



Java SE Real-Time in the Financial Community

Eric Bruno
Systems Engineer
Sun Microsystems, Inc.

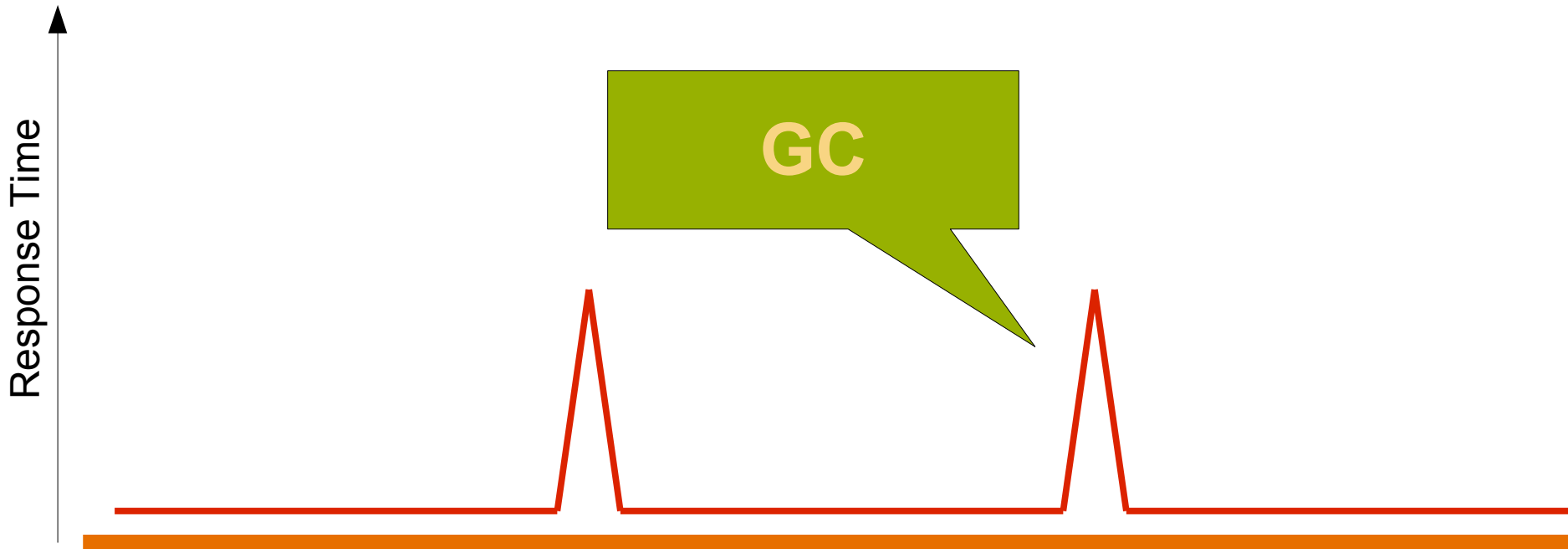


Agenda

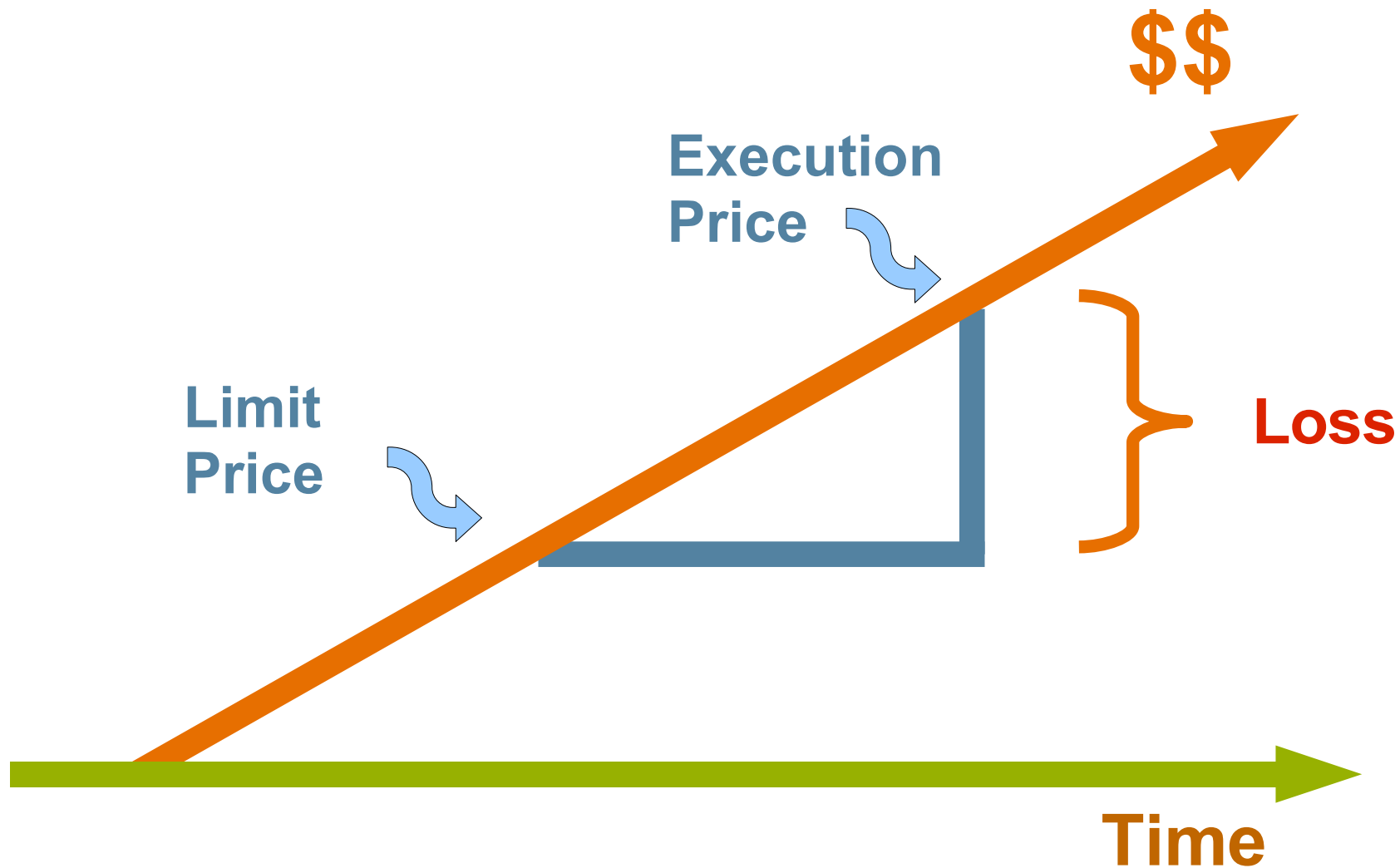
- Real-time overview
- Problem Space
- Demo Architecture
- Demo
- More information, next steps...

Problem:

