The Sun Java™ Real-Time System Meets Wall Street

Jim Clarke—Principal Engineer
Jim Connors—Systems Engineer

Sun Microsystems, Inc.
http://java.sun.com/javase/technologies/realtime.jsp

TS-1205

Applying Sun Java Real-Time System to High-Performance Financial Systems

Real-time development isn't limited to C/C++

Learn how the Java Real-Time System (Java RTS) can enable financial program trading systems to execute time-critical trades without interruption from the Garbage Collector or other Interrupts.

Agenda

The Problem?

Overview of RTSJ—JSR 001, JSR 282

Overview of the Java Real-Time System 2.0

Demonstration Design Overview

Live Demonstration

Lessons Learned and Recommendations

Agenda

The Problem?

Overview of RTSJ—JSR 001, JSR 282

Overview of the Java Real-Time System 2.0

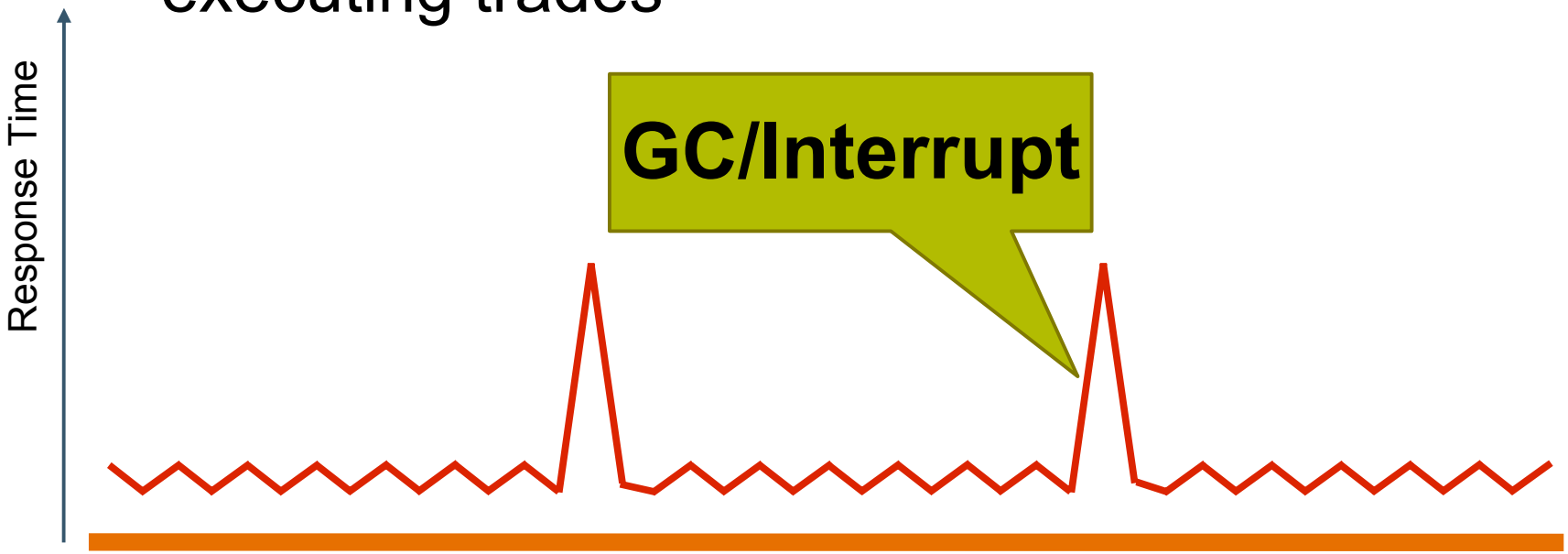
Demonstration Design Overview

Live Demonstration

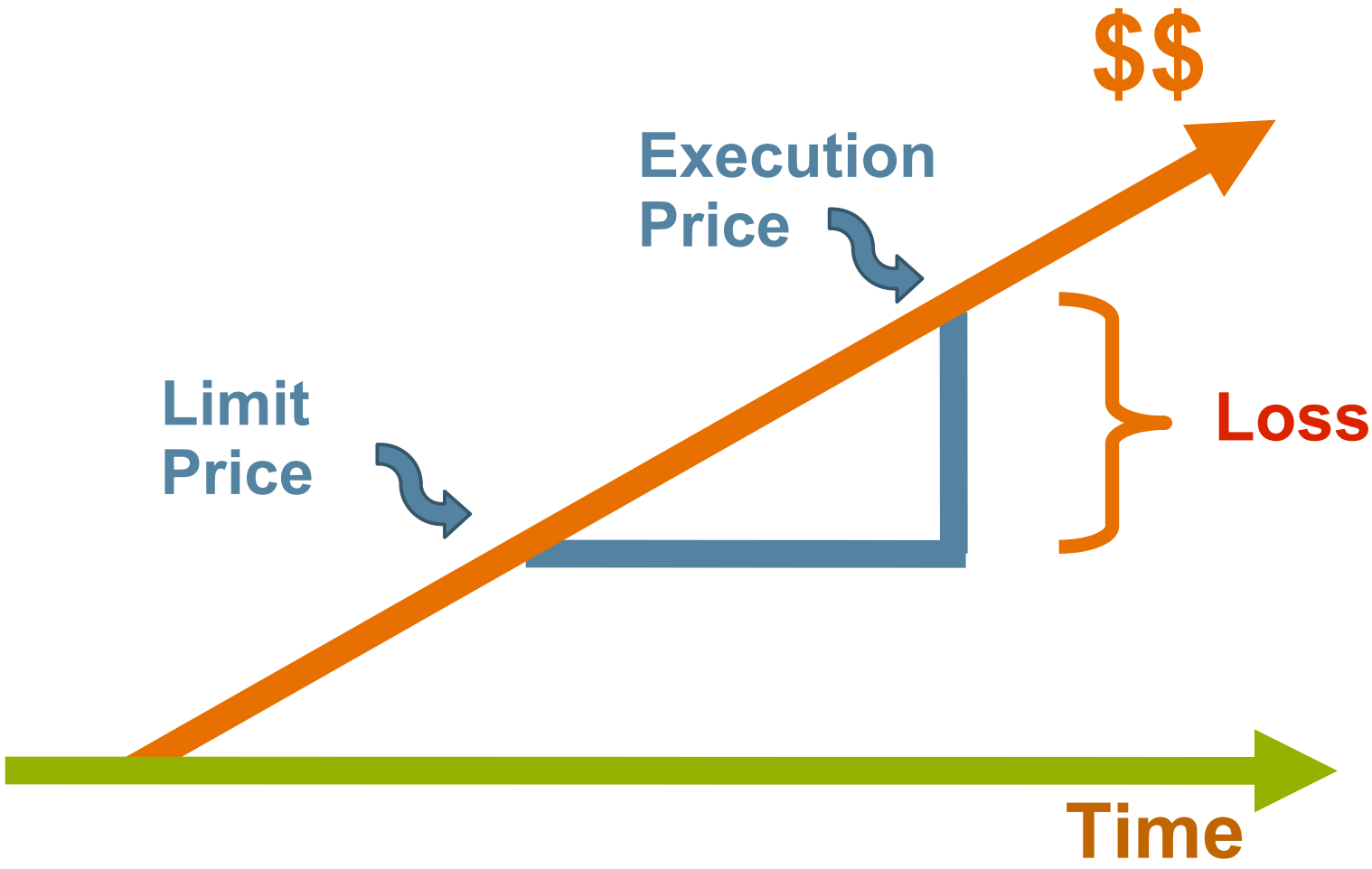
Lessons Learned and Recommendations

Problem

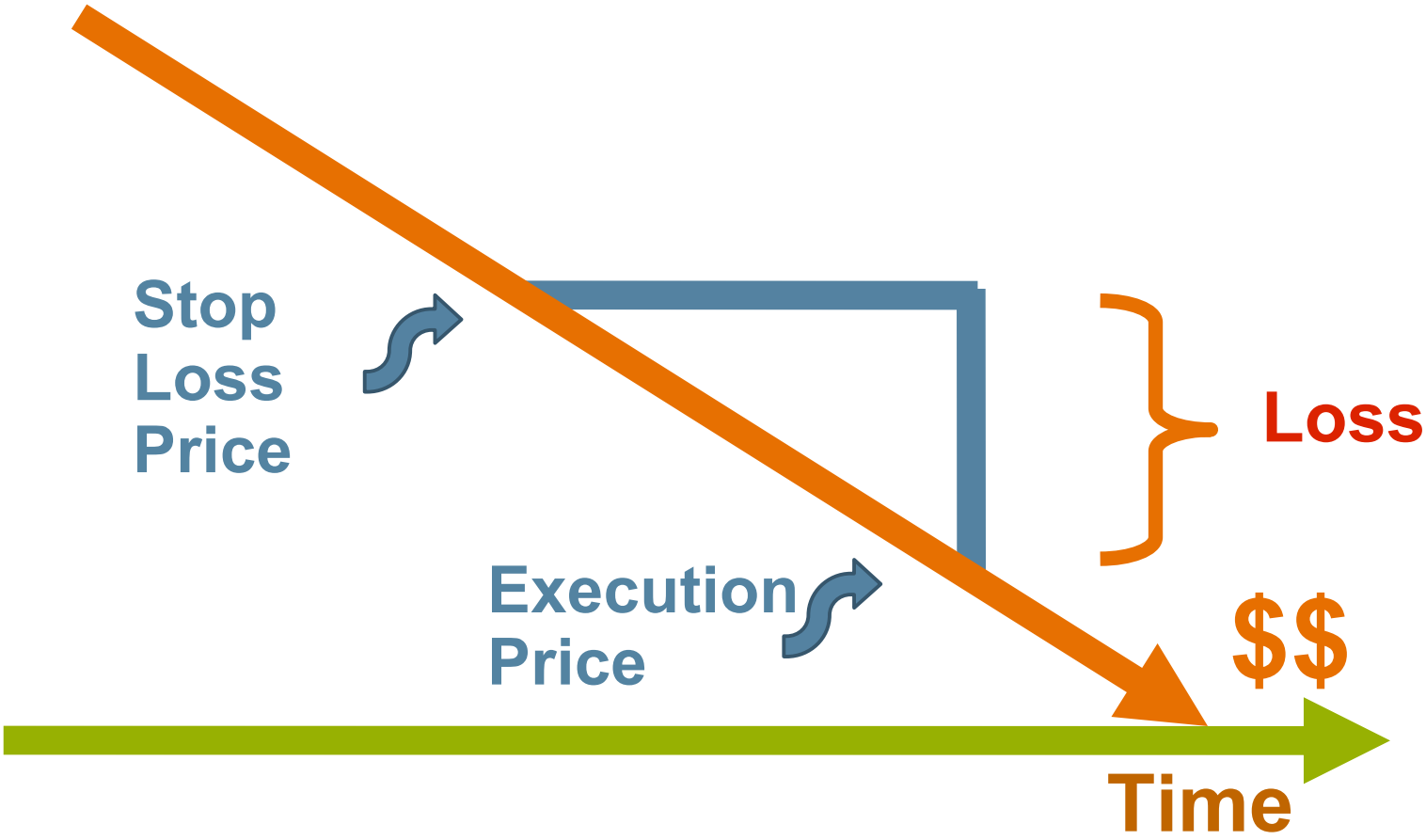
