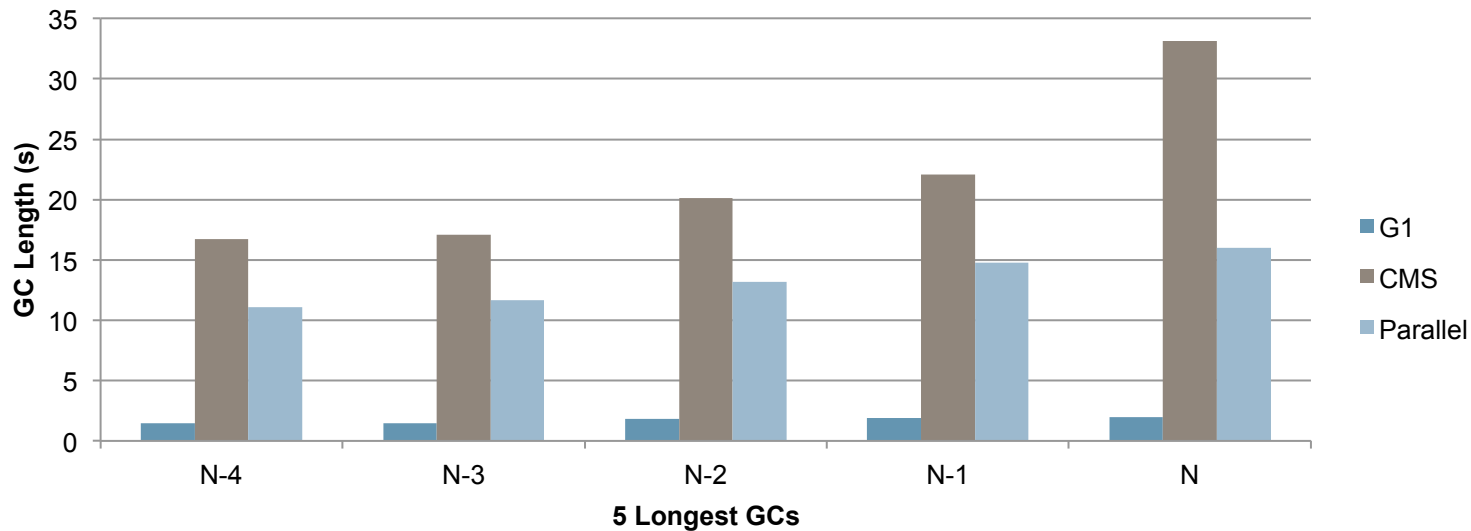
Typical Java EE Application



G1 – Garbage First

Performance

Typical Java EE Application with Fragmentation



Runtime Compiler Improvements

Performance

- TieredCompilation
 - Interpreter -> Client Compiler (C1) -> Highly Optimized Server Compiler (C2)
 - Not on by default
- Tuning challenges
 - Current heuristics are insufficient
 - Combined C1+C2 compiled code puts pressure on ReservedCodeCache



Sockets Direct Protocol (SDP)

Infiniband Support

- Transparent support for IB networks using regular Sockets
 - `java.net` and NIO
- SDP support is disabled by default
 - Create an SDP configuration file.
 - Set the system property that specifies the location of the configuration file.
- <http://docs.oracle.com/javase/tutorial/sdp/index.html>



Performance Benefits

Performance Features

Java Class Libraries

- Avoid Contention in Date
 - Changed from Hashtable to ConcurrentHashMap
- BigDecimal improvements, CR 7013110
- Copy elision in many of the sunpkcs JNI calls, CR 6988081

Performance Features


Java Class Libraries

- Crypto configuration file updates, CR 7036252
 - Improves out of the box crypto experience
- User land crypto for SPARC T4, CR 7013347, (uses T4 crypto instructions), targeted for 7u2
 - Much faster crypto on SPARC T4 and later
- Adler32 & CRC32 to leverage T-series hardware CRC, CR 7039848, possibly implement similar intrinsic approach being implemented for “user land crypto”, targeted for 7u2



