Smart Work for a Smarter Planet

Real-Time Java

- New Real-Time Platform -

Speaker Name : 노 주환 Ph.D

E-Mail : jason@namooinc.com

IMPACT Korea 2009

Agenda

- Real-Time 정의
- Real-Time 시스템의 현 주소
- The Real-Time Specification for Java (RTSJ)의 등장
- RTSJ의 주요 기술적 특성
- RTSJ Benchmark 결과 와 주요 Reference
- ㈜ 나무의 RTSJ기반 CoreCode Framework 소개

What does Real-Time mean?

REAL-TIME ≠ “REAL FAST”

Real-Time == Deadline

== Determinism

== Predictability

Who need Real-Time Systems?

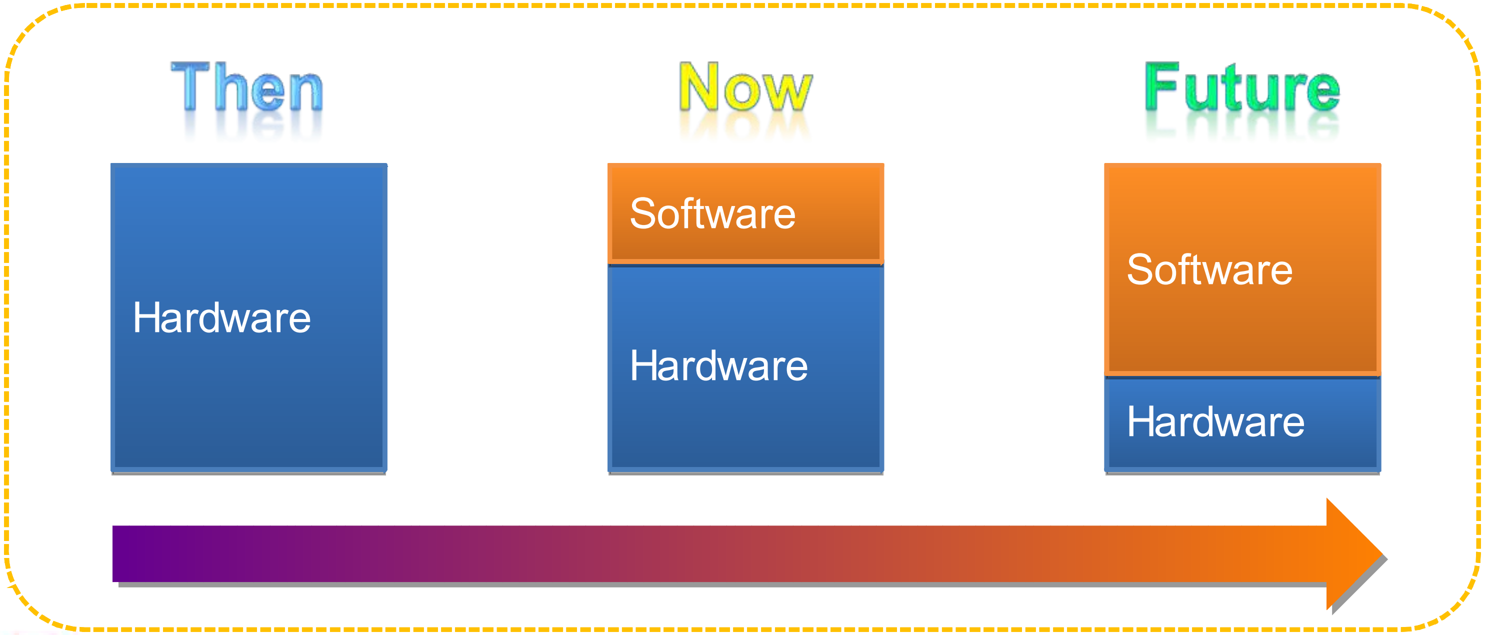
A better question would be… who doesn‘t.

- Improved predictability
  - Safety Critical: Want to stop when you hit the brake?
  - Process Control: The caster must stop when you hit the button on the HMI*
  - Web Servers: Click 'Reload' – it's taking too long.
- Military - Accurate missile tracking
- Telecommunication Infrastructure
  - VoIP, PBX, IMS, new 3G services
  - Predictable call connection; avoid irritating the user
- Banks - Responsive trading

*Human Machine Interface

Real-Time Systems: Where are we?

- Classical real-time systems are getting more complex
  - Complex real-time code in devices
  - Military, telecom, financial, industrial, automotive
- Real-time systems becoming part of enterprise IT
  - Merger of networking and devices:
    U-City, U-Building, Sensor Networks etc.



Real-Time Systems: Where are we?

- Most of the real-time/embedded systems are
  - developed in C / C++
  - C/C++ is more productive than assembly code
  - NOT the most productive, error-free languages

- Increasingly difficult to find C/C++ programmers or to retain them

- Starting to struggle with the maintenance costs of C/C++ applications

Real-Time Systems: Where are we?

Increasing the need for a common, high-level, fully supported, correct, advanced (Java-based) real-time application development platform.