- Java platform GC, other applications/threads, and system interrupts cause delays when executing trades



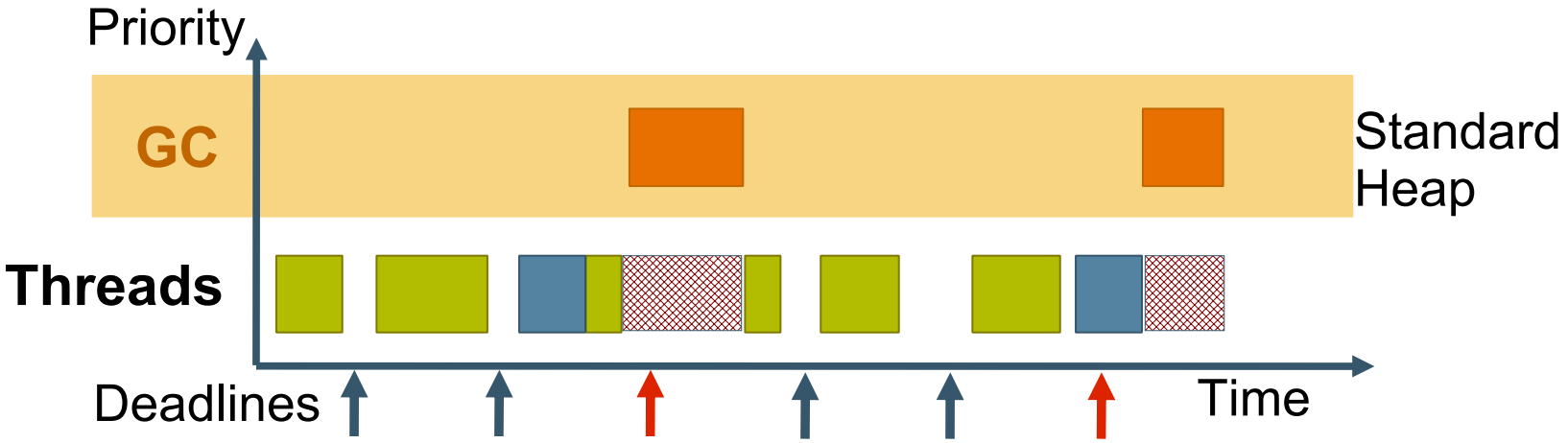
Impact—Limit Buy Order



Impact—Stop Loss Order/Limit Sell



GC Pauses Are Unpredictable



- Threads Doing Desired Work
- Unplanned Event
- Thread Pause Due to GC

Garbage Collection Can Happen at Unexpected Times—Leading to Missed Deadlines

Garbage Collection Is Very Difficult to Predict Accurately—But You Can Expect That It Will Happen at the Worst Time

Why Real-Time for Java Technology ?

- **Predictability**
  - ***Real-Time*** in this context does not mean “super-fast”—rather, it means “respond within a predictable time”
- **Architecture flexibility**
  - Instead of a flat topology for requests, discriminate between more and less important events
  - Handle most important events first, allow others to complete when possible—all on one system
- **Dealing with the Real-World**
  - “Stuff happens”—Java Real-Time System helps you deal with it

Agenda

The Problem?