Performance Features

Java Class Libraries

- `String(byte[] bytes, String csn)` and `String.getBytes(String csn)`
 - Optimized `String char[]`  `byte[]` conversion
 - 2 – 3x performance improvement in micro-benchmarks with small Strings
- Java Concurrency APIs (JSR 166y)
 - New improvements in JDK 7
 - Fork-join Framework



Performance Features

JSR 166y: Fork Join Framework

- Goal is to take advantage of multiple processor
- Designed for task that can be broken down into smaller pieces
 - Eg. Fibonacci number $\text{fib}(10) = \text{fib}(9) + \text{fib}(8)$
- Typical algorithm that uses fork join

`if I can manage the task`

`perform the task`

`else`

`fork task into x number of smaller/similar task`

`join the results`



Performance Features

HotSpot JVM

- Updated native compilers
 - gcc 4.2
 - Oracle Studio
- -XX:+UseNUMA on Java 7
 - Linux kernel 2.6.19 or later
 - glibc 2.6.1



Performance Features

HotSpot JVM: Partial Perm Gen Removal

- Partial Permanent Generation removal in JDK 7
 - Interned Strings moved to Java heap
- First step: Full removal in JDK 8
- May be necessary to adjust GC tuning
 - Interned Strings now in heap
 - Applications with high GC time
 - Applications with high interned String count
 - Larger heap size may be needed



Performance Features

HotSpot JVM: Internal JVM Data Structures

- Interned Strings
- Distinct class names loaded
- Hashtable implementation
 - Default table size is 1009
 - Significant performance impact if more than 1009
- Increase sizes if needed:
 - Interned Strings: `-XX:StringTableSize=n`
 - Distinct Class names: `-XX:+UnlockExperimentalVMOptions`
`-XX:PredictedClassLoadCount=#`



Performance Features

Client Library Updates

- Nimbus Look and Feel
- Platform APIs for shaped and translucent windows
- JLayer (formerly from Swing labs)
- Optimized 2D rendering
- Improved Xrender support on Linux



Performance Features

JDBC 4.1 Updates

- Allow `Connection`, `ResultSet` and `Statement` objects to be used with the try-with-resources statement
 - Implement `AutoCloseable` interface
- `RowSetFactory` and `RowSetProvider` classes added to `javax.sql.rowset` package



Performance Features

XML APIs Updates

- JAXP 1.4.5 (Parsing)
 - Bug fixes, conformance, security, performance
 - StAX upgraded to version 1.2
- JAXB 2.2.3 (Binding)
 - Minor changes to `XMLRootElement` interface
- JAX-WS 2.2.4 (Web Services)
 - Minor changes
 - Support for Async Servlet Transport using Servlet 3.0 API



