



IBM Real Time for Deterministic Workloads
Benjamin Vera-Tudela
Mary Ann Fisher
IBM Corporation

IBM Real Time for Deterministic Workloads

- Background on Real Time
 - What is it? Value Proposition...
- What's New with Real Time Linux and Java?
 - Open and COTS Approach
 - Real Time Linux
 - Real Time Java
- Current Deployment
- What's Next
 - Responsive Technology
 - Real World Aware Systems with Real Time Response

What Is Real Time?

*Real-time computing – Computation that fails if it does not complete within a specified deadline

- A **“hard” real-time** application has tasks which have hard, firm deadlines for completion of their computation, typically on the order of a few microseconds. Missed deadlines cause system failure (at best) or life-threatening catastrophe (at worst).
- Examples:
 - Federal mission critical and Financial trading
 - Industrial automation (robot picking up an object from a moving belt)
 - Aircraft flight control
 - Medical devices/imaging
- A **“soft” real-time** application can tolerate some missed deadlines and does not typically require hard scheduling guarantees from the OS.
- Examples:
 - Point of sale system
 - Consumer devices (PDA, phones, etc...)
 - Massive multiplayer game servers
 - Media display/playback

*

Why Is Real Time Important?

*Predictable Response Times

Federal:

- *Air Traffic Control*
- *Defense Weapon Control*

Financial Services:

- *Trading and Analytics*

Energy & Utilities:

- *Grid Fault Tolerance*

Industrial

- *Plant Automation, Quality Control*

Telecommunications

- *Text Messaging, Streaming*



1729R	U.S.	FEBRUARY	INDU
1728RH	#DOW	JONES	INDUSTR
2487DH	#DJIA	TOPS	10000 P
INDU	+42.18	VOLU	77,275
INDP	10000.95	UVOL	48,904
UTIL	+.60	DVOL	20,289
TRAN	-7.91	TRIN	.49

Problem Statement

*Specialized Hardware

- Black box, not flexible

*Specialized Real Time Operating Systems (RTOS)

- Closed source and/or proprietary
- Linux® not widely used for real-time systems due to
 - *Unpredictable scheduling*
 - *Low timer resolution (10 ms granularity)*
 - *Non-preemptible kernel*

*Applications written in C, C++, ADA

- Required specialized skills, and applications were not reusable or portable
- Java™ not widely used for real-time systems due to
 - *Regular Java™ Threads*
 - *Garbage Collection*
 - *Class Loading*
 - *Just-in-time (JIT) Compiling*

What Has Changed?

*Real Time Solution that is Open and Flexible:

- x86 Architecture and Linux®
 - *Control over architecture*
 - *Avoid vendor lock-in*
 - *Facilitate interoperability across heterogeneous systems*
 - *Real Time Linux® expedites offerings in marketplace*

- Java™ Programming Language
 - *Access to broader skill set than C, C++, ADA*
 - *Increased programmer productivity*
 - *Expedited Real Time application development and support*
 - *Expedited innovation through reuse and collaboration*
 - *Massive community of ISVs*

The Power of Linux[®] and Java[™] Combined

*WebSphere[®] Real-Time (WRT)

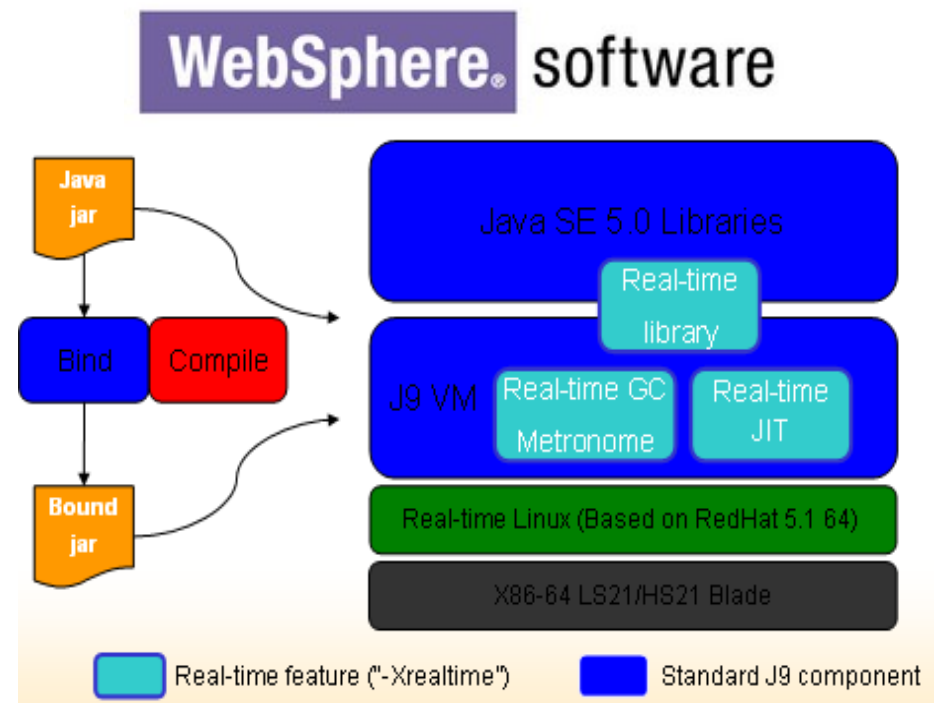
- Java[™] 2 Standard Edition
- Real-Time Specification for Java[™] (RTSJ – JSR 1)
- Metronome Garbage Collector (GC)
- Compilation Strategies for Real-Time (AOT, JIT)

*Real-Time Linux[®]

- Enhancements to optimize real-time workloads
- High resolution time and timers
- Fully pre-emptible kernel
- Threaded interrupt handlers
- Priority inheritance and fast user-space mutexes
- Symmetric Multiprocessing (SMP) RT scheduling

*Select IBM System x Hardware

- Enhancements to optimize real-time workloads



Select IBM System x™

- * Currently LS21 and HS21 XM Blades have been optimized to support real-time w
 - System Management Interrupts (SMIs) are used to perform a variety of tasks at the CPU level
 - Report fatal and non-fatal hardware errors
 - Perform power management (thermal throttling, power capping)
 - SMIs introduce latency which are hard to detect
 - SMI remediation:
 - Moves non-fatal event handling to the OS
 - BIOS handles only fatal events
 - BMC no longer requests to throttle the CPU
 - New OS service “ibm-prtm” manages entering and exiting SMI-free state

Real Time Linux® Extensions

- * High resolution time and timers
 - High-resolution timers provide single nanosecond granularity
 - Timers can now expire within a few microseconds of each other
- * Fully pre-emptible kernel
 - Kernel spin-locks replaced with mutexes for higher levels of pre-emption
- * Threaded interrupt handlers
 - Allows real-time processes to be prioritized against interrupt handler processes
- * Priority inheritance and fast user-space mutexes
 - Kernel avoids priority inversion by raising priority of locking process to that of the highest priority process
 - Fast-user space mutexes (futexes) help reduce regular mutex overhead
- * SMP real-time scheduling
 - Allows real-time processes to be scheduled against all processor run queues

Metronome Garbage Collection

- *Unique technology from IBM T.J. Watson

- Garbage collection is scheduled as just another task on task
- Provides bounded pause times as small as 1

- *Enables the use of off-the-shelf Java™ compilers

- No need for specialized allocation schemes or
- Greatly simplifies real-time application development
- Enables complex real-time applications through
-

Figure 1. Traditional GC pauses



Figure 2. Short pause times but little application time

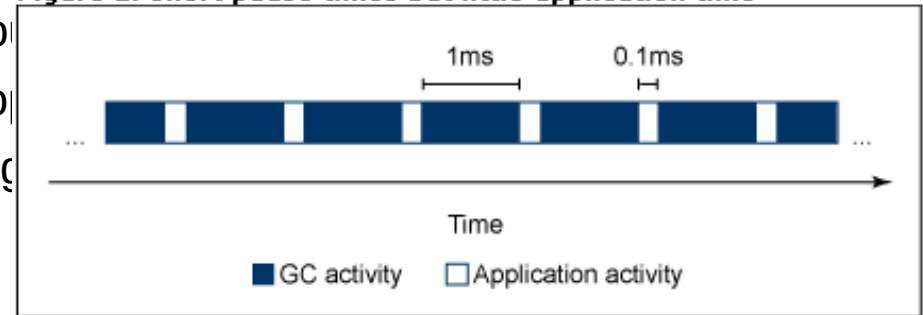
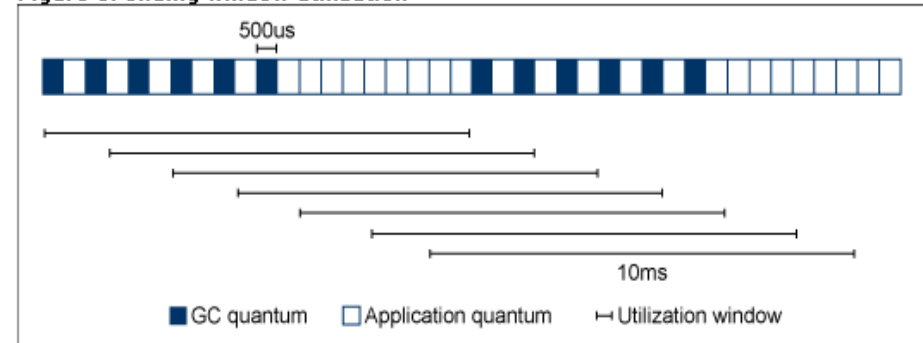


Figure 3. Sliding window utilization



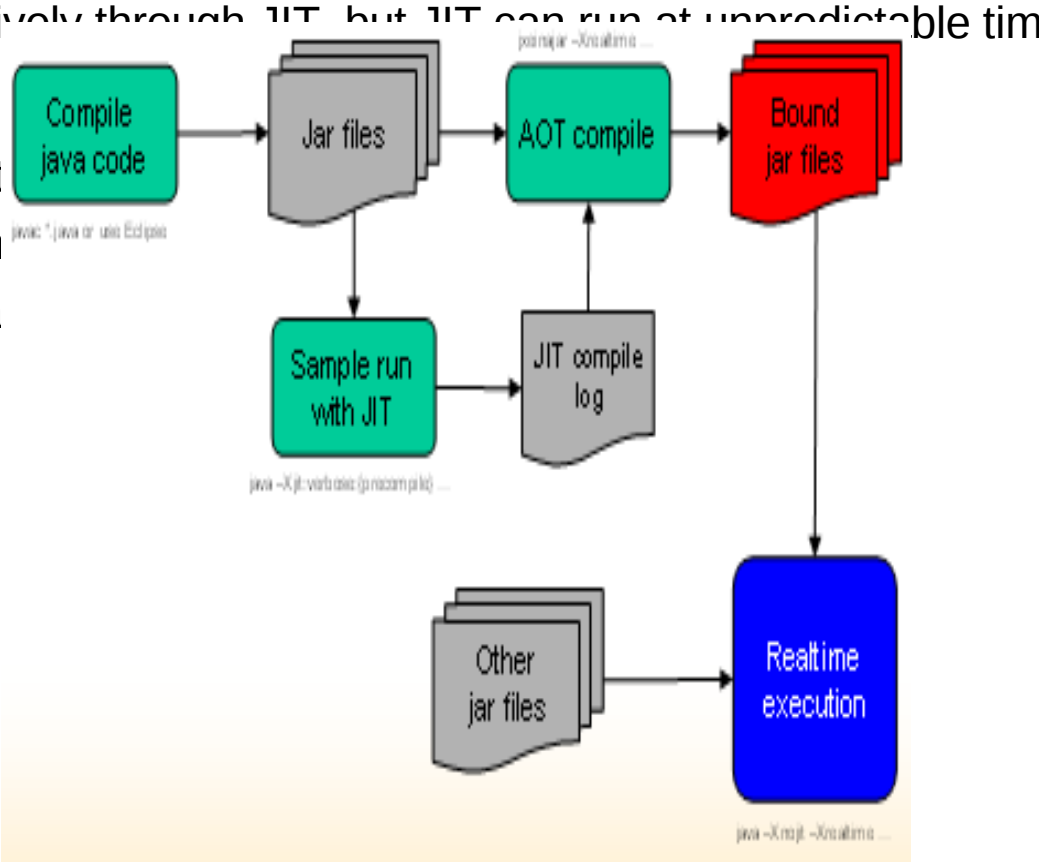
Compilation Strategies for Real Time

- *JVM Compilation is dynamic by default

- Code may be interpreted or compiled selectively through JIT, but JIT can run at unpredictable times

- *For real-time, there are multiple choices

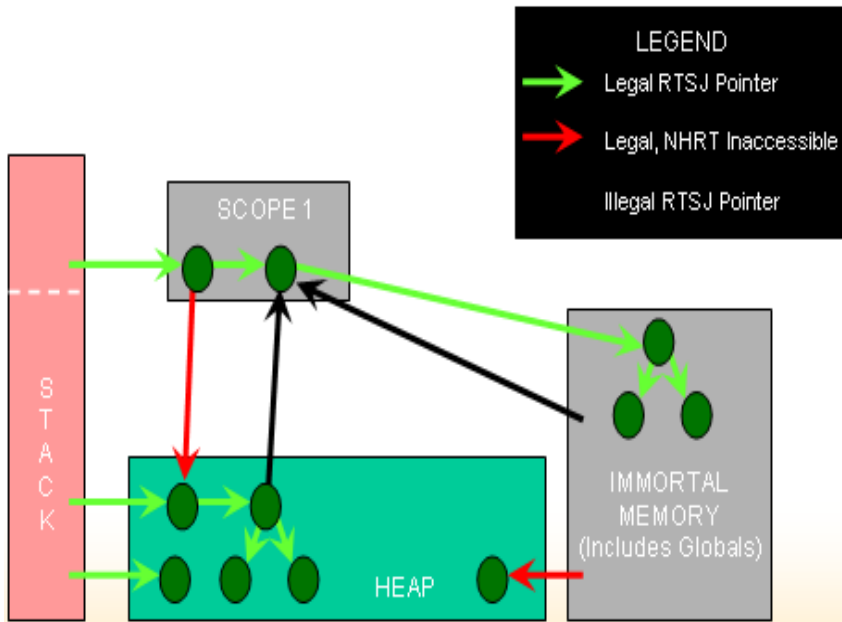
- Ahead-of-time (AOT) compilation (much better performance)
- User-controlled JIT (better than AOT performance)
- JIT-at-low-priority (best achievable performance)
-



Real Time Specification for Java™ (RTSJ)

*Thread scheduling

- “RealtimeThread” allows specification of scheduler
- Fixed priority scheduling and additional priorities



*Memory Management

- Partitioned, non-garbage collected memory space
- Allows special threads to run at higher priority than standard threads
- Lower latency can be achieved using standard RTSJ

Evaluation Program

Getting Started with Real Time Now

- *IBM WebSphere® Real Time**
- *Moon Lander Demo**
- *Sample Source Code**
- *Technical Documentation**
- *Phone and Email Support ***
- *Hardware and Real Time Linux® ***

-
-
-
-
-

(*) Loaner hardware is subject to qualification and availability. Phone and email support will be conducted on an availability basis and at no charge.

Some Scenarios

*Military

- In today's computer-intensive military, the ability to react rapidly

*Financial Services

- A trading desk at a brokerage firm cannot ensure the integrity



Linux and Open Source Matter to the Navy

Challenge

✓ Build a Real Time Linux Operating System that would compliment the RT Java to meet the performance demands of the DDG-1000 program while working with the Linux Community to mainline the enhancements.



DDG 1000
Zumwalt Class

Solution

- ✓ Real Time Linux – Led by the IBM Linux Technology Center and built on the work of Red Hat and Open Source Community
- ✓ IBM System x and BladeCenter based solution
- ✓ Fully preemptive kernel, reducing critical path latencies
- ✓ Priority inheritance enabled kernel and userspace locking

Key Benefits

- ✓ Open Source Solution that is on track to be adopted by the Linux Mainline
- ✓ Context switch latency under 25 μ s (99.9999% under 20 μ s)
- ✓ Open Real Time Stack: RT Linux- RTSJ & RT GC RT Java- x86 Blades
- ✓ Reduced Risk and Reduced Total Cost of Ownership

© 2007 IBM Corporation



IBM's Role in DDG 1000 TSCEi Program

- **IBM was awarded a sub contract to provide the Total Ship Computing Environment Infrastructure**
- **An end to end RT Enterprise Infrastructure solution based on open standards and leading technology**
 - Platform for Strategic COTs 3rd Party Open Architecture – based Integrations
- **A 'Blade' computing infrastructure – a dense, integrated, modular, scalable, open architecture footprint**
- **IBM's RT Java with deterministic Garbage Collection, and Open Source RT Linux from Redhat**



DDG 1000 – A Collaborative Partnership

- IBM and Redhat raised the bar to meet the demands of Raytheon and the US Navy
 - Developed a Real Time Linux in collaboration with the Linux Open Source Community providing all patches to the public domain
 - IBM Real-Time Java and Garbage Collection (Metronome) brought forward from IBM Research to solve a serious problem and cut development costs substantially for the program
 - Collaborative relationships between Raytheon Engineers and IBM Development / R&D Engineers to breakthrough barriers
 - IBM development schedules aligned to Raytheon's DDG 1000 program milestones



IBM's Value Proposition to TSCE

- **Greater ease of Tech refresh**
- **Highly Available system**
- **Flexible, dynamically reconfigurable footprint**
- **Simplified maintainability, ease of use, reduction in manpower to support**
- **Built on Open Standards and aligned to Navy Open Architecture Initiative**
- **Dramatically reduced application life cycle costs (development and support)**
- **Demonstrated technology roadmap, adhering to open architecture design principles**
- **IBM Research partnership - Innovation infusion**
-
-



Core Solution Components for DD(X) Response Today's Technology Which Will Lead to Tomorrow's Delivery

BladeCenter will be the core processors of the DD(X) Solution. With the capability to integrate switching and the ability to run Windows and either Linux or vendor POSIX OS, the IBM BladeCenter offers a dense, high performance, highly reliable solution.

POWER(X) based processors will provide an SMP solution for the Large datacenter environments. In today's technology, the P5-570 provides a modular scalable solution with excellent density.



IBM Director will provide a robust and secure hardware management platform to manage BladeServers and possibly pSeries servers at time of delivery



Tivoli Storage Manager (TSM) will perform hierarchical storage management of data throughout the small, medium and large environments.

The DS6000 offers the requisite density TODAY with 4.5 TB (raw) in 3U of space. The DS6000 can scale up to 224 Disk drives or 67 TB today in 42U of space. This dense solution saves power and space, giving rack space back.

IBM LTO-3 Capable Libraries will Provide Scaleable Tape Solutions to all Data System Configurations

IBM Hardware Solution Components Are ...

- Integrated
- Modular
- Dense
- Power Efficient
- High Performance
- Highly Available
- Manageable

IBM's TSCEi Building Block Approach to meet small, medium and large configurations

Two Chips per Blade (Quad Core in 2008)



- DS4700- 4.8 TB in 3U space
- Scales up to 33.6 TB in 42U



- Blade Center
- Up to 14 blades per chassis
- Opteron or PowerPC
- RT Linux



- TS3100 Tape Library
- LTO-3



BladeCenter Chassis

- Gigabit Ethernet Switches
 - **Portfolio of switches (IBM,Cisco,Nortel)**
 - **Lower cost via Integration**
 - **Functions range from Layer 2 thru Layer 7**
- Fibre Channel Switches (2Gb FC Fabric)
 - **Portfolio of Switches (IBM, Brocade)**
 - **Potentially lower cost via integration**
 - **Full support of FC-SW-2 standards**
- Power Subsystem
 - **Upgradeable as required**
 - **Redundant and load balancing for high availability**
- Calibrated, Vectored Cooling™
 - **Highly fault tolerant**
 - **Allow maximum processor speeds**
- BladeCenter Management Modules
 - **Full remote video redirection**
 - **Out-of-band / lights out systems management**
 - **Concurrent Serial connectivity**
 -



**KVM Switch /
Management Module**

IBM's interest in real-time

- Classical real-time systems are getting more complex
 - Military, telecom, industrial, automotive, gaming
- Real-time systems becoming part of enterprise IT
 - Sensor networks, Event processing
- Commercial systems have unpredictable performance
 - Service Level Agreement failures when overloaded
- A need for a new way to build real-time systems
 - Engineered for predictability and reliability
 - Using the latest programming tools and techniques

IBM's investment in Java

- Java is building block for hundreds of IBM applications
 - Provides a consistent 'operating system'
 - Safe, efficient language to develop code in
 - Consistent, high quality tooling for all dev phases
- Significant Performance work
 - Heavy investment in JVM, GC, JIT, Class Lib
 - Hardware ranges from MIPS,ARM,SH4 to x/p/zSeries
 - JVM designed for easy target to new OS and HW

What's Next?

- Responsive Systems

- Systems that are real-time, event-based or time dependent



- Real World Aware Systems with Real Time Response

- Solutions that integrate business information processing with sensor and actuator manipulations of the external world



Trends driving Responsive Systems

- Business Information processing systems are expected to meet stringent latency requirements in handling requests for service
-
- Integration of embedded Systems into Command and Control
-
- Improved Message oriented middleware facilitate growth in event processing applications
-
- Service oriented and component based techniques reduce cost of developing and deploying responsive systems

Summary:

- Real Time for Deterministic Workloads is:
 - COTS
 - Open
 - Linux
 - Java
- Infrastructure is expanding to support
 - Responsive Systems
 - Real World Aware Solutions
- Service Oriented Architecture with Real Time:
 - Make responsive solution development affordable

Resources

- **For sales opportunity questions and assistance please contact:**

Bruce Bogart, Linux WW Sales Lead – bbogart@us.ibm.com

- **For access to WebSphere® Real Time, please visit:**

<https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=swg-wsrt>

- **Technical documentation is available through developerWorks:**

Part 1: Using the Java™ language for real-time systems

Part 2: Comparing compilation techniques

Part 3: Threading and synchronization

Part 4: Real-time garbage collection

Part 5: Writing and deploying real-time Java™ applications

Part 6: Simplifying real-time Java™ development

Sample applications including source code

URL: http://www.ibm.com/developerworks/views/java/libraryview.jsp?search_by=Real+time+Java+Part

Questions?



Trademark and Registered Trademark Attributions

- Linux is a registered trademark of Linus Torvalds in the United States,
- Java and all Java-based trademarks are trademarks of Sun Microsystems
- Other company, product, or service names may be trademarks or services