Overview of RTSJ—JSR 001, JSR 282

Overview of the Java Real-Time System 2.0

Demonstration Design Overview

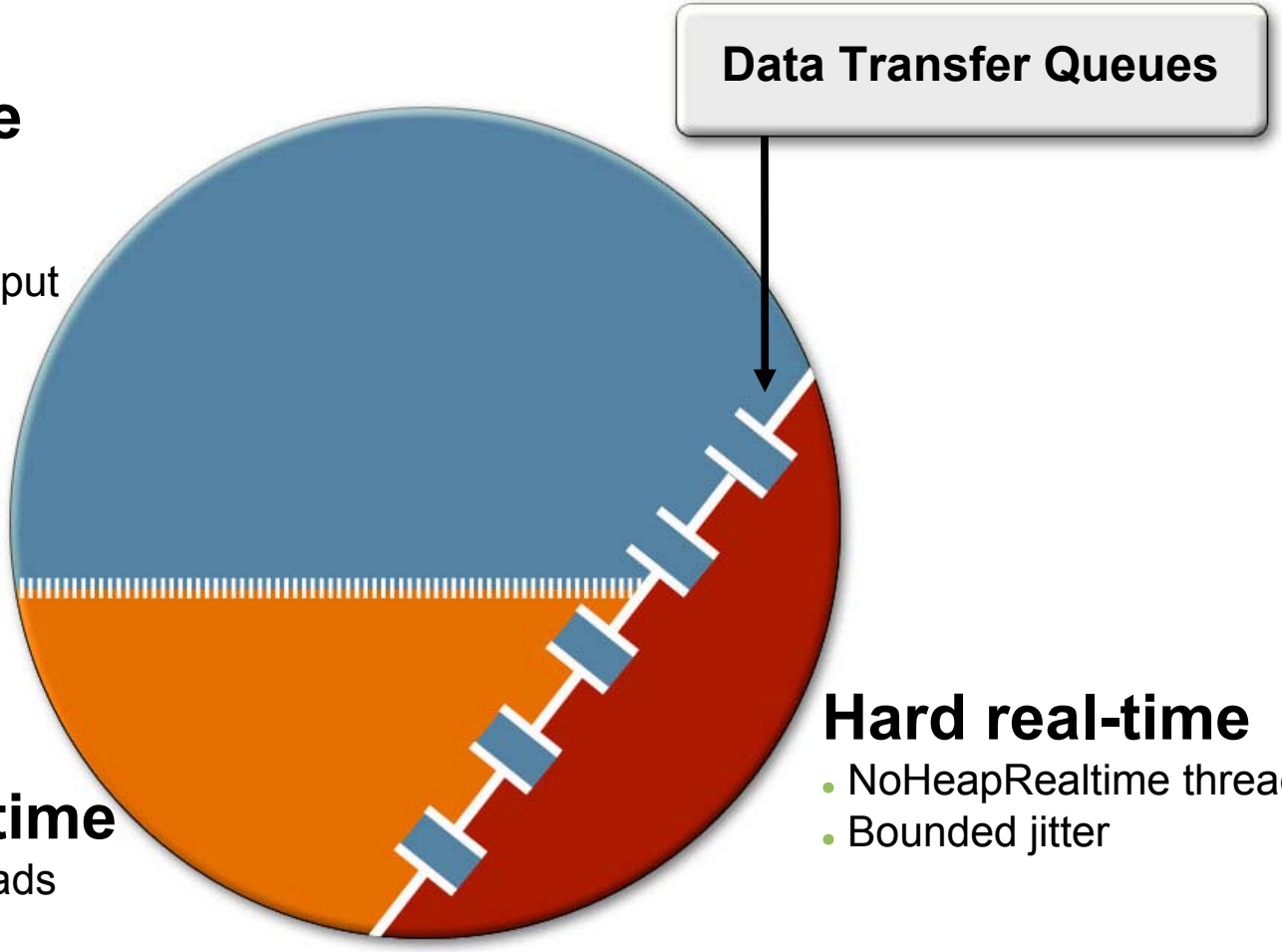
Live Demonstration

Lessons Learned and Recommendations

RTSJ Technical Highlights

- **The Real-Time Specification for Java (RTSJ) JSR 001 (1.0), JSR 282 (1.1)**
  - A standard that defines how real-time behavior must occur within Java technology
- **Attributes**
  - Not a silver bullet or magic wand, but a much sharper tool
  - Higher-level, portable, address temporal concerns such as costs and deadlines
  - 100% Java technology
  - Developed by many experts

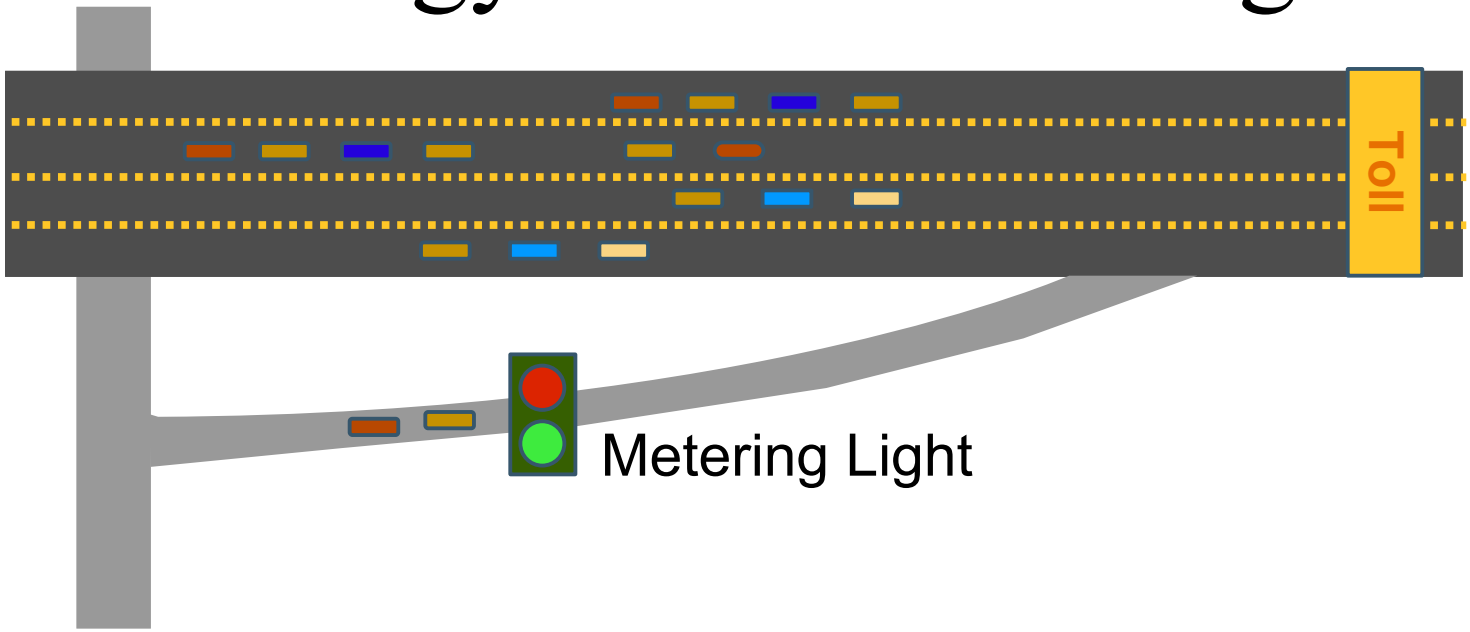
RTSJ System Model



New Memory Management Models

- What distinguishes each memory model is the life-cycle of objects
- Memory models defined by RTSJ
  - **Heap**
    - ***GC'd***
  - ***Immortal***
    - ***Not GC'd, never reclaimed***
  - ***Scoped***
    - ***Not GC'd, but reclaimed when no longer used***