Performance Features

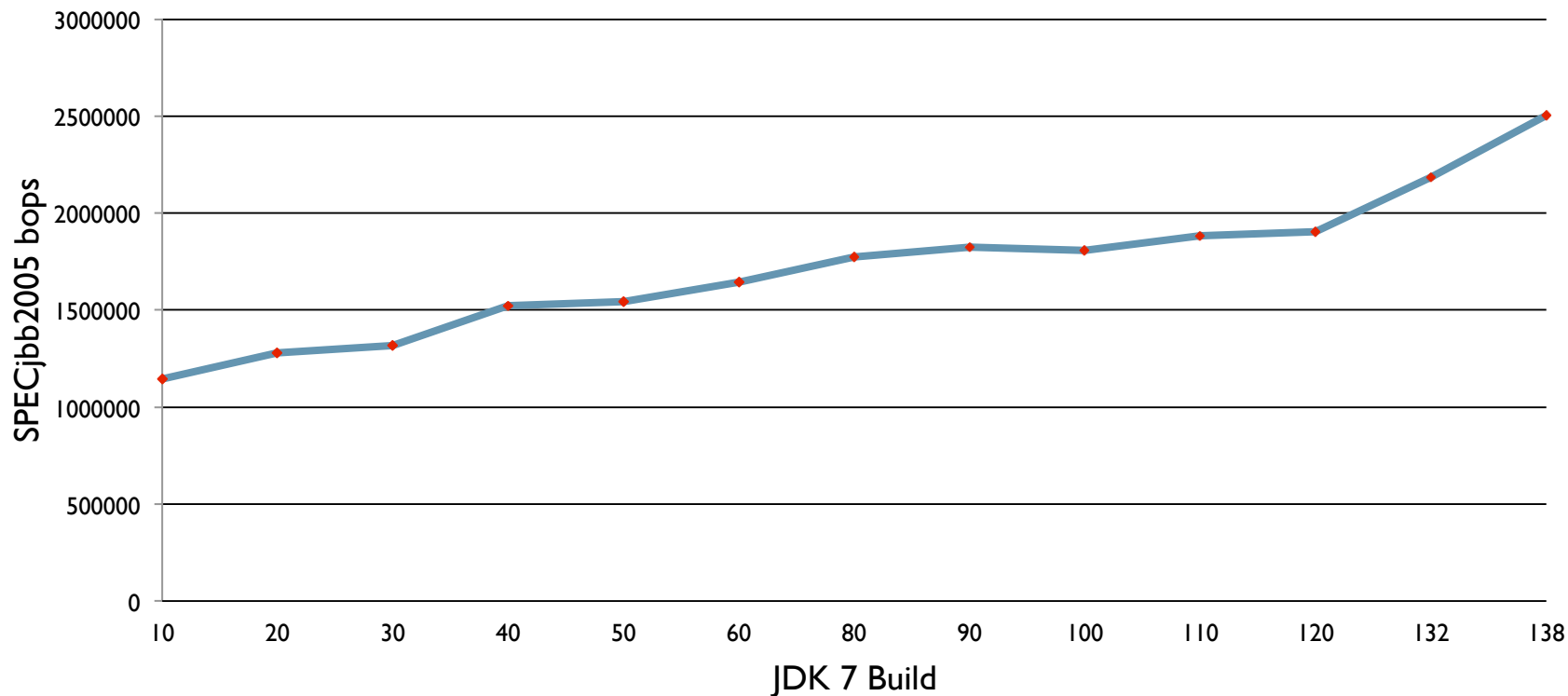
Java Class Libraries – Asynchronous I/O

- `java.io` a short history
 - Pre JDK 1.4 – blocking I/O
 - JDK 1.4 onwards – non blocking I/O, memory mapped files, file lock, channels and buffers
 - JDK 7 – asynchronous I/O
- Provides asynchronous I/O to both sockets and files
 - Take advantage of operating systems I/O facilities where available