Java… comes to the rescue

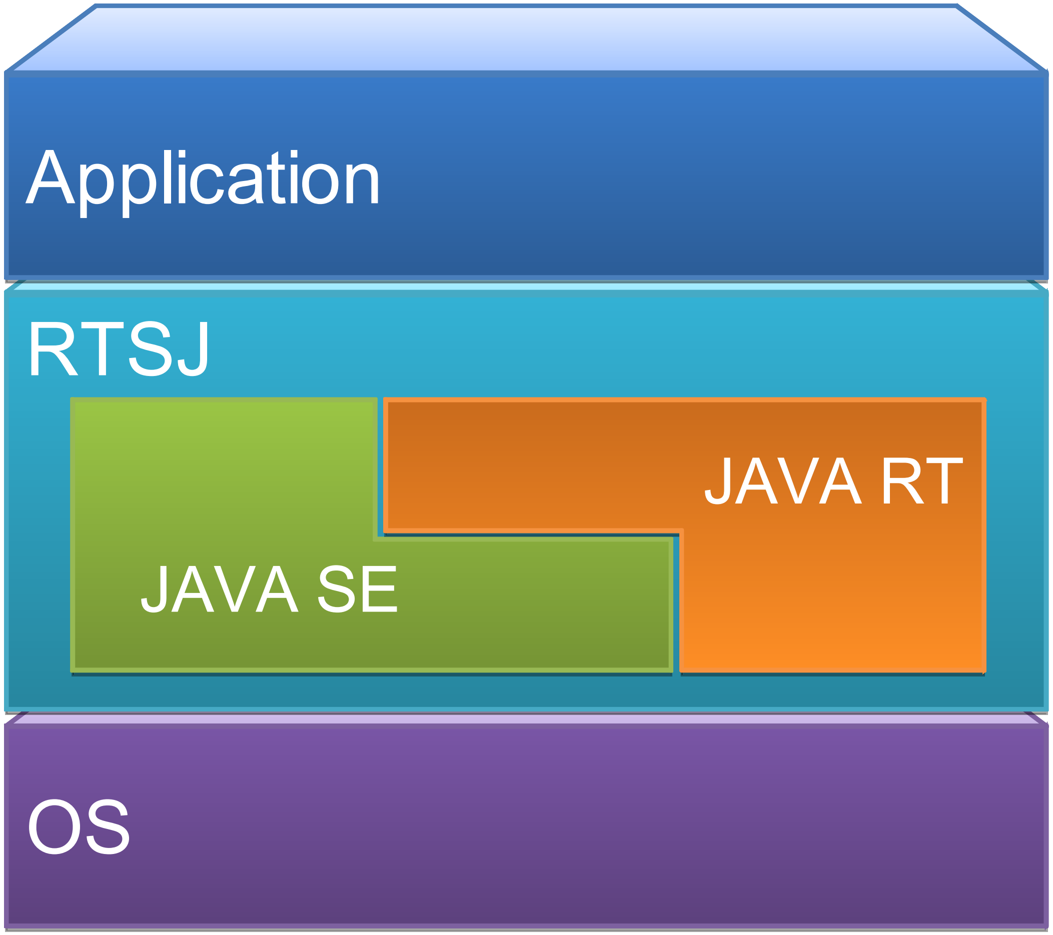
Java… Stochastic

- Java is unsuitable for developing real-time systems
  - Java is Slow (Hmmm…)
  - Non-deterministic GC (Stop the World!)
  - JIT Compilation – Dynamic class (un)loading
  - Inconsistent Memory Allocations

Java… Stochastic

- Java Language Shortcomings
  - Java thread scheduling is purposely under-specified (to allow easy implementation of JVM on as many platform)
  - The GC can preempt Java Threads
  - Java provides coarse-grained control over memory allocation, and it does not provide access to raw memory
  - Java does not provide high resolution time, nor access to signals, e.g. POSIX Signals

Real-Time Specification for Java (RTSJ)



RTSJ Chronology

1998

Real-Time Specification for Java (JSR-001) proposal submitted

Many companies represented : IBM, Sun, Ajile, Apogee, Motorola, Nortel, QNX, Thales, TimeSys, WindRiver

2002

JSR-001 approved by the Java Community Process

TimeSys Reference Implementation

2005

RTSJ update proposal submitted (JSR-282)

Several JSR-1 compliant products: IBM, Sun, Apogee

2007

RTGC added to JVMs

JSR-1 APIs added to RTGC enhanced JVMs

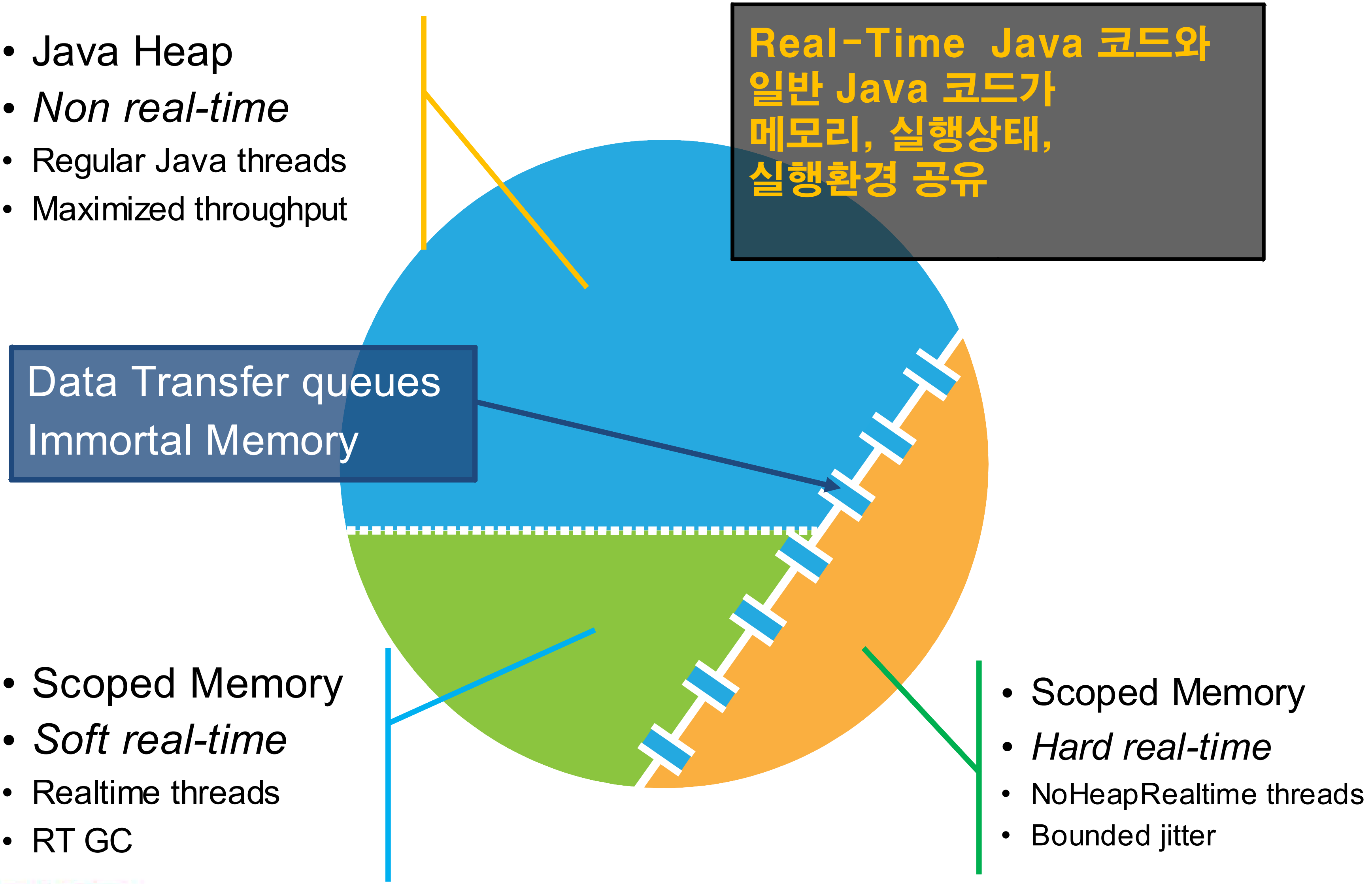
2008

NEW IBM/SUN JSR

RTSJ – Key Features

- Thread Scheduling & Dispatching
  - Priority-preemptive scheduling
- Enhanced Synchronization
  - Priority inversion avoidance
- New Memory Management
  - Allocation contexts without garbage collection
- Added Asynchronous Event Processing
  - Internal events, external “happenings”, and handlers
- Time, Clocks and Timers