RT Analogy: Commuter Highway

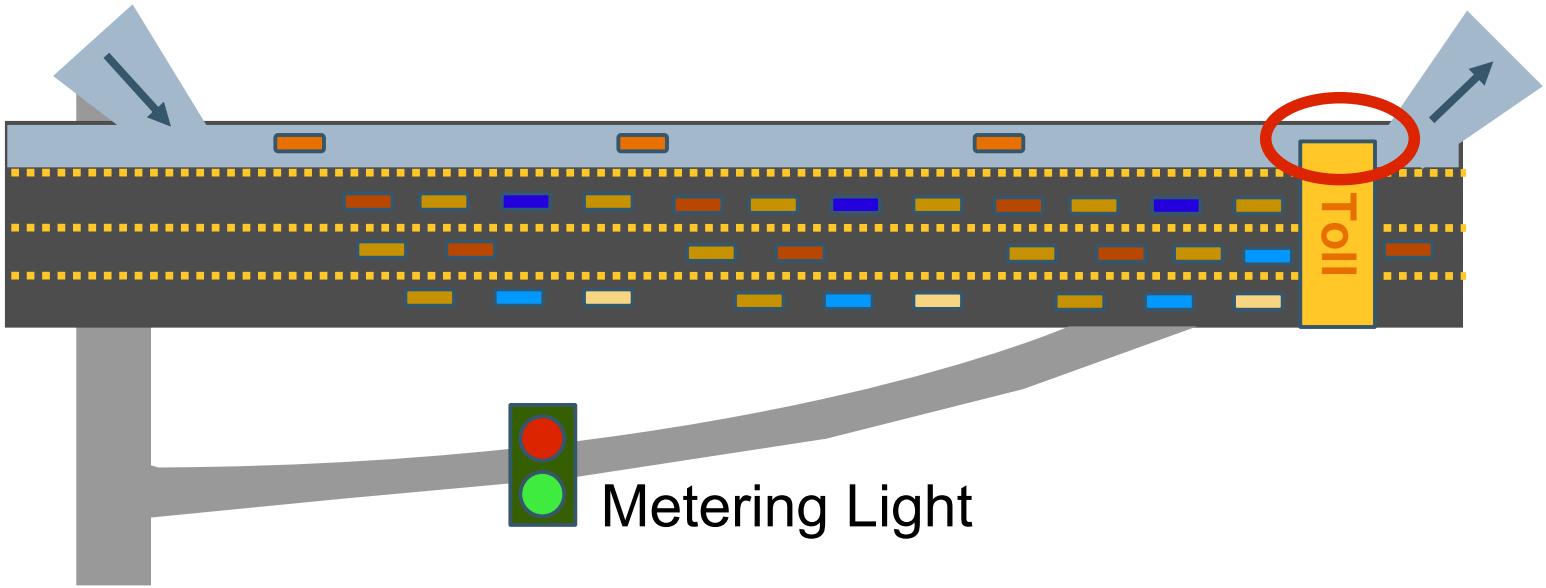


Commuter Highway Uses Access Control via Metering Lights to Maintain Traffic Flow—or Throughput

Many Production Systems Use a Similar Approach by Minimizing the Load on Servers to Ensure Throughput via Less “Traffic”

RT Analogy: Real-Time Threads

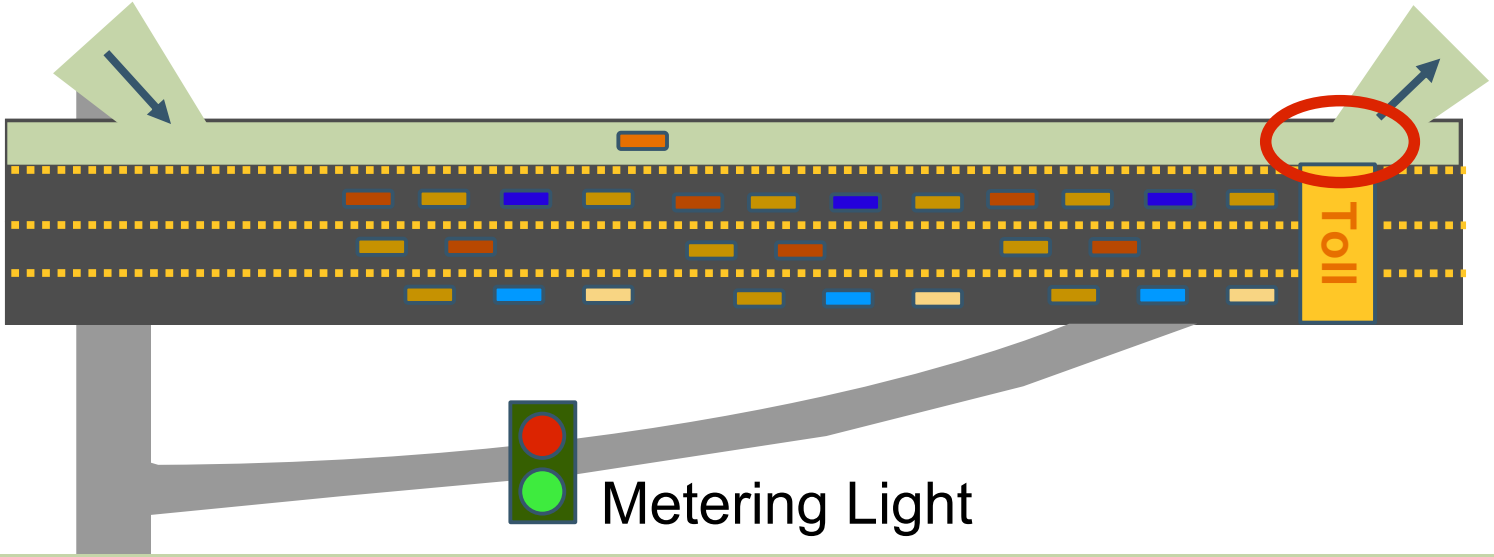
Real-Time Threads Are Like the Commuter Lane (Assume You Get On and Off w/out Merging)—Your Travel Time Is Better Here



However, This Lane May Be Shared With Other Cars (Threads) and It Can Fill Up and Slow Down if Too Much Traffic Happens Here

RT Analogy: No-Heap Real-Time Threads

No-Heap Real-Time Threads Are Like Your Own Personal Lane; You Can Predict Your "Trip" Precisely With No Interruptions



RTSJ Allows You to Use NHRT, RTT, and Normal Threadsall on the Same System to Meet Your Requirements—But Not That for a Given "Highway" (System), Only So Many Cars (Work) Can Pass Through the System

Agenda

The Problem?