Java 7 Performance

SPECjbb2005 throughout Java 7 Development Builds

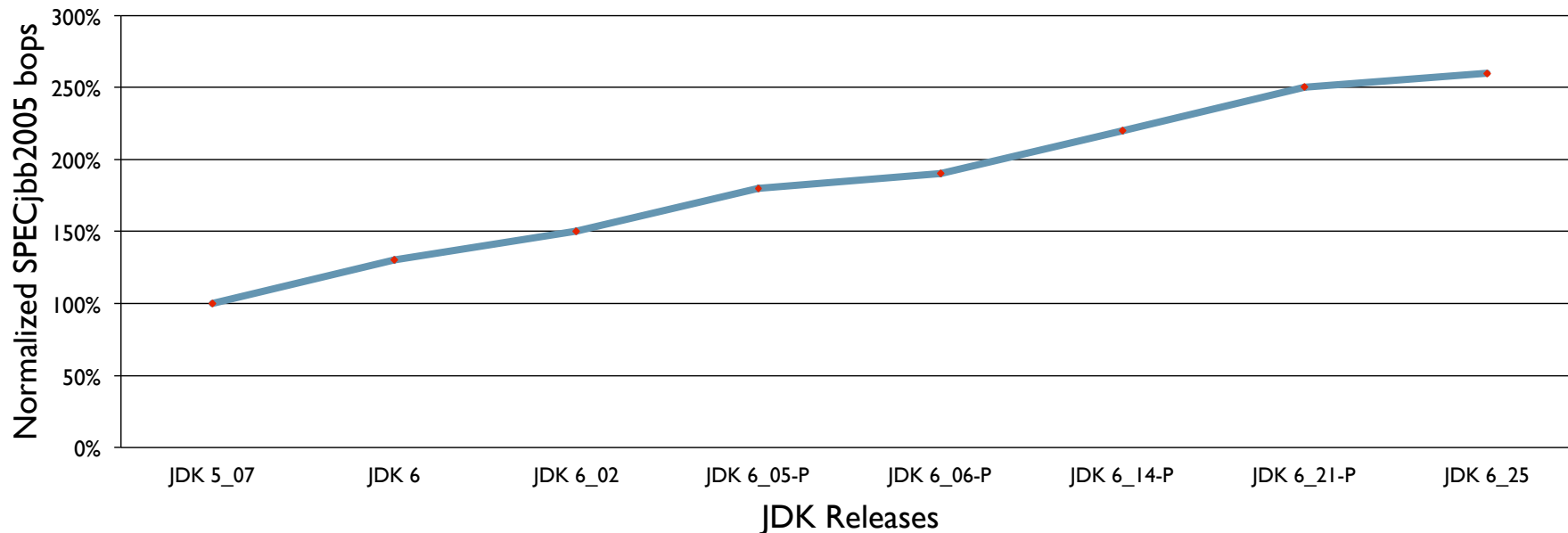


- 4x2.4GHz WSM-EX, Oracle Solaris 11 Express snv_156 X86
- 2.2x Improvement through JDK 7 development