RTSJ System Model



- Java Heap
- *Non real-time*
- Regular Java threads
- Maximized throughput

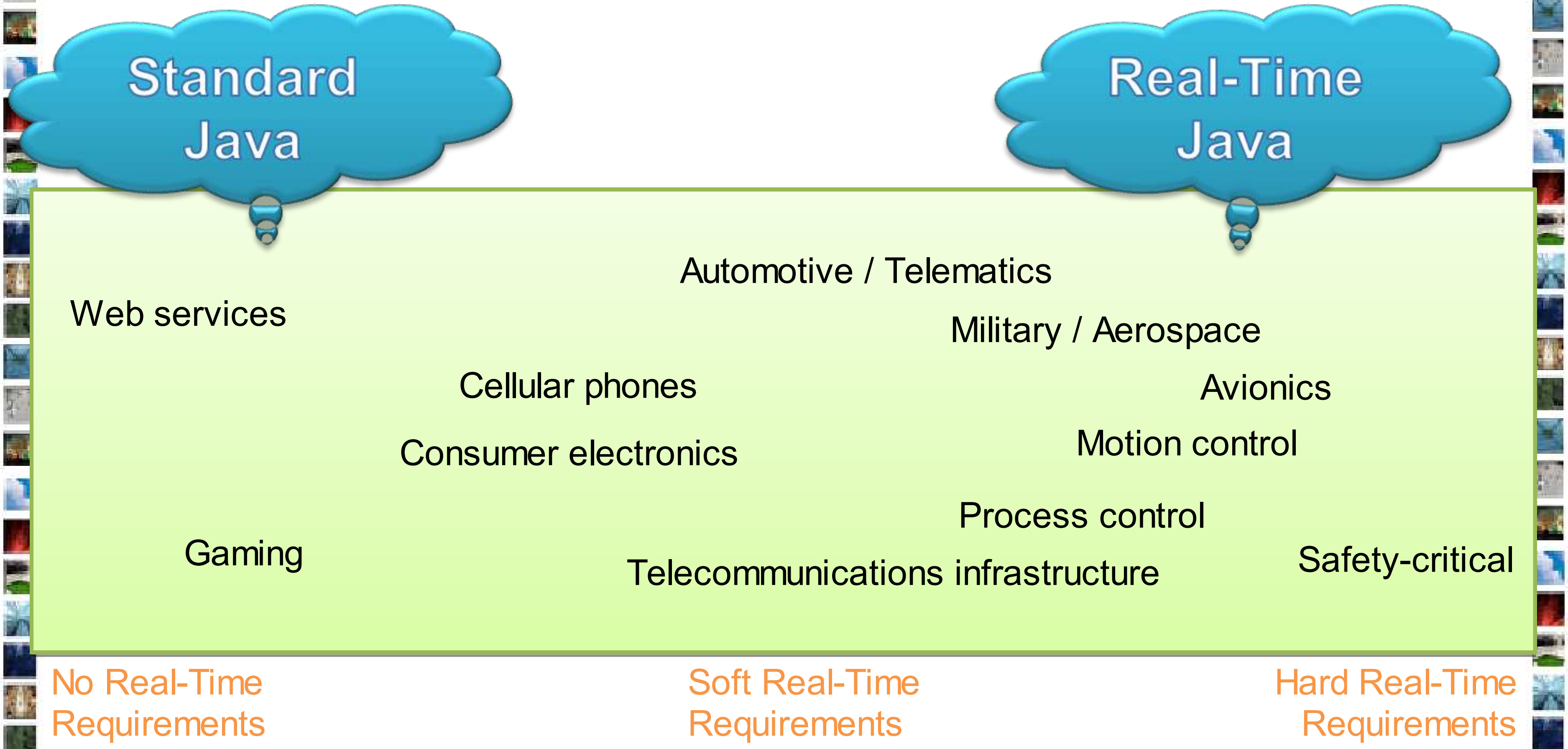
Real-Time Java 코드와 일반 Java 코드가 메모리, 실행상태, 실행환경 공유