Overview of RTSJ—JSR 001, JSR 282

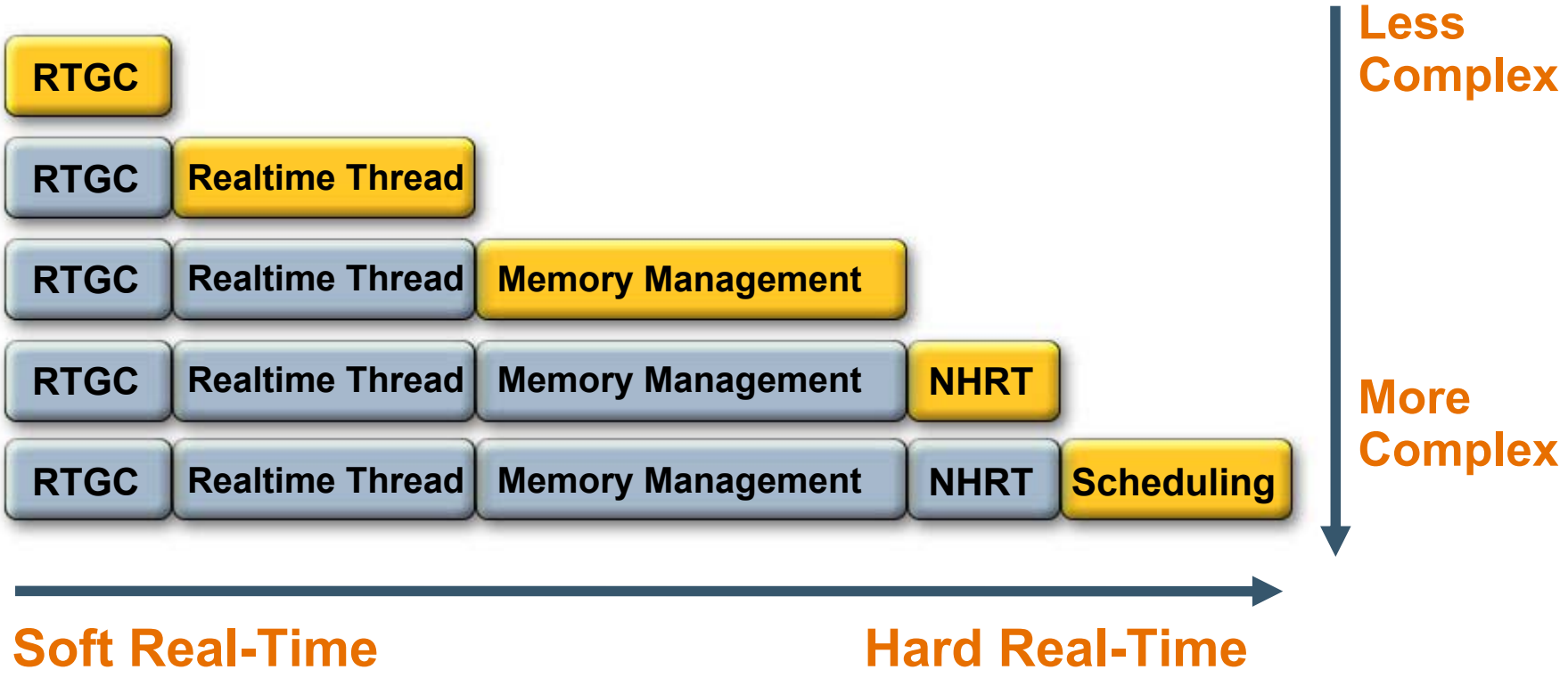
Overview of the Java Real-Time System 2.0

Demonstration Design Overview

Live Demonstration

Lessons Learned and Recommendations

Java RTS Usage Options



Java RTS—RTGC

Apply policy to garbage collection

- **Goal: Smallest latencies for high priority GC'd threads**
  - Defer GC work for high-priority threads
- **Advantages**
  - Scalable: no issues with multi-processor support
  - Flexible: works with different policies for low priority RT threads
    - GC overhead can be paid by these threads if total memory consumption goes up
    - More efficient policies possible: could pause certain threads
    - Simpler policies like running the GC on a dedicated CPU

Programming Soft Real-Time in Java RTS

Step #1:

Replace:

```
Thread T = new Thread(myRunnable);
```

with

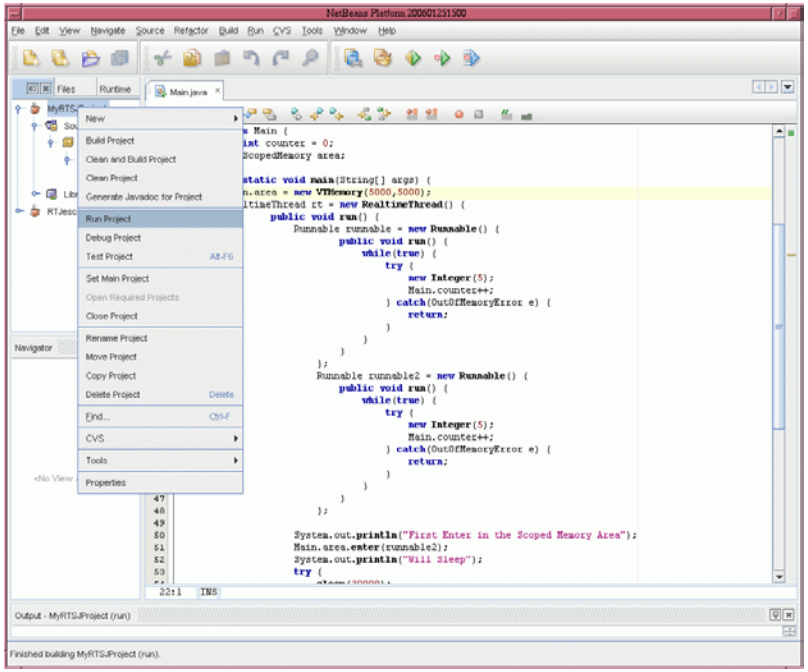
```
RealtimeThread RT =
      new RealtimeThread(..., myRunnable);
```

Then, Step #2…

!There is no step #2!

NetBeans™ Integrated Development Environment (IDE) Support

- Download the Java RTS plug-in, and:
  - > Cross-develop on the host
    - > Deploy over the network
      - > Execute on the target
- …from the NetBeans IDE



Further Reducing Latencies and Jitter in Java RTS

- Class **pre-initialization**
  - Load and initialize all critical classes at initialization
- Class **pre-compilation**
  - Pre-compile all critical code at initialization
- Coupled with Solaris capabilities…
  - **Processor sets**
    - Bind critical tasks to **dedicated** CPUs
  - **Interrupt sheltering**
    - Field hardware interrupts on non-critical CPUs

Java Real-Time System 2.0—Features

- Java Platform, Standard Edition (Java SE platform) 5.0
- Solaris™ Operating System (Solaris OS) 10
- x86, x64, and SPARC® Architecture
- Real-Time Garbage Collector
- NetBeans Module

Agenda

The Problem?