Java 7 Performance

SPECjbb2005 Improvement through JDK optimization

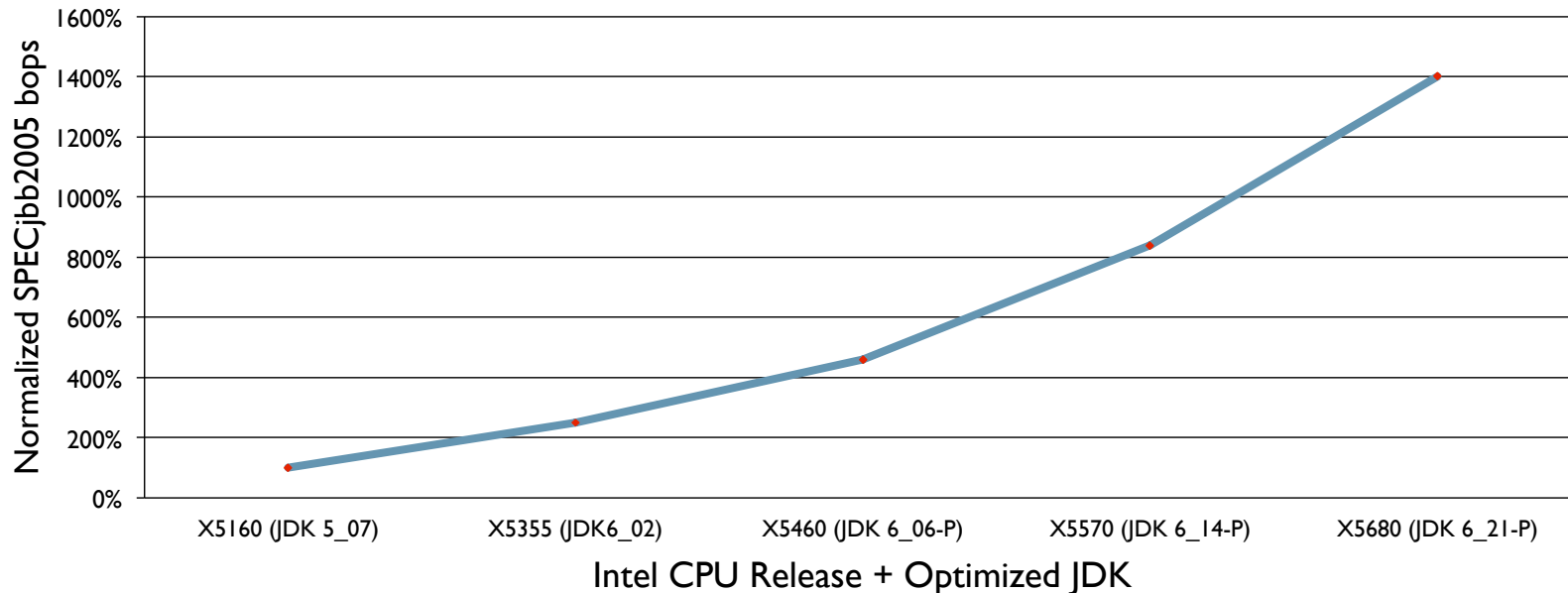


- 2x3.33GHz WSM-EP-B1, Windows 2008 Server Enterprise
- **2.5x performance improvement over the last three years**



Java 7 Performance

SPECjbb2005 Improvement through JDK optimization



- **14x HW + SW improvement over 5 Intel CPU Releases**

Java 7 Performance

Competitive Landscape

- SPECjbb2005
 - JDK 7 (IBM J9) is current 4-CPU Performance leader
- SPECjEnterprise2010
 - JDK 7 (Oracle HotSpot JVM) is current performance leader on x86_64
- SPECpower_ssj2008
 - JDK 7 (IBM J9) is amongst the power/performance leaders

Java 7 Performance

Real World Performance

- JDK 7 Update 4 and Later
 - Faster than Java 6 for 97% of Applications tested
 - Faster than JRockit for 96% of Applications tested

Backwards Compatibility

Backward Compatibility

- **Source compatibility** – old Java source code continues to compile
- **Binary compatibility** – old Java class files continue to link
- **Behavioral compatibility** – execution generates the same result
- Java SE 7 and JDK 7 Compatibility
 - <http://www.oracle.com/technetwork/java/javase/compatibility-417013.html>



Heap Retuning

- First iteration of PermGen removal
 - Reduce memory usage in PermGen
 - Interned and final Strings moved to regular heap
 - Impact: Fine tuned PermGen and Heap sizes might not be optimal

Main Pitfalls

What people have experienced

- More stringent bytecode verifier for Java SE 7 class files
 - Mainly an issue when doing bytecode modification
 - Most applications already updated to generate correct code
 - Work around: -XX:-UseSplitVerifier
- Order of methods returned from `getMethods()` changed
 - Was always specified to be unspecified
 - Most affected applications already updated
 - Work around: None



Summary

Summary

Why Should I Switch to Java SE 7?

- Mature and proven release
- Continued free public updates
- Several new features and performance improvements
- Recommended and fully supported release

Final Comment

After Updating to Java SE 7

- Now when you are running Java SE 7,
why not take a look at the new language features
- A sample of the sessions this year
 - TUT4629 – JDK 7 in Action: Using New Core Platform Features in Real Code, Joe Darcy
 - HOL2846 – The Power of Java 7 NIO.2 by Example, Mohamed Taman
 - CON5357 – Exercising Java 7 Features in Enterprise Applications While Avoiding Pitfalls, Anil Kumar
 - CON7236 – Advanced JVM Tuning, Dave Keenan
- and in the evenings check out last years sessions...
 - <http://www.parleys.com/#st=4&id=102979>