Data Transfer queues
Immortal Memory

- Scoped Memory
- *Soft real-time*
- Realtime threads
- RT GC

- Scoped Memory
- *Hard real-time*
- NoHeapRealtime threads
- Bounded jitter

Already widely Deployed



Standard Java

Real-Time Java

Automotive / Telematics

Web services

Military / Aerospace

Cellular phones

Avionics

Motion control

Consumer electronics

Process control

Gaming

Safety-critical

Telecommunications infrastructure

No Real-Time Requirements

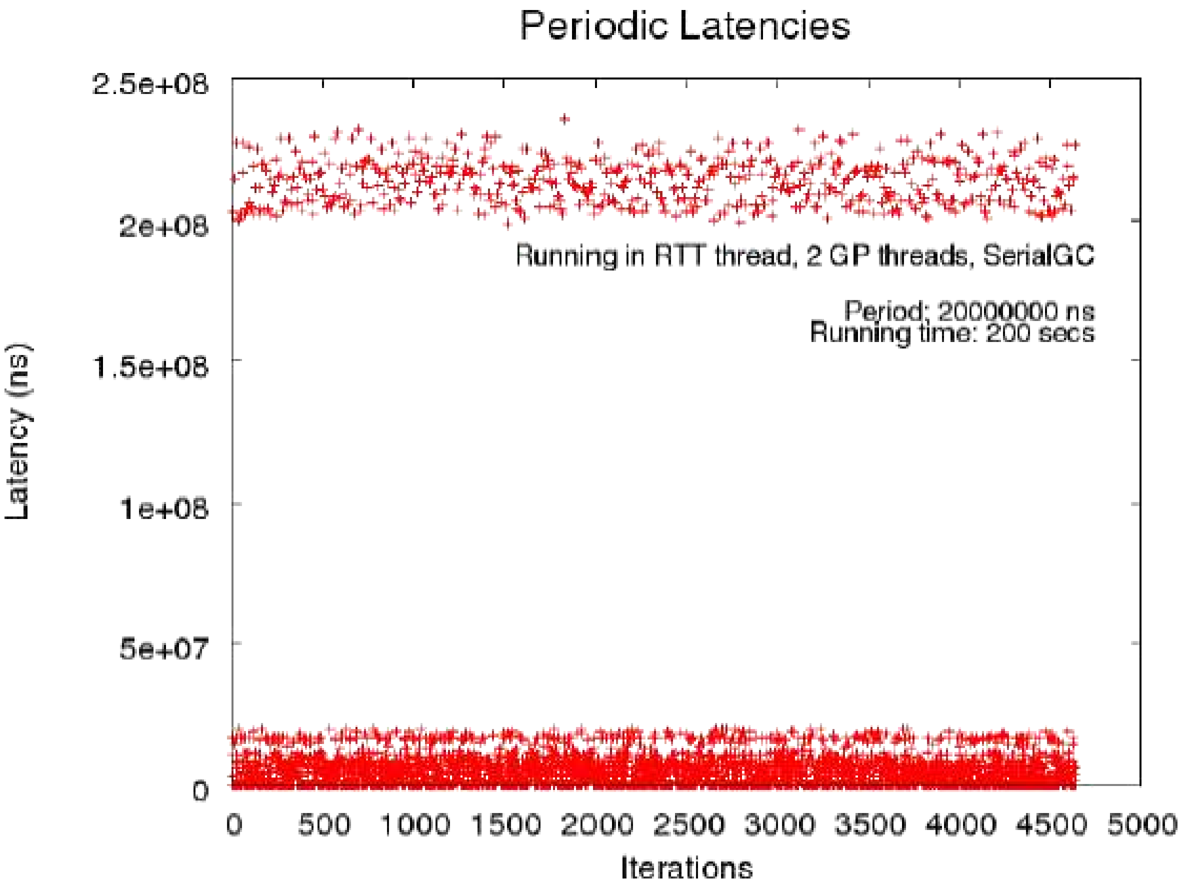
Soft Real-Time Requirements

Hard Real-Time Requirements

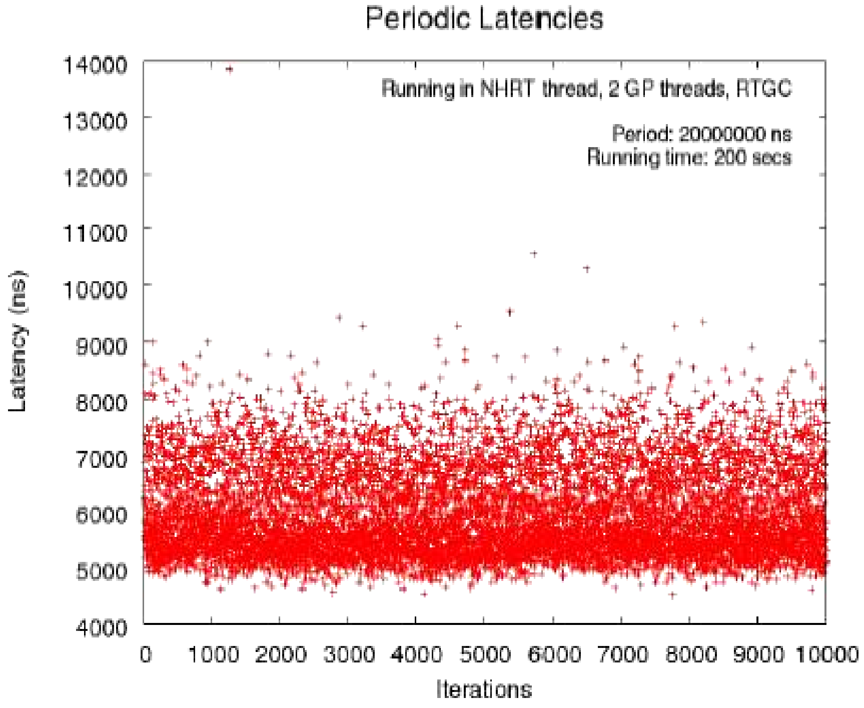
Various application domains are already exploiting Real-Time Java !

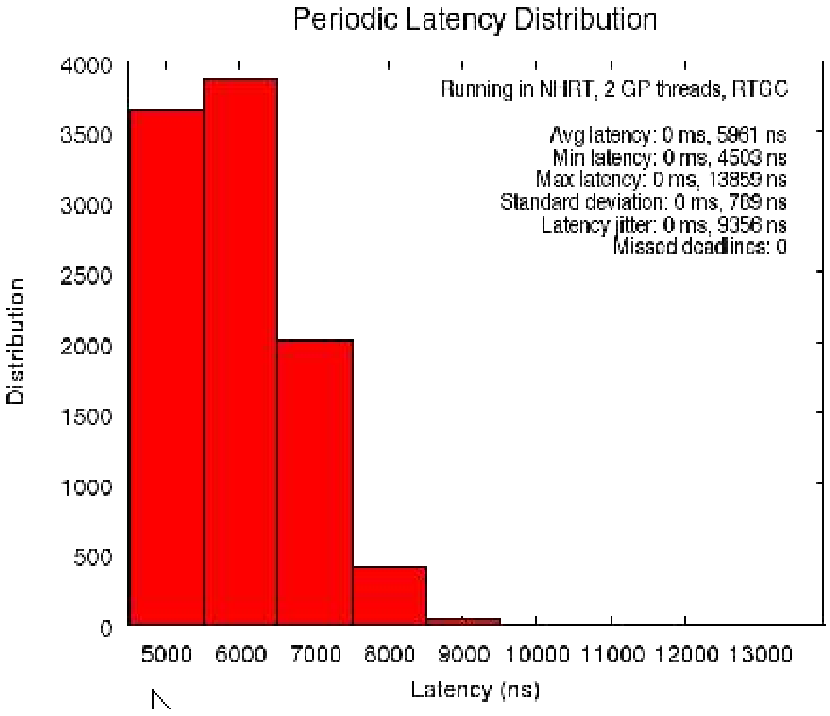
So, how deterministic is the RTSJ?