Overview of RTSJ—JSR 001, JSR 282

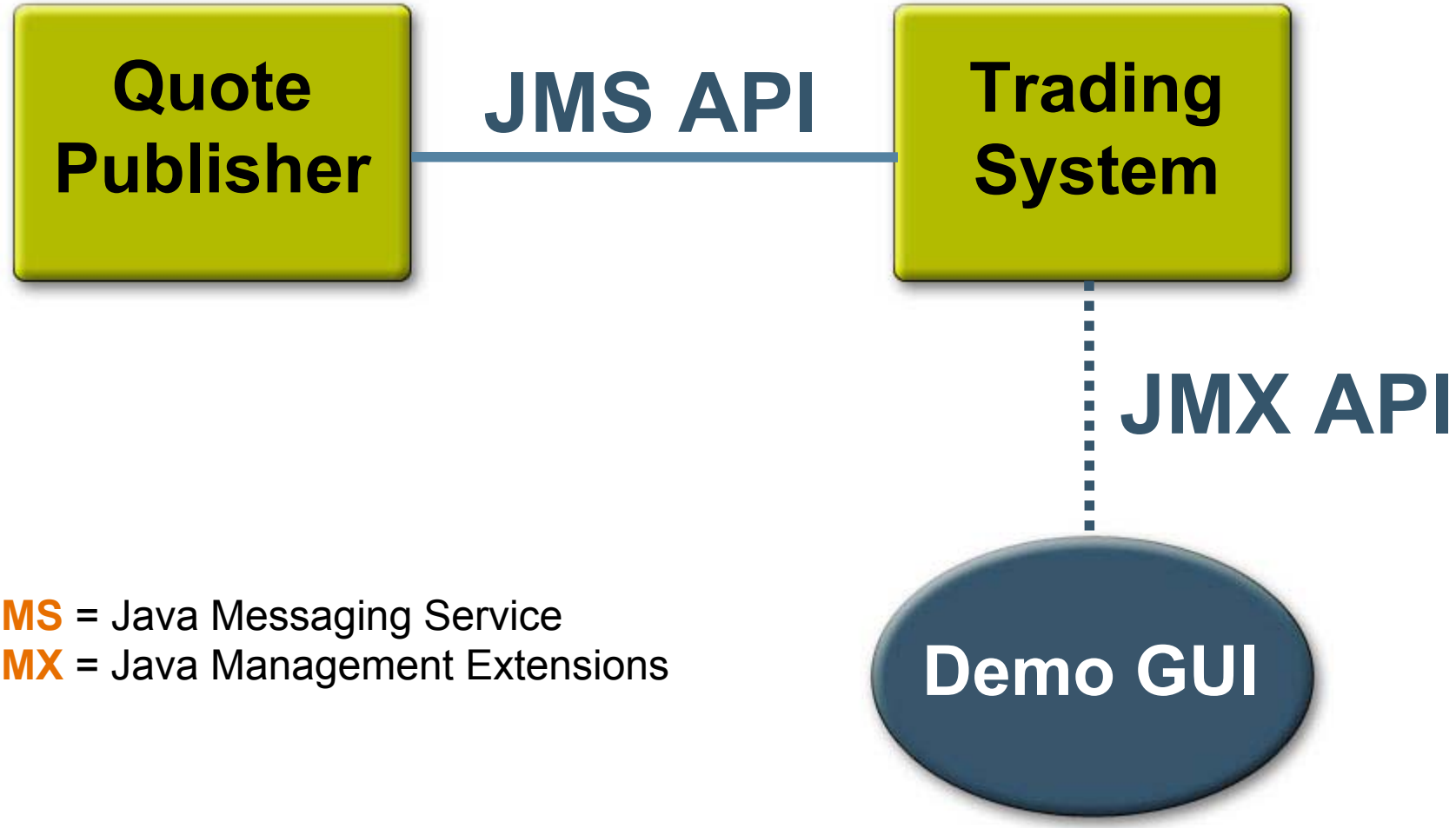
Overview of the Java Real-Time System 2.0

