IBM Software

IBM Java 7 and WAS v8.5: Features and benefits

Andy Clarke
Java Development Manager

Please note

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal 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.

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.

A little bit about me

Andy Clarke is the IBM Java Development Manager at the Hursley Labs near Winchester, UK.

He currently leads the global delivery of all IBM's Java development and has over 10 years experience in roles related to development and support of the IBM SDK for Java.

In his spare time, he rides motorbikes, skis and is passionate about sailing.

A little bit about you

- What's the current knowledge and exposure to Java in the room?
- JSE and/or JEE focused?
- What are you looking to learn during this presentation?

What I'm going to talk about

A little bit about me

A little bit about you

What I'm going to talk about

You are here already ☺

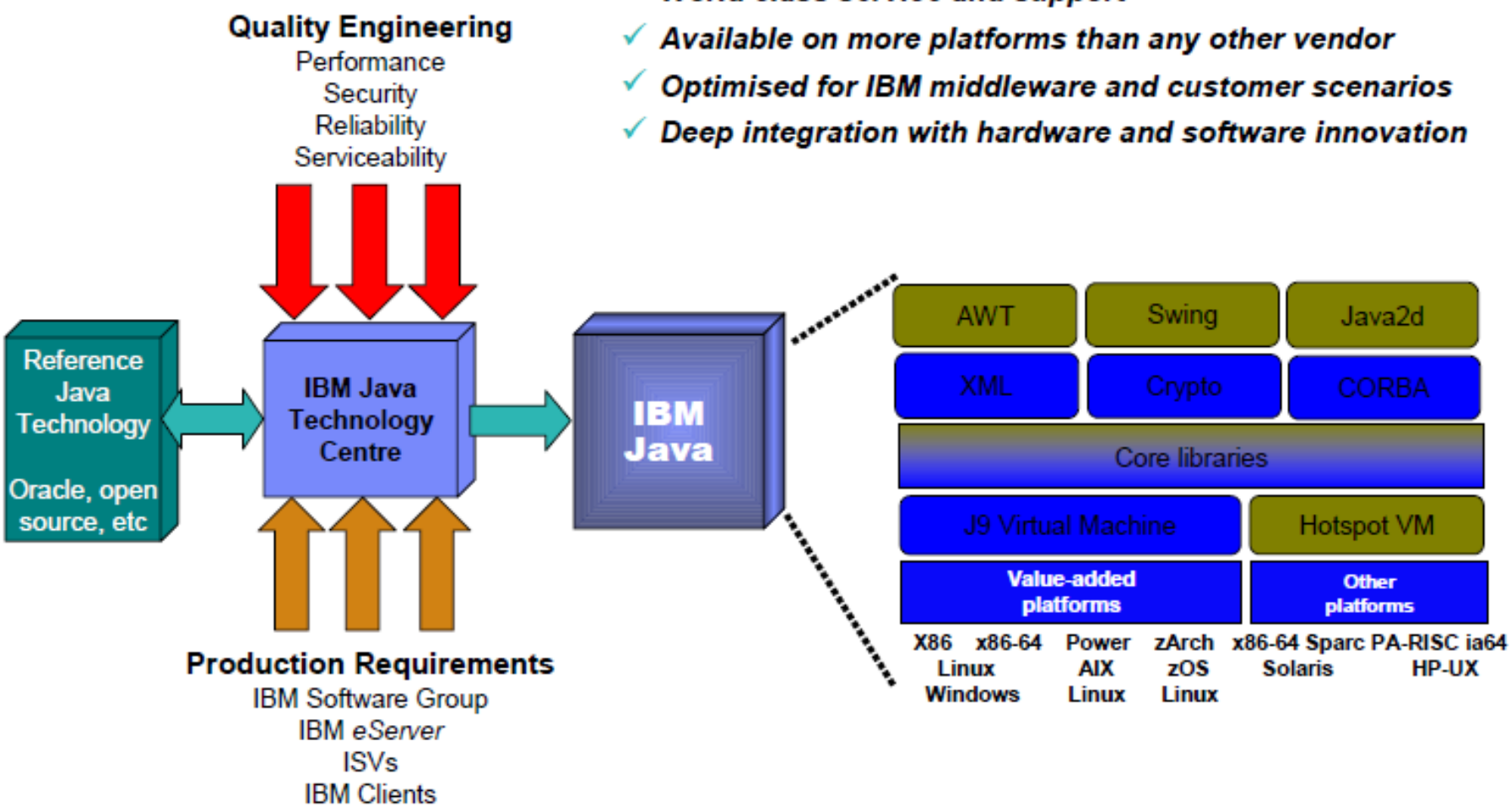
Java 7: What is it? What's new? Base features. IBM features.