RTSJ Performances - Response time

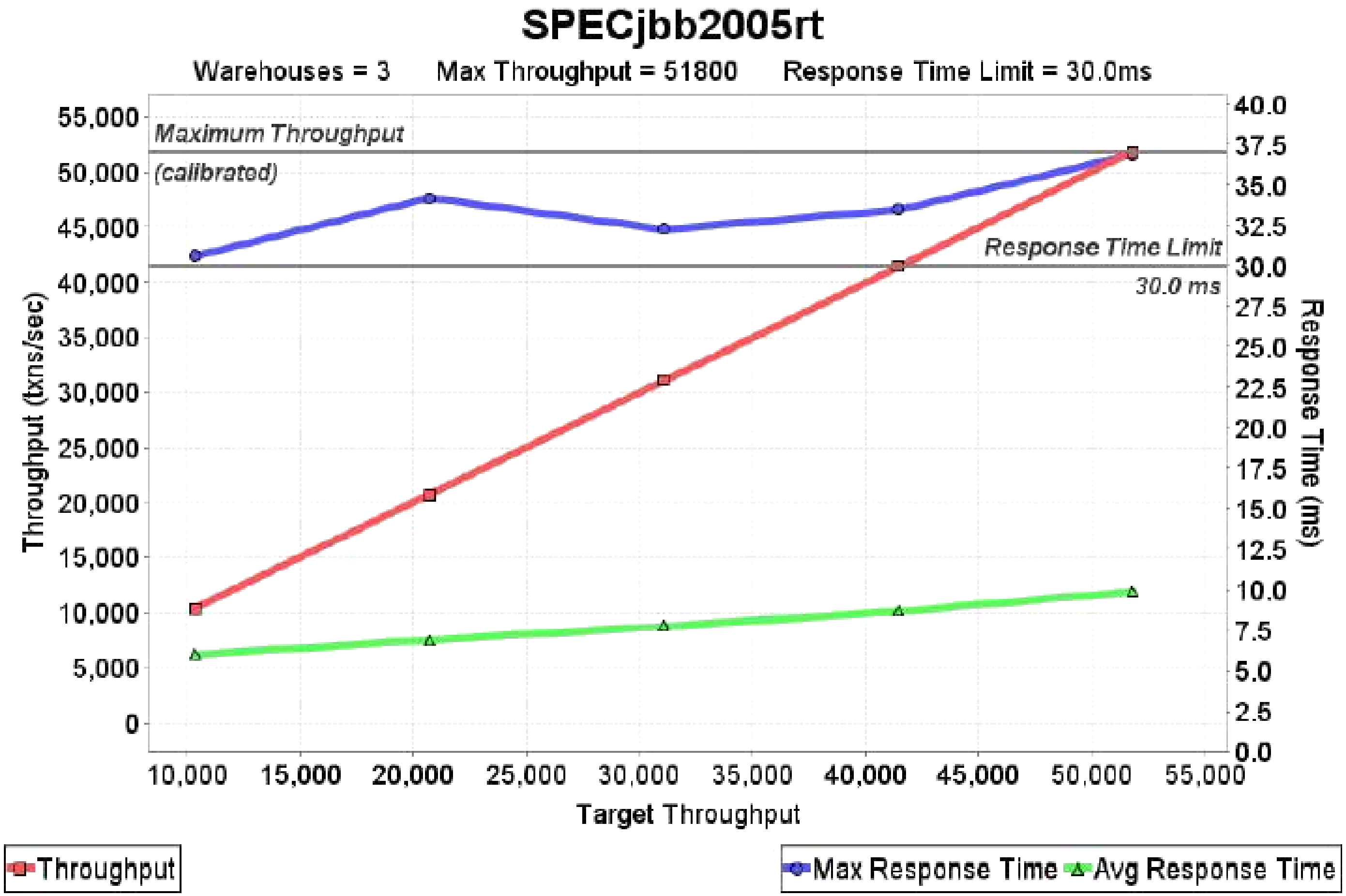


RTSJ Performances - Response time

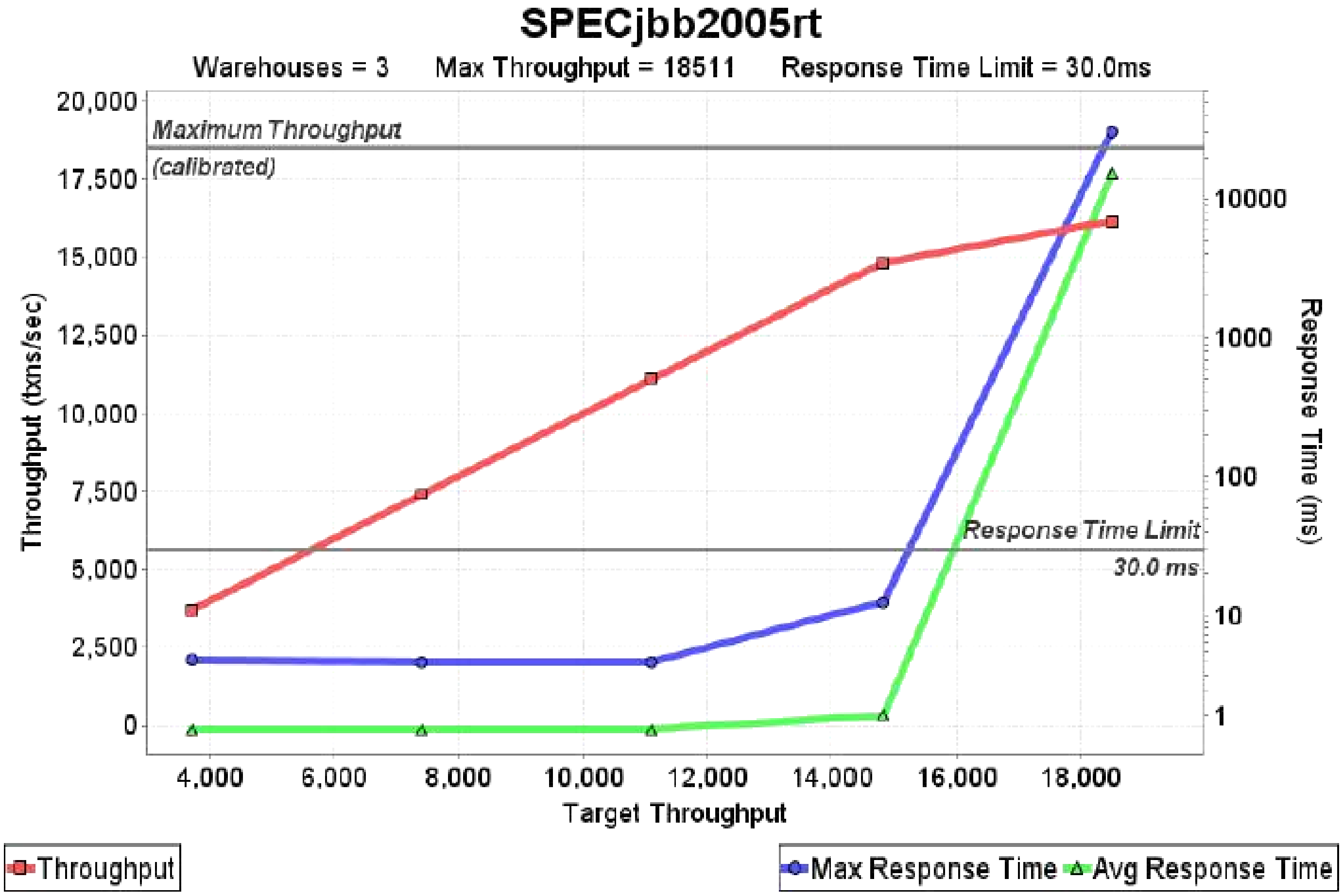




RTSJ SPECJ2005rt Results



RTSJ SPECJ2005rt Results

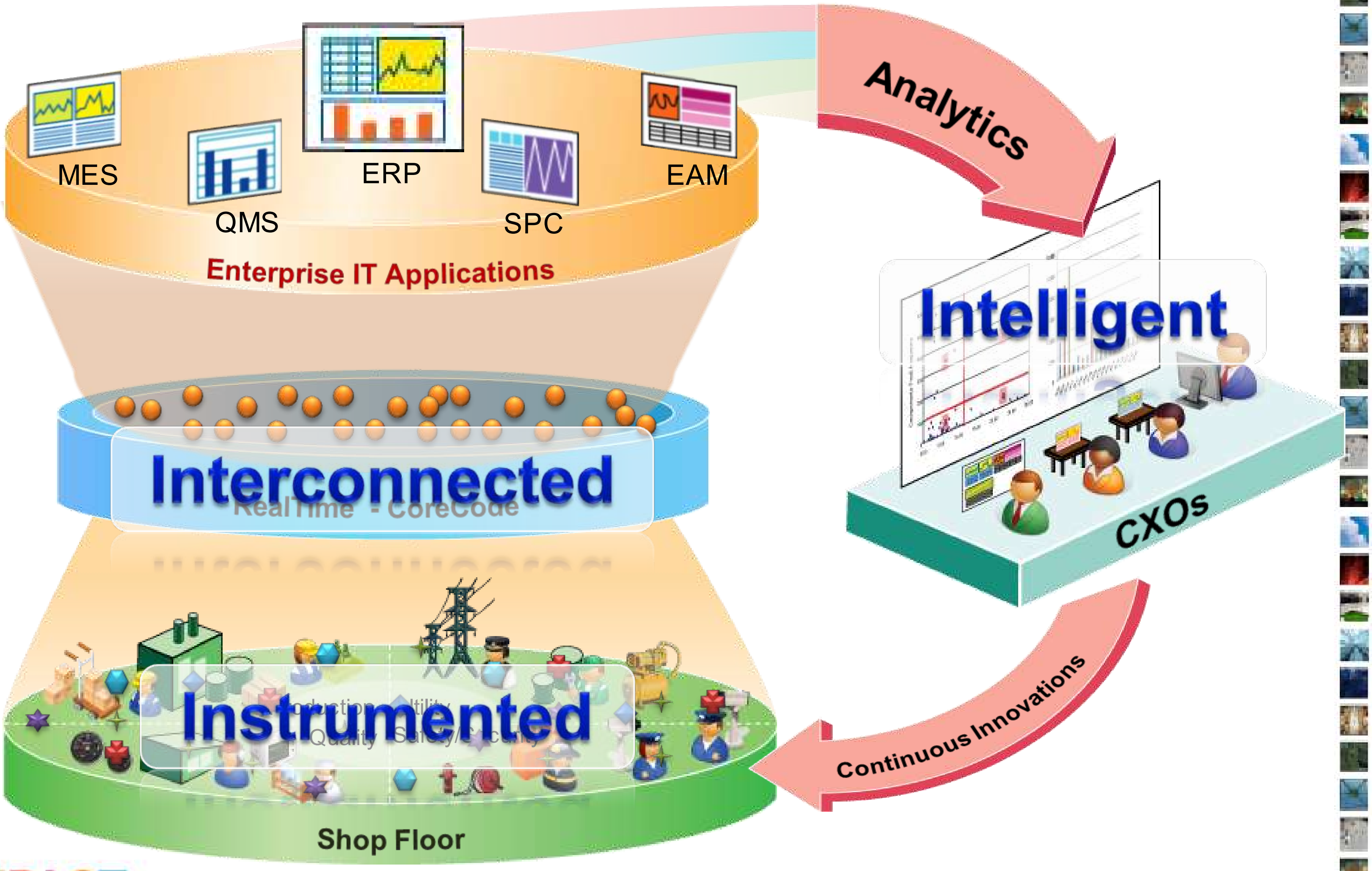


References

- Financial Trading/Analytics Systems
  - NASDAQ - fast time to market pushing drive to Java from C
- Network Routers
  - Packet routing - tighter timing typically single-digit ms
- Industrial Devices / Process Automation
  - Mitsubishi PLC, Project Blue Wonder (SUN)
  - POSCO Mg. Plant
- Military & Aerospace Industry
  - BOEING, NASA, Air Force Research Laboratory
  - DARPA – Autonomous vehicle control
- TELCO & N/W Industry
  - CISCO – IP Phones
  - Set top Boxes

And now, tell me
what does Real-Time have
to do with Namoo ?