Demonstration Design Overview

Live Demonstration

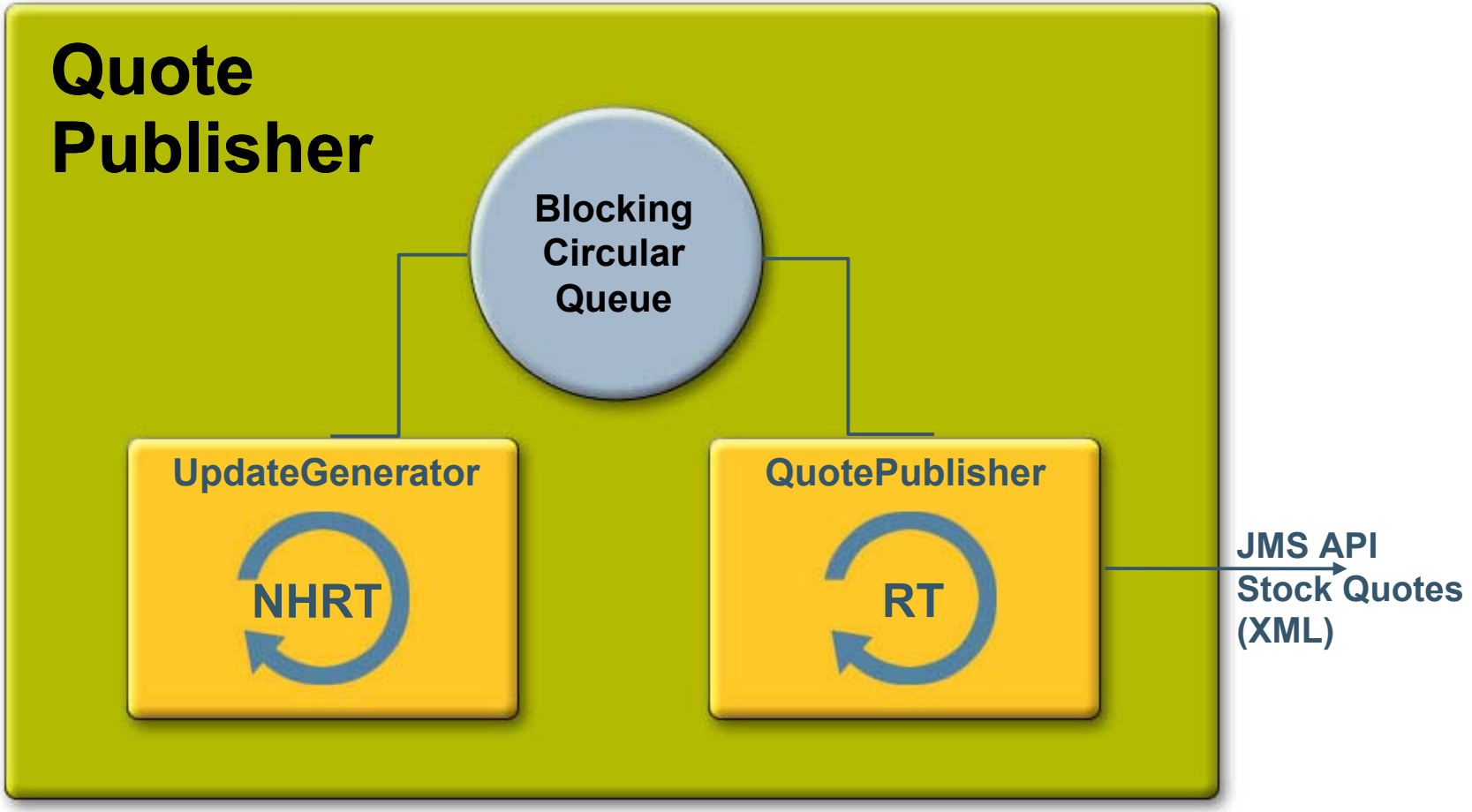
Lessons Learned and Recommendations

Demo Architecture

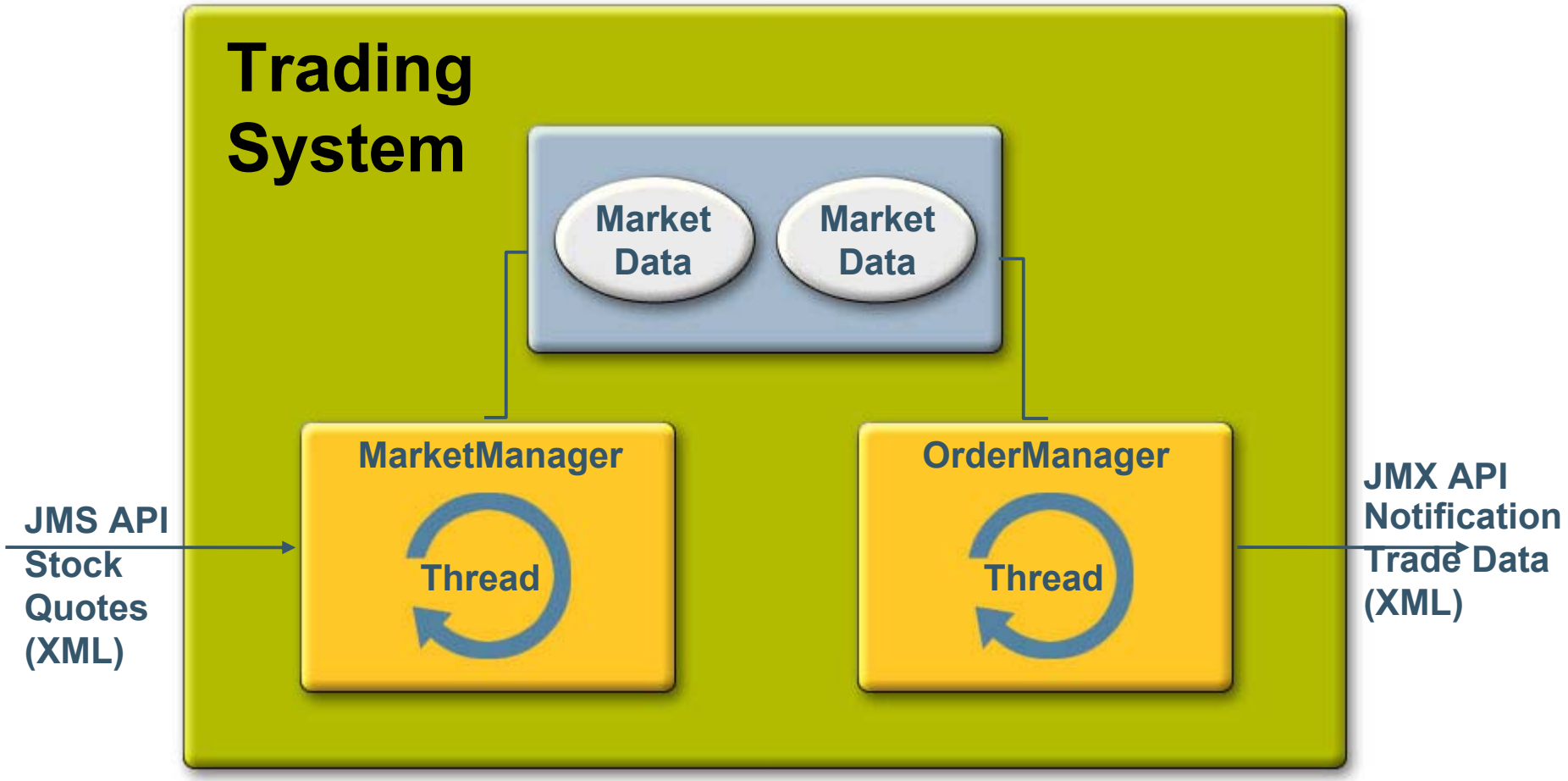


JMS = Java Messaging Service
JMX = Java Management Extensions

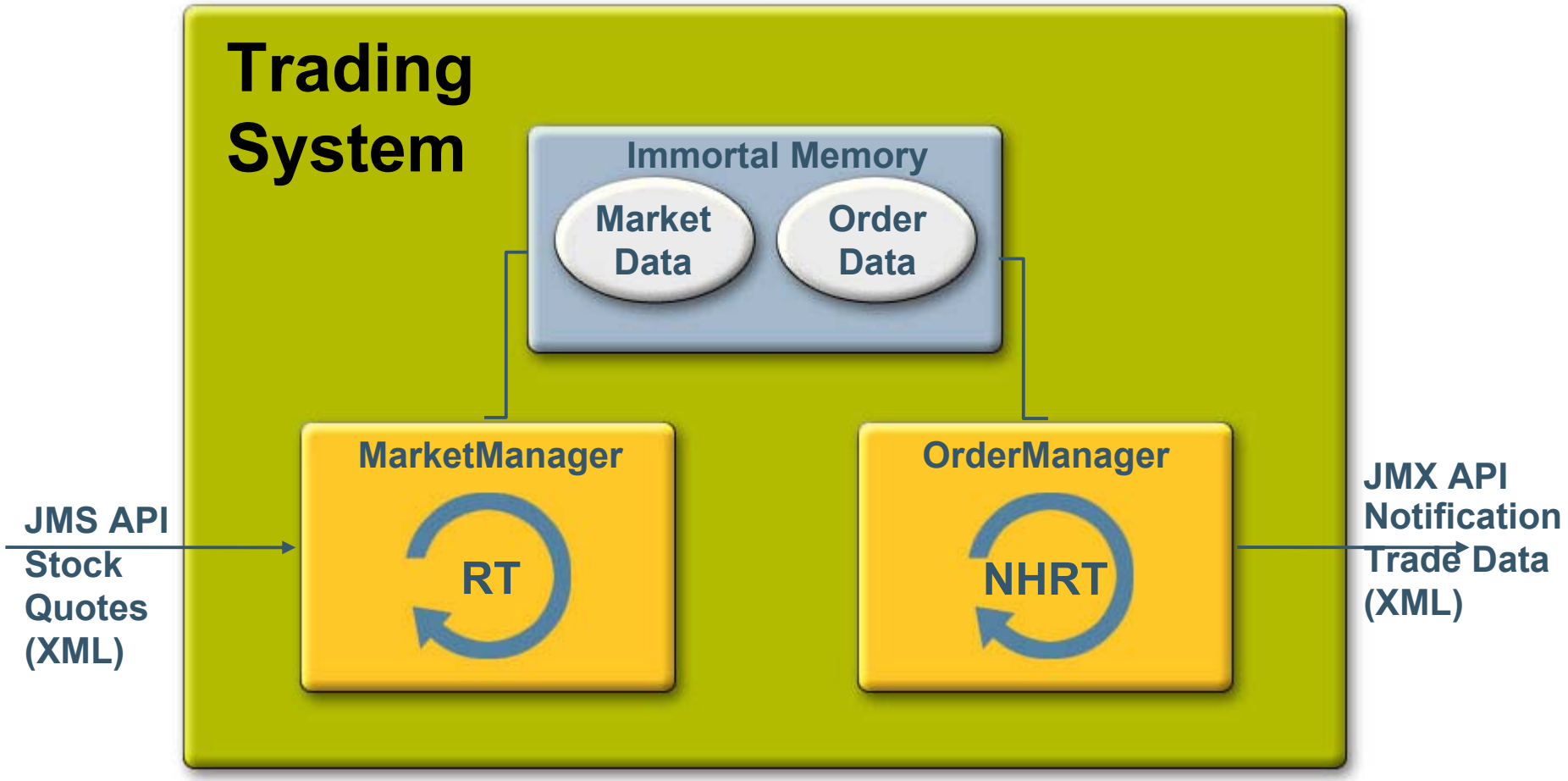
Demo Architecture—Quote Publisher



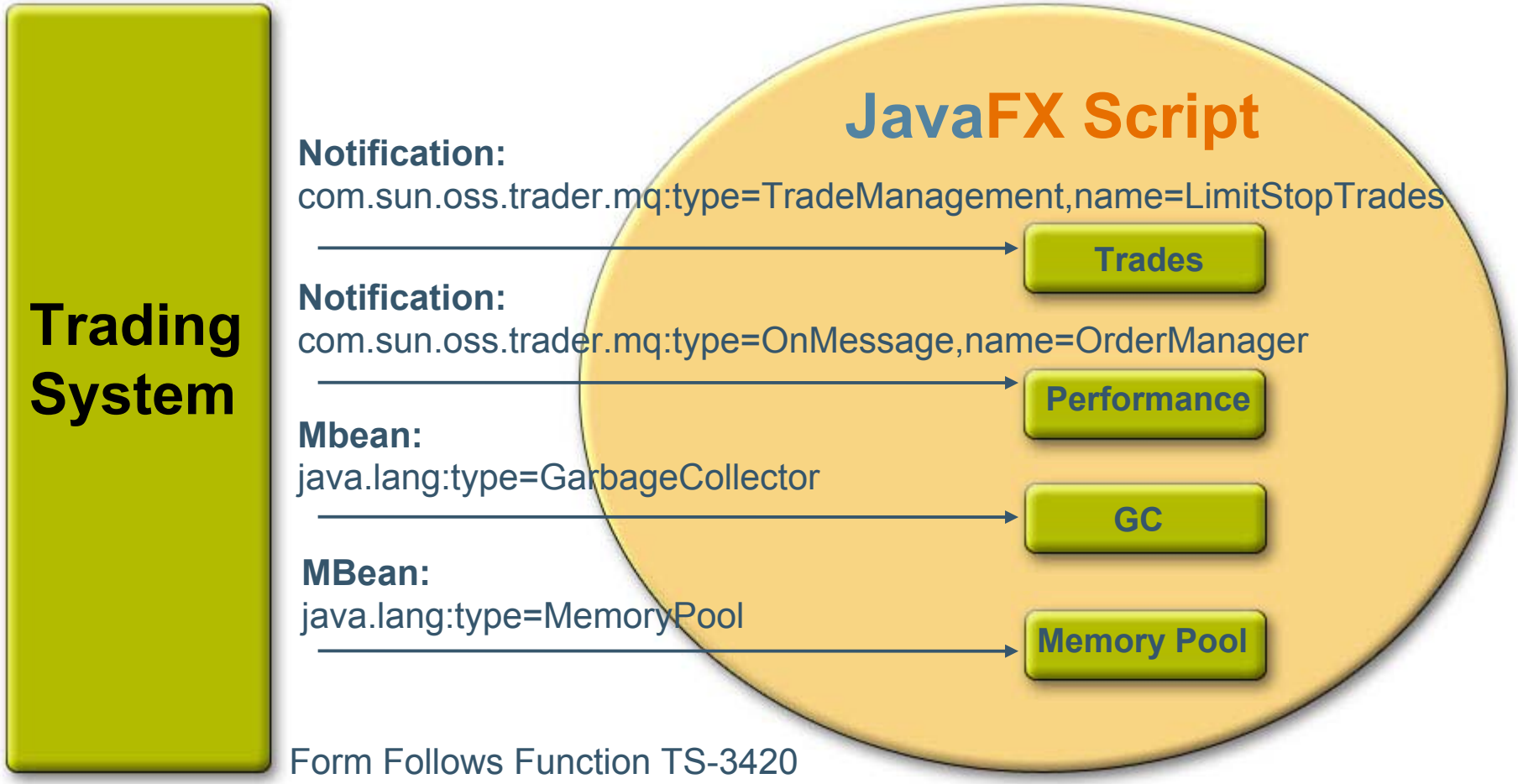
Demo Architecture—Trading System—Non-Real-Time Version



Demo Architecture—Trading System—Real-Time Version



Demo Architecture—JavaFX™ Technology



Form Follows Function TS-3420
Wed. 4:10PM and Thurs. 1:30PM

Agenda

The Problem?

Overview of RTSJ—JSR 001, JSR 282

Overview of the Java Real-Time System 2.0

Demonstration Design Overview

Live Demonstration

Lessons Learned and Recommendations

Demo Results (Non-Real-Time Version)

- The market moves fast and continuously
- Price thresholds are missed by the order manager:
  - Limit orders cannot be traded (very bad situation)
  - Stop orders trade beyond their desired value (lose money)
  - **Graph dips down into the negative region = lost money!**

Demo Results (Real-Time Version)

- The market moves fast and continuously
- Price thresholds are met:
  - Limit orders always trade (very good)
  - Stop orders trade at their set values (no money lost)
  - **Graph stays even (flat) = no missed trades, no money lost**

DEMO

RT Financial Demo

Agenda

The Problem?

Overview of RTSJ—JSR 001, JSR 282

Overview of the Java Real-Time System 2.0

Demonstration Design Overview

Live Demonstration

Lessons Learned and Recommendations

Lessons Learned