WAS 8.5: Java 7 in WAS. "Java version switch" feature. Migration toolkit.

Questions

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*



Java 7: What is it?

Major platform release, touching on all aspects of the language and JVM.
Incorporating the following themes:

Compatibility

"Any program running on a previous release of the platform must also run unchanged on an implementation of Java SE 7"

Productivity

"...promote best coding practices and reduce boilerplate... minimal learning curve..."

Performance

"...new concurrency APIs... enable I/O-intensive applications by introducing a true asynchronous I/O API.."

Universality

"...accelerate the performance of dynamic languages on the Java Virtual Machine."

Integration

"Java SE 7 will include a new, flexible filesystem API as part of JSR 203..."

Java 7: What's new?

Some of the larger features

- JSR 334 – Small language enhancements (Project Coin)
- JSR 203 – More new I/O APIs for the Java platform (NIO.2)
- JSR 292 – New invoke dynamic bytecode
- JSR 166y – Concurrency and collections updates

Plus some smaller features (TLS 1.2, UNICODE 6.0…)

JSR 334 Small language enhancements (Project Coin)

Goal: Improve programmer **productivity** with small enhancements to the Java language and class libraries

- Strings in `switch` statements

```
switch (myString) {
     case “one”: /* do something */ break;
     case “red”: /* do something else */ break;
     default: /* do something generic*/;
   }
```

- Improved type inference for generic instance creation (diamond)

```
Map<String, MyType> foo = new HashMap<String, MyType>();
```

becomes:

```
Map<String, MyType> foo = new HashMap<>();
```

JSR 334 Small language enhancements (Project Coin)

- Better support for literals
  - **binary literals** for integral types (`byte`, `short`, `int` and `long`):
    0b10011010
  - **underscores in numeric literals** to help visual blocking :
    34_409_066 36.100_23f (only between digits though)

- Simplified **varargs** method invocation
  - Moves warnings to method declaration rather than on each use.
    Reduces unavoidable warnings.

JSR 334 Small language enhancements (Project Coin)

- Handle multiple exception types in a single `catch`

```
try {
    …
} catch (ParseException a) {
  handle(a);
} catch (NumberFormatException b) {
  handle(b);
}
```

becomes:

```
try {
    …
} catch (ParseException|NumberFormatException a) {
  handle(a);
}
```

- In this new construct a will act as if declared `final` with type Throwable
  None of the exception classes used may be a subclass of any of the others

JSR 334 Small language enhancements (Project Coin)

- Automatic resource management
  - Dealing with all possible failures is hard
  - Closing resources is hard

  Idea: Get the compiler to help

  - Define an interface on resources that knows how to tidy up automatically
    - The new interface is called `AutoCloseable`
    - Relevant Java SE classes have been retro-fitted with this interface
  - Add new syntax to make using this interface convenient
    - The new syntax is referred to as **try-with-resources**

JSR 334 Small language enhancements (Project Coin)

- Automatic resource management

meaning this:

```
public void processFiles() throws IOException {
        InputStream in = null;
        OutputStream out = null;
        try {
            in = new FileInputStream(aFilename);
            out = new FileOutputStream(bFilename);
            …
        } finally {
            if (in != null) {
                try { in.close(); } catch (IOException e) { … }
            }
            if (out != null) {
                try { out.close(); } catch (IOException e) { … }
            }
        }
    }
```

JSR 334 Small language enhancements (Project Coin)

- Automatic resource management

becomes:

```
public void processFiles() throws IOException {
      try (InputStream in = new FileInputStream(aFilename);
           OutputStream out = new FileOutputStream(bFilename)) {
          …
      }
  }
```

JSR 203 More new I/O APIs for the Java platform (NIO.2)

Goal: Enable Java programmers to unlock the more powerful I/O abstractions

- Asynchronous I/O
  - Enable significant control over how I/O operations are handled enabling better scaling
  - Socket & file classes available
  - 2 approaches to completion notification
    - `java.util.concurrent.`**`Future`**
    - `java.nio.channels.`**`CompletionHandler`**
  - Flexible thread pooling strategies, including custom ones

JSR 203 More new I/O APIs for the Java platform (NIO.2)

- New file system API
  - Address long-standing usability issues and boilerplate
    - Modeling of more file system concepts like symlinks
    - File attributes modeled to represent FS-specific attributes (eg owner, permissions, …)
    - `java.nio.file.`**`DirectoryStream`** iterates through directory entries
      - Scales very well, using less resources
      - Allows glob, regex or custom filtering
    - `java.nio.file.`**`Path`** is a new representation of a file location
  - Model entirely artificial file systems with **`FileSystem`** providers
  - File Change Notification
    - Improves performance of apps that currently poll to observe changes
  - All tied together with conveniences in the `java.nio.file.`**`Files`** class

JSR 203 More new I/O APIs for the Java platform (NIO.2)

- Example: Directory visit

```
Files.walkFileTree(myPath, new SimpleFileVisitor<Path>() {
      public FileVisitResult visitFile(Path file,
                                       BasicFileAttributes attrs) {
          try {
              // do some processing of file
              …
          } catch (IOException e) {
              // failed, do error handling
          }
          return FileVisitResult.CONTINUE;
      }
  });
```

JSR 166y Concurrency and collection updates

Goal: Provide Java programmers with more powerful tools to take advantage of the prevalence of multicore

- Major new abstraction: Fork/Join framework
  - Very good at 'divide and conquer' problems
  - Specific model for parallel computation acceleration
  - Implements **`ExecutorService`**
  - Implements work stealing for lopsided work breakdowns
    - Hence more efficient than normal **`Thread`**/**`Executor`** -based solutions
- Other enhancements
  - **`TransferQueue`**: model producer/consumer queues efficiently (similar to **`BlockingQueue`**)
  - **`Phaser`**: very flexible synchronization barrier (similar to **`CyclicBarrier`**)

JSR 292 New invoke dynamic bytecode

Goal: Enable dynamic languages to run more efficiently on the JVM

The JVM is now home to many languages, but it lacks some fundamentals that help make those languages go fast.

- Decouple method lookup from method dispatch
  - Get away from being purely Java language centric
- Approach
  - Add a new bytecode to directly execute a method
  - Allow this to be updated at run time:
    - Change which method is executed
    - Provide a framework for mutators (add/remove parameters, etc)
  - Ensure the JIT-compiler can continue to generate efficient compiled code

Java 7: Smaller features

- ClassLoader changes
  - Enabled parallel classloading capability via new “safe” API
  - URLClassLoader gains a **`close()`** method
- I18N
  - Unicode 6.0, Locale enhancement, Separate user locale and user-interface locale
- TLS 1.2 – Security updates
- JDBC 4.1 – ARM (Auto Resource Manager) awareness
- Client (UI) updates
  - Create new platform APIs for 6u10 graphics features
  - Nimbus look-and-feel for Swing
  - Swing Jlayer component
  - XRender support
- Update the XML stack

Java 7: What's new?

IBM features

- Performance & platform exploitation – z196, POWER 7, …
- Garbage Collector updates
  - General improvements
  - New balanced GC policy
- Technology evaluation of WebSphere Real Time
- Serviceability and consumability improvements

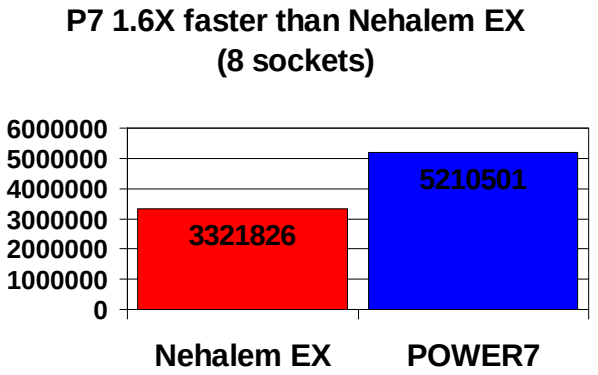
Java 7: Performance & platform exploitation

- "4 out of 5 publishers prefer J9"
  - http://www.spec.org/jbb2005/results/res2010q4
  - 88% SPECjbb2005 publishes in 2010 with J9
    - 94 with J9, 9 with HotSpot, 5 with Jrockit

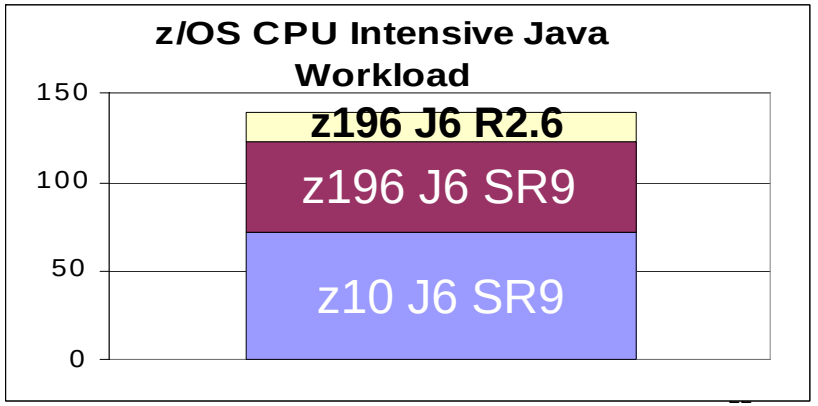
SPECjbb2005

Company	BOPS	JVM	# cores	# chips	Published
Cisco Systems	1017141	IBM J9 VM (build	16	2	Oct-2010
Hewlett-Packard Company	1000188	IBM J9 VM (build	16	2	Dec-2010
IBM Corporation	318556	IBM J9 VM (build	4	1	Oct-2010
IBM Corporation	317811	IBM J9 VM (build	4	1	Oct-2010
SGI	3421167	Oracle Java HotS	64	8	Oct-2010

- POWER7 exploitation
  - New prefetching capabilities
  - Extended divide instructions
  - Conversion between integer and float
  - Bit permutation and popcount instructions
  - BCD assist exploited through Java BigDecimal



- System zEnterprise 196 exploitation
  - 70+ new instructions
    - High-word facility
    - Interlock-update facility
    - Non-destructive operands
    - Conditional load/store
  - **93% aggregate improvement**
    - Some from J9 v26 JVM, majority from hardware



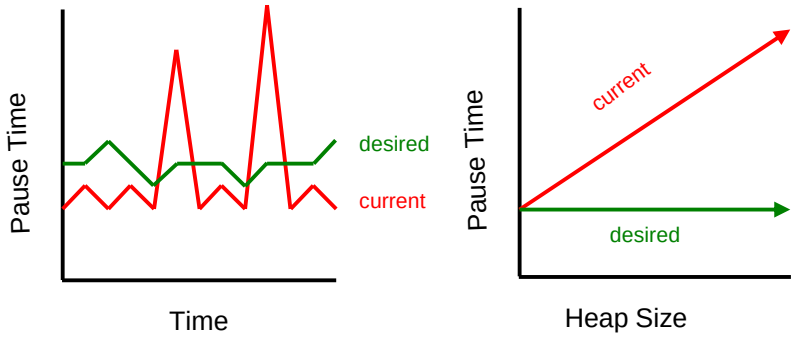
Java 7: General GC improvements

- Default GC policy changed to **gencon**
  - Generational and concurrent collection algorithm provides best out of the box performance for most applications
- Object header size reduction
  - Object headers are 4-16 bytes depending on object type and reference size
  - Reduces GC frequency and provides better object locality
- Scalability improvements across all policies
  - GC pauses reduced on large n-way machines (#CPU > 64)
  - Highly parallel applications benefit from improved allocation scalability
    - As a result **subpool** GC policy now an alias for **optthruput**
- New format verbose:gc logs
  - Event-based instead of summary-based
  - Provides more detailed information for deeper analysis

Java 7: New GC policy added: balanced

Address next generation hardware challenges

- Meet need for scaling to large heaps
  - Existing policies exhibit
    - a mixture of frequent short pauses and long pauses
    - pause times that increase with heap size

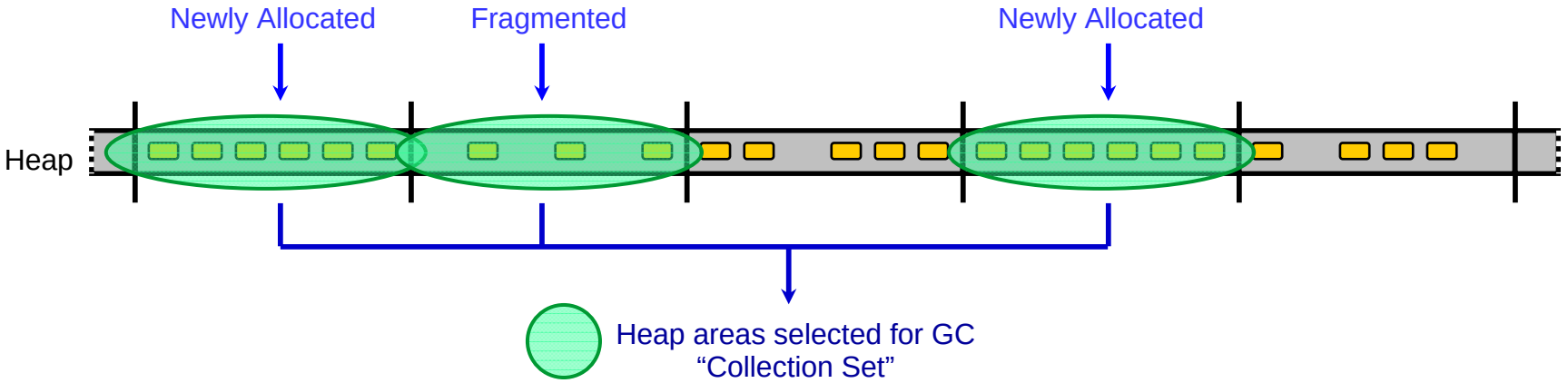


- Provide strong adaptive performance without expert advice
  - Flexibility and adaptive behaviour to provide a good first impression
  - Every tuning option increases complexity by an order of magnitude
  - Showcase hardware capabilities through exploitation of platform facilities

- Maintain and increase competitive edge through innovation
  - Address new developments in industry quickly

Java 7: New GC policy added: balanced

- Incrementally collect areas of the heap that meet our needs
  - Partial Garbage Collect (PGC)
    - Reduced pause times
    - Free up memory
- Heap "collection set" selection based on best ROI
  - Highest potential to free memory
    - Recently allocated objects
    - Areas that would reduce fragmentation
  - Various technologies applied
    - Copy Forward (default)
      - High level of object mobility (similar to gencon)
    - Mark / Sweep / Compact
      - Separates notion of "collection" vs. "compaction"



Java 7: New GC policy added: **balanced**

- Suggested deployment scenarios
  - Large heaps (>4GB in size)
  - Frequent global collections
  - Excessive time spent in global compaction
  - Relatively frequent allocation of large arrays (>1MB in size)

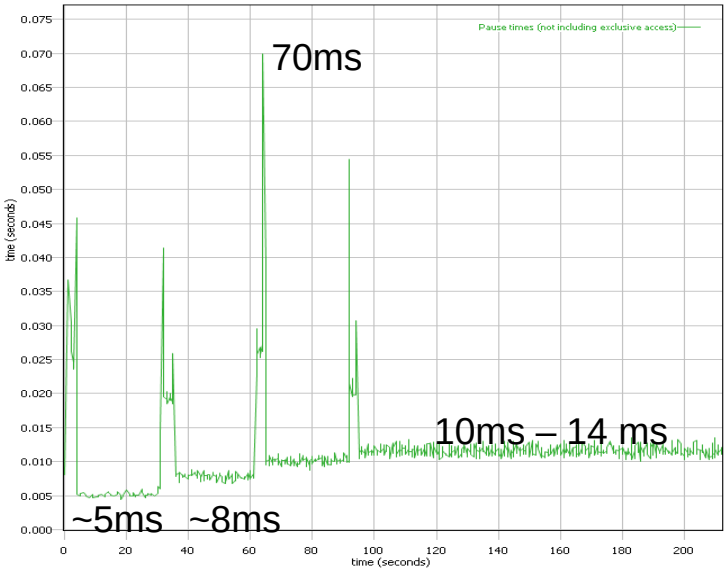
- Fully supported on all IBM JDK 7 64-bit platforms
  - First class citizen with other GC policies

Java 7: Technology evaluation of WebSphere Real Time

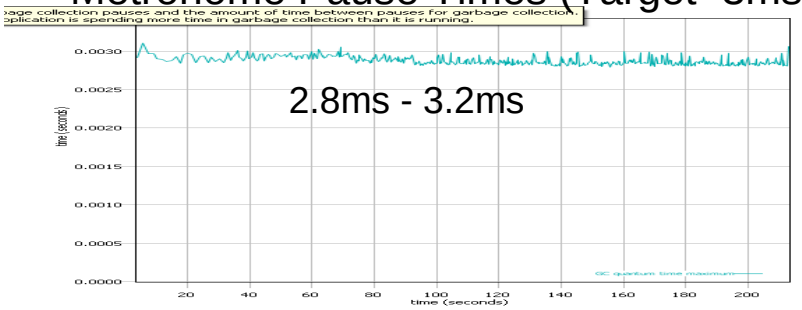
- WebSphere Real Time is a Java Runtime built with J9 technology that provides consistent performance
  - Incremental GC means consistently short (3ms) GC pause times
  - JIT compilations cannot block application threads
  - Also a Hard Real Time flavor that runs on Real-Time Linux® (e.g. RHEL MRG, Novell SLERT)
- IBM Java 7 includes an evaluation version of WRT-V3
  - New pause time target option lets you configure GC pause times
  - Throughput performance improvements
  - 32- and 64-bit Linux on x86, 32- and 64-bit AIX® on POWER®
- Just add **-Xgcpolicy:metronome** to your Java 7 command line to try it out!

Java 7: Technology evaluation of WebSphere Real Time

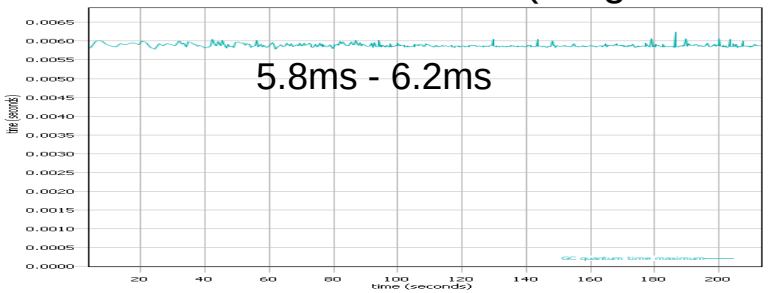
Gencon pause times



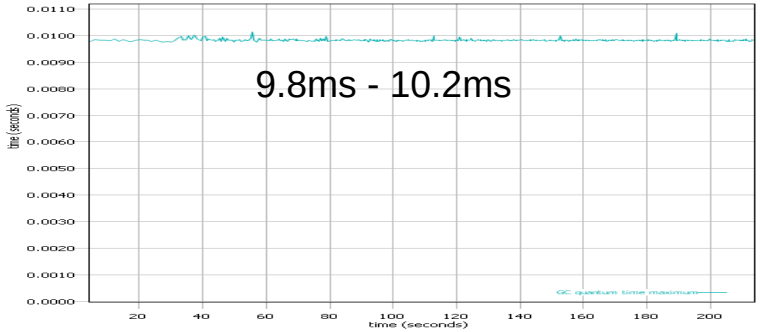
Metronome Pause Times (Target=3ms)



Metronome Pause Times (Target=6ms)



Metronome Pause Times (Target=10ms)



GC pauses: gencon and metronome

- Most GC policies have pause times ranging upwards of 10 – 100 ms
- Metronome controls pause times to as short as 3ms
- Throughput impact, varies by application

Java 7: Serviceability and consumability improvements

- Dump agents (-Xdump)
  - Native stack traces in javacore
  - Environment variables and ULIMITs in javacore
  - Native memory usage counters in javacore and from core dumps via DTFJ
  - Multi-part TDUMPs on z/OS® 64 bit systems
- Trace engine (-Xtrace)
  - Tracepoints can include Java stacks (jstacktrace)
- Message logging (-Xlog)
  - Messages go to the Event log on Windows, syslog on Linux, errlog or syslog on AIX, MVS console on z/OS

Java 7: Serviceability and consumability improvements

Native memory usage counters

```
NATIVEMEMINFO subcomponent dump routine
=======================================

JRE: 555,698,264 bytes / 1208 allocations
|
+--VM: 552,977,664 bytes / 856 allocations
|  |
|  +--Classes: 1,949,664 bytes / 92 allocations
|  |
|  +--Memory Manager (GC): 547,705,848 bytes / 146 allocations
|  |  |
|  |  +--Java Heap: 536,875,008 bytes / 1 allocation
|  |  |
|  |  +--Other: 10,830,840 bytes / 145 allocations
|  |
|  +--Threads: 2,660,804 bytes / 104 allocations
|  |  |
|  |  +--Java Stack: 64,944 bytes / 9 allocations
|  |  |
|  |  +--Native Stack: 2,523,136 bytes / 11 allocations
|  |  |
|  |  +--Other: 72,724 bytes / 84 allocations
|  |
|  +--Trace: 92,464 bytes / 208 allocations
|  |
```

Wait, where did we get to?

A little bit about me

A little bit about you

What I'm going to talk about

Java 7: What is it? What's new? Base features. IBM features.

WAS 8.5: Java 7 in WAS. "Java version switch" feature. Migration toolkit.

Questions

Congratulations, you made it this far! ☺

WAS 8.5: Java 7 in WAS

- **Goals**
  - Enable developers to use Java 7 features immediately without forcing global migrations
  - Decouple Java SE version from WAS version
    - Potentially decrease server migration costs
    - Allow customer flexibility in technology choices
    - Acknowledge future Java versions might be added to existing product versions
  - Standardize Java SE version switching across platforms
    - WAS for z/OS and iOS already supported some mixing
    - Facilitate management of future Java versions
  - Enable specific & targeted migrations

WAS 8.5: Java 7 in WAS

- **Caveat**: does **not** support “bring your own” Java SE implementation
- Suggested implementation pattern:
  1. Install WAS
  2. Install Java 7 optional package
  3. Use 'managesdk' to set all defaults to Java 7
  4. Create a profile
- Alternative pattern options
  - Add a single Java 7 node
  - Add Java 7 to specific servers in a node, migrate only some apps
  - Move profile by profile

WAS 8.5: “Java version switch” feature

- “managesdk” controls what JVMs are used in WAS.
  - Enumerates JVMs available in the product
  - Associates and configures JVMs for profiles, including future defaults for new profiles
  - Found in the app_server_root/bin/ directory

```
..\AppServer\bin> managesdk -listAvailable
CWSDK1003I: Available SDKs :
CWSDK1005I: SDK name: 1.6_64
CWSDK1005I: SDK name: 1.7_64
CWSDK1001I: Successfully performed the requested
  managesdk task.
..\AppServer\bin>
```

WAS 8.5: “Java version switch” feature

- Enabling the Java 7 SDK
  - Set new profile default

```
..\AppServer\bin> managesdk -setnewprofiledefault -sdkname 1.7_64
  CWSDK1022I: New profile creation will now use SDK name 1.7_64.
CWSDK1001I: Successfully performed the requested managesdk task.
```

  - Enable sdk in all profiles

```
…\AppServer\bin> managesdk -enableprofileall -sdkname 1.7_64
CWSDK1001I: Successfully performed the requested managesdk task.
```

- Considerations
  - Between JVMs, ensure the consistency of:
    - Command line options
    - Properties files
    - User-added extension and endorsed jars
    - JNI DLLs
    - Monitoring (eg: ITCAM)

WAS 8.5: Java version compatibility

“Write Once, Run Anywhere” - what does this mean?

- Java commits to **binary** compatibility, not **source** compatibility
  - Existing binaries (ie: class files) continue to work on newer Java versions
  - But: existing Java source may not compile

- Backwards compatible, but not forward compatible
  - Build .java on old, run on new => works
  - Build .java on new, run on old => fails

WAS 8.5: Java version compatibility

Example source compatibility problem

```
public class F {
    public static int enum = 5;
}

> javac -source 1.4 -target 1.4 F.java
F.java:2: warning: as of release 5, 'enum' is a keyword, and
may not be used as an identifier
1 warning

> javac -source 1.5 -target 1.5 F.java
F.java:2: as of release 5, 'enum' is a keyword, and may not be
used as an identifier
(use -source 1.4 or lower to use 'enum' as an identifier)
1 error
```

WAS 8.5: Java version compatibility

- What kinds of change should we watch for?
  - Language syntax
  - API enhancements
  - Unspecified behaviour changes
  - Unspecified class changes
  - Bugs (fixed and introduced)
- Ensure 3rd party libraries work on Java 7
  - Almost all will due to backwards compatibility, but a small minority may have some very specific issues. (e.g.: version checking)
  - Most bytecode instrumentation libraries need new versions
    - ASM -> version 4.0
    - BCEL -> TBD – under development

WAS 8.5: Caveats, Tricks and Tips

- To use a Java 7 runtime, you do NOT need to rebuild your code.
- How to remain Java 6 compatible?
  - Warning: "`-source 1.7 –target 1.6`" does NOT work.
  - Can use "`-source 1.6 –target 1.6`" but beware use of new classes / methods.
  - Optional practice: compile against both JDK 6 & JDK 7 to be sure.
- Old trick of "build at new level, target old" does NOT work. ARM in particular requires significant JVM plumbing to support.

WAS 8.5: Migration toolkit

- Migrate WebSphere applications from older releases to WebSphere Application Server v7.0, v8.0 or v8.5
  - Migrate applications from v5.1, v6.0, v6.1, v7.0 and v8.0
- Migrate From Oracle (WebLogic & OracleAS), JBoss or Tomcat to WebSphere Faster/Easier
  - Migrate applications 2x as fast
  - Migrate web services 3x as fast
- The tool programmatically scans applications and identifies the changes required.
  - In many cases the tool is capable of making the application change itself, in other cases it provides guidance on how to make the required change.
  - Generate reports to assess the migration task.
- With this Free toolkit, you ease the migration process and speed time to value

Summary

- Java 7 base features
  - Base features
    - JSR 334 – Small language enhancements (Project Coin)
    - JSR 203 – More new I/O APIs for the Java platform (NIO.2)
    - JSR 292 – New invokedynamic bytecode
    - JSR 166y – Concurrency and collections updates
    - Some smaller features (TLS 1.2, UNICODE 6.0…)
  - IBM JDK features
    - Performance & platform exploitation – z196, POWER 7, …
    - Garbage Collector updates
    - Technology evaluation of WebSphere Real Time
    - Serviceability and consumability improvements
- WAS 8.5
  - Java 7 in WAS
  - “Java version switch” feature
  - Java version compatibility
  - Caveats, tips and tricks
  - Migration toolkit

IBM Knowledge Center

- The new home for IBM product documentation
  - A one-stop shop for IBM technical publications; replaces Information Centers
  - Create collections that apply to you
  - Collaborate with IBM and your colleagues; share comments and rate pages
- Want to learn more about IBM Knowledge Center?
  - Come and visit the stand
  - Look for representatives wearing these badges

- IBM Knowledge Center collections that might interest you:
  - IBM SDK, Java Technology Edition — http://ibm.biz/javasdkdocs
  - IBM Monitoring and Diagnostic Tools for Java — http://ibm.biz/javasdktoolsdocs
  - IBM WebSphere Real Time — http://ibm.biz/wrtdocs

Questions?

References

- Java 7
  - Project Coin
    - https://www.ibm.com/developerworks/mydeveloperworks/blogs/javaee/entry/5_minute_guide_to_project_coin9?lang=en
  - NIO.2
    - http://www.ibm.com/developerworks/java/library/j-nio2-1/index.html
  - Fork/Join
    - http://www.ibm.com/developerworks/library/j-jtp11137/index.html

- WAS 8.5
  - What's new
    - http://www.ibm.com/developerworks/websphere/techjournal/1206_alcott/1206_alcott.html
  - Using Java 7 in WAS 8.5
    - http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.installation.base.doc/ae/tins_installation_jdk7.html
    - http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.nd.multiplatform.doc/ae/rxml_managesdk.html
  - Migration Tools
    - http://www.ibm.com/developerworks/websphere/downloads/migtoolkit/index.html

Copyrights and Trademarks