CoreCode Vision



MES

QMS

ERP

SPC

EAM

Enterprise IT Applications

Analytics

Intelligent

CXOs

Interconnected

Real Time - CoreCode

Instrumented

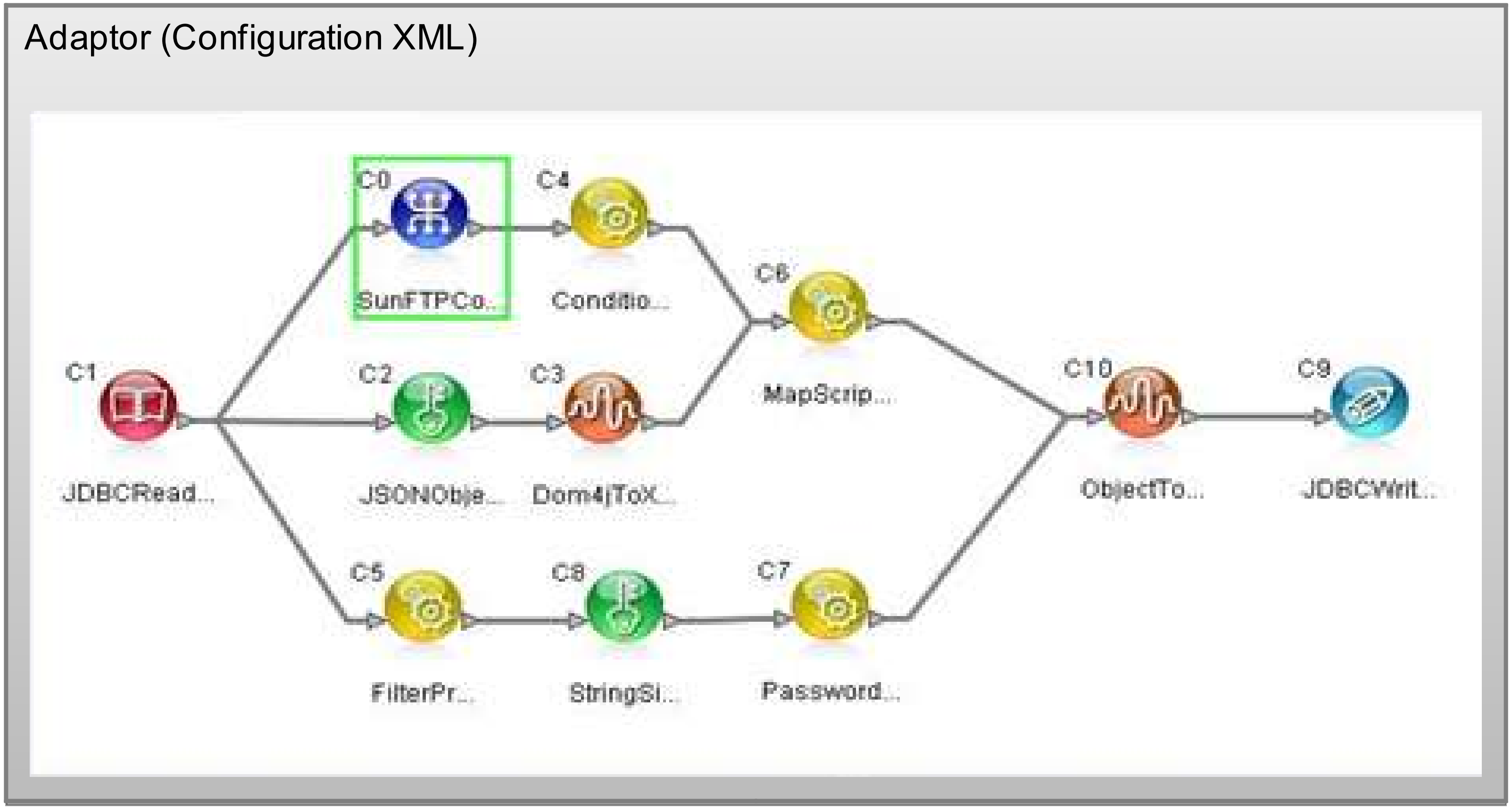
Production Utility

Quality Safety/Security

Shop Floor

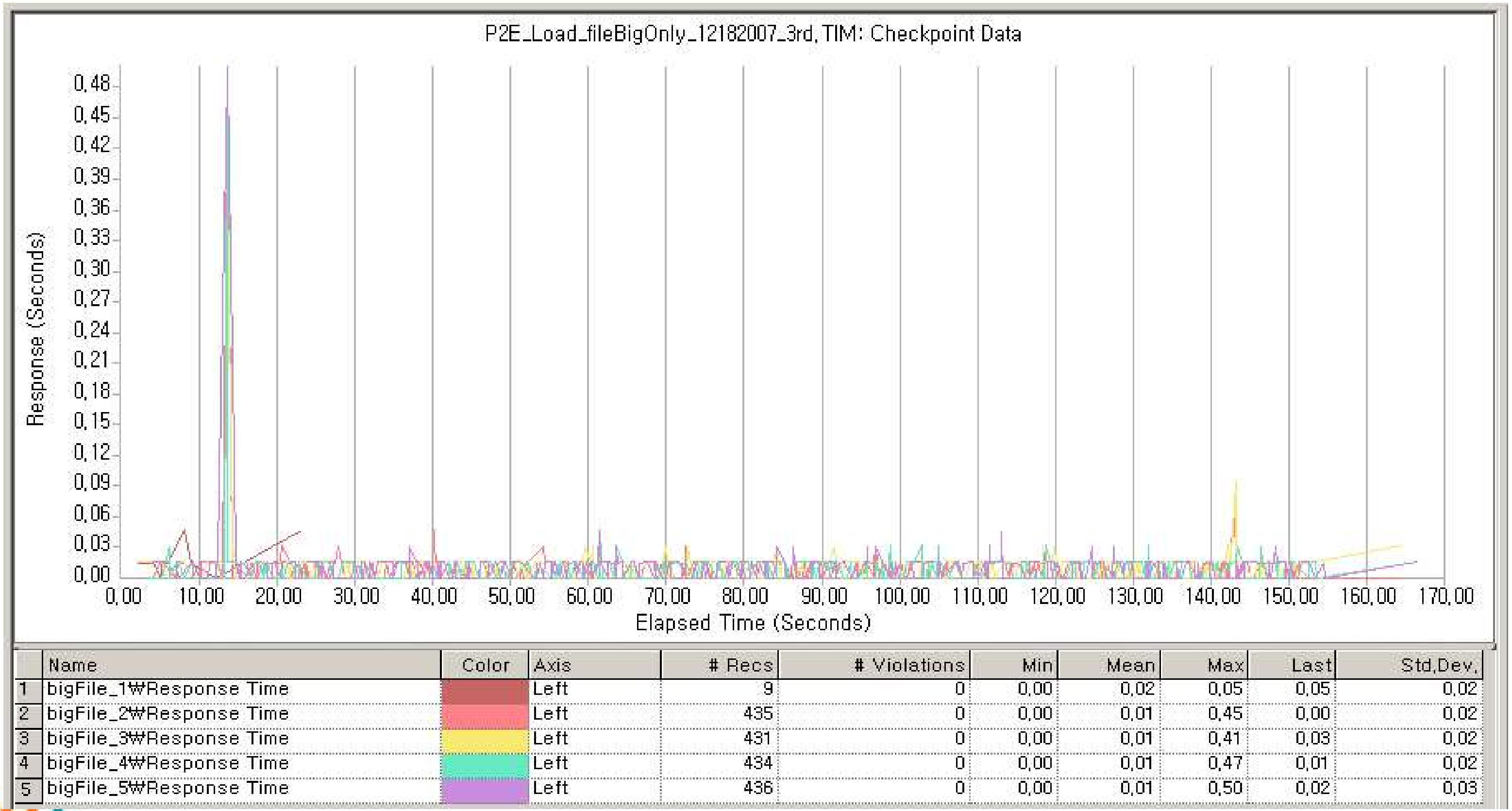
Continuous Innovations

CoreCode Adaptor Process Model



CoreCode Real world performance

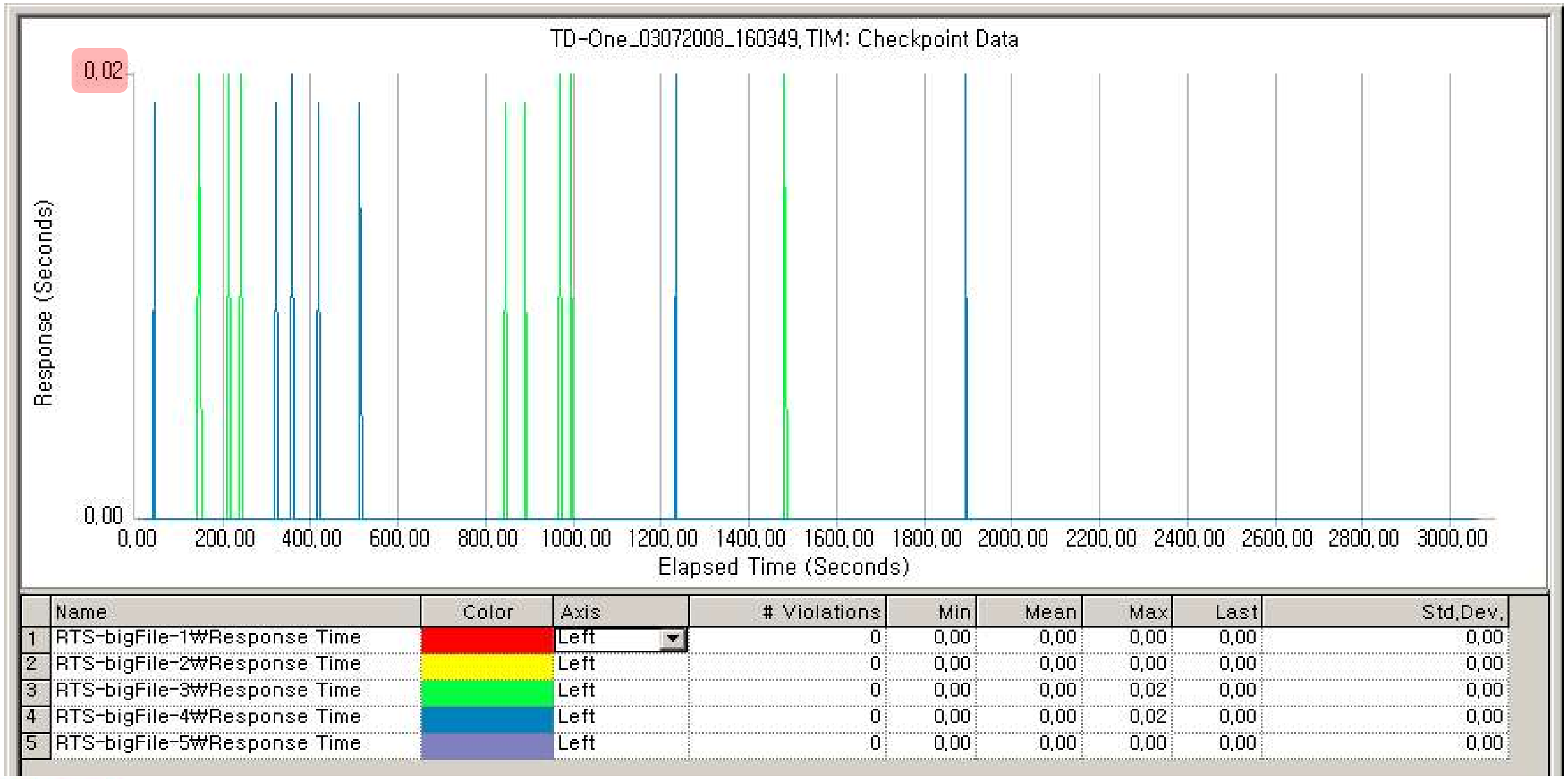
- Test Scenario
  - Read in 10K flat file -> convert it ot OrderedMap then to XML -> append to output file
- Response Time <Standard Java>



	Name	Color	Axis	# Recs	# Violations	Min	Mean	Max	Last	Std,Dev,
1	bigFile_1₩Response Time	🟥	Left	9	0	0,00	0,02	0,05	0,05	0,02
2	bigFile_2₩Response Time	🟥	Left	435	0	0,00	0,01	0,45	0,00	0,02
3	bigFile_3₩Response Time	🟨	Left	431	0	0,00	0,01	0,41	0,03	0,02
4	bigFile_4₩Response Time	🟩	Left	434	0	0,00	0,01	0,47	0,01	0,02
5	bigFile_5₩Response Time	🟪	Left	436	0	0,00	0,01	0,50	0,02	0,03

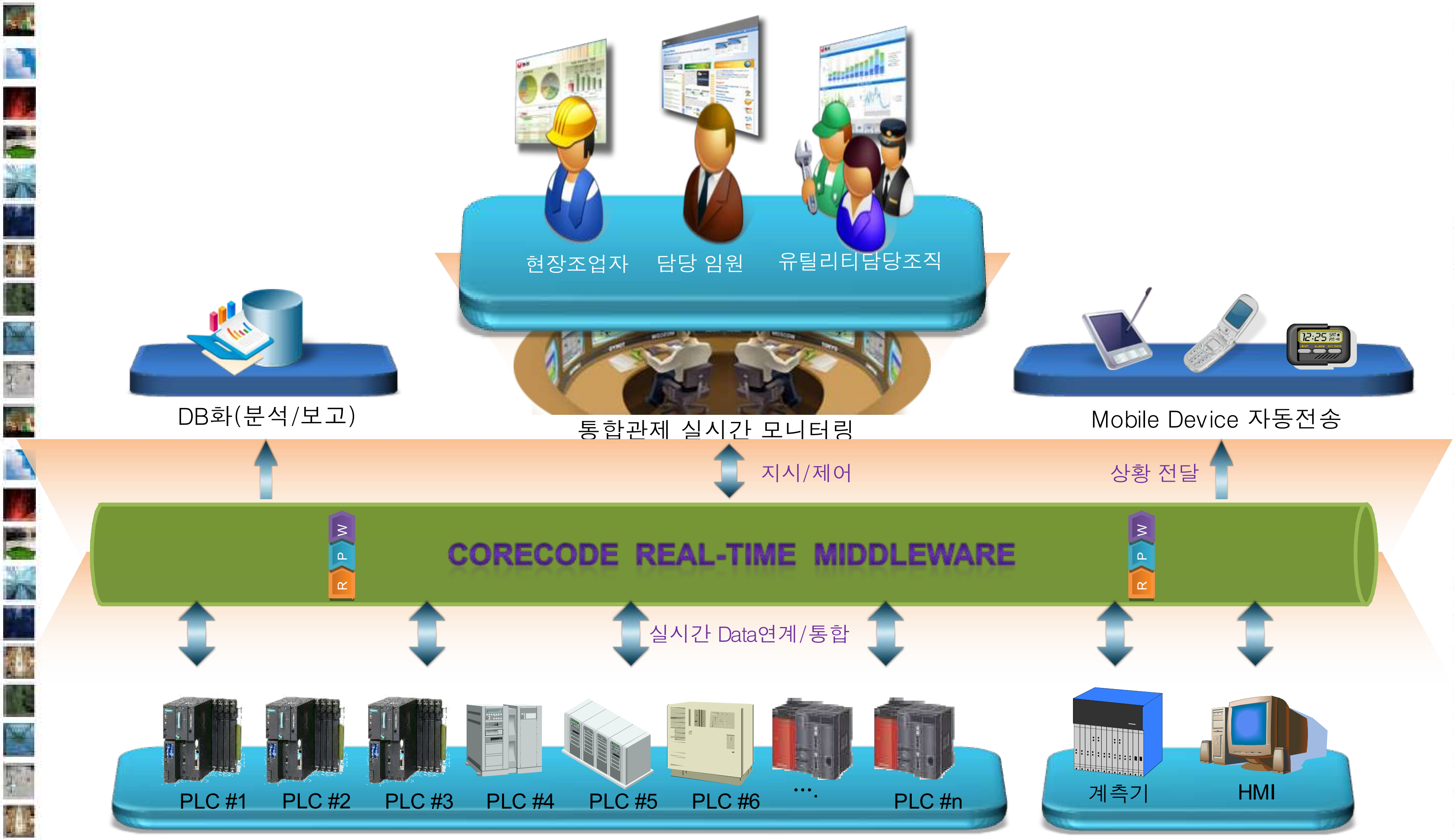
CoreCode Real world performance

- Test Scenario
  - Read in 10K flat file -> convert it ot OrderedMap then to XML -> append to output file
- Response Time - <Real-Time Java>



	Name	Color	Axis	# Violations	Min	Mean	Max	Last	Std.Dev.
1	RTS-bigFile-1₩Response Time	🟥	Left	0	0.00	0.00	0.00	0.00	0.00
2	RTS-bigFile-2₩Response Time	🟨	Left	0	0.00	0.00	0.00	0.00	0.00
3	RTS-bigFile-3₩Response Time	🟩	Left	0	0.00	0.00	0.02	0.00	0.00
4	RTS-bigFile-4₩Response Time	🟦	Left	0	0.00	0.00	0.02	0.00	0.00
5	RTS-bigFile-5₩Response Time	🟪	Left	0	0.00	0.00	0.00	0.00	0.00

CoreCode - Sensor To Boardroom



Summary

- Not a Silver Bullet, but a Sharper Tool
- The benefits of RTSJ are REAL*, not theoretical
  - Architectural Flexibility, Predictable Solution Development
- High time to dig into RTSJ
- IBM WebSphere Real Time (WRT) V2 is GA
  - http://www-306.ibm.com/software/webservers/realtime/

RTSJ delivers predictable performance!

* Pun intended

감사합니다.

Q&A