- Allocate time for **training**
- Hardware/OS
  - Many laptop BIOS'es do not fully expose access to APIC
    - No high-resolution timer
  - **Multi-CPU/core** machines strongly recommended
  - Solaris software RBAC and Java RTS
    - Bash(1) shell not RBAC aware, ***must use pfexec***
- Memory Usage and Access, ***plan***!
  - Where, ***immortal, heap, scoped***?
  - Calculate runtime memory usage
  - Caution when using **immutable** objects, like String

Lessons Learned—NoHeapRealtimeThread Programming

- **IllegalAssignmentError/MemoryAccessError**
  - Not caught by default
  - Generic try/catch around suspicious code
- Java **library** classes
  - Many cause memory errors inside NHRTs
  - Need to re-invent the wheel
- **Java Message Service (JMS)** currently not a good candidate for NHRT
  - Need RT enabled JMS API???
- Try Running RealTimeThreads at a priority higher than GC Thread before jumping into NHRT

More Java RTS Information

- The RTSJ Specification
  - http://www.rtsj.org/specjavadoc/book_index.html
- Sun's Java RTS Page
  - http://java.sun.com/javase/technologies/realtime.jsp
- Java RTS Article by Eric Bruno
  - http://www.devx.com/Java/Article/33475
- Java RTS Articles by Greg Bollella
  - http://ddj.com/dept/java/193402050
  - http://java.sun.com/developer/technicalArticles/Interviews/Bollella_qa2.html

Q&A

Jim Clarke jim.clarke@sun.com
Jim Connors jim.connors@sun.com

The Sun Java Real-Time System Meets Wall Street

Jim Clarke—Principal Engineer
Jim Connors—Systems Engineer

Sun Microsystems, Inc.
http://java.sun.com/javase/technologies/realtime.jsp

TS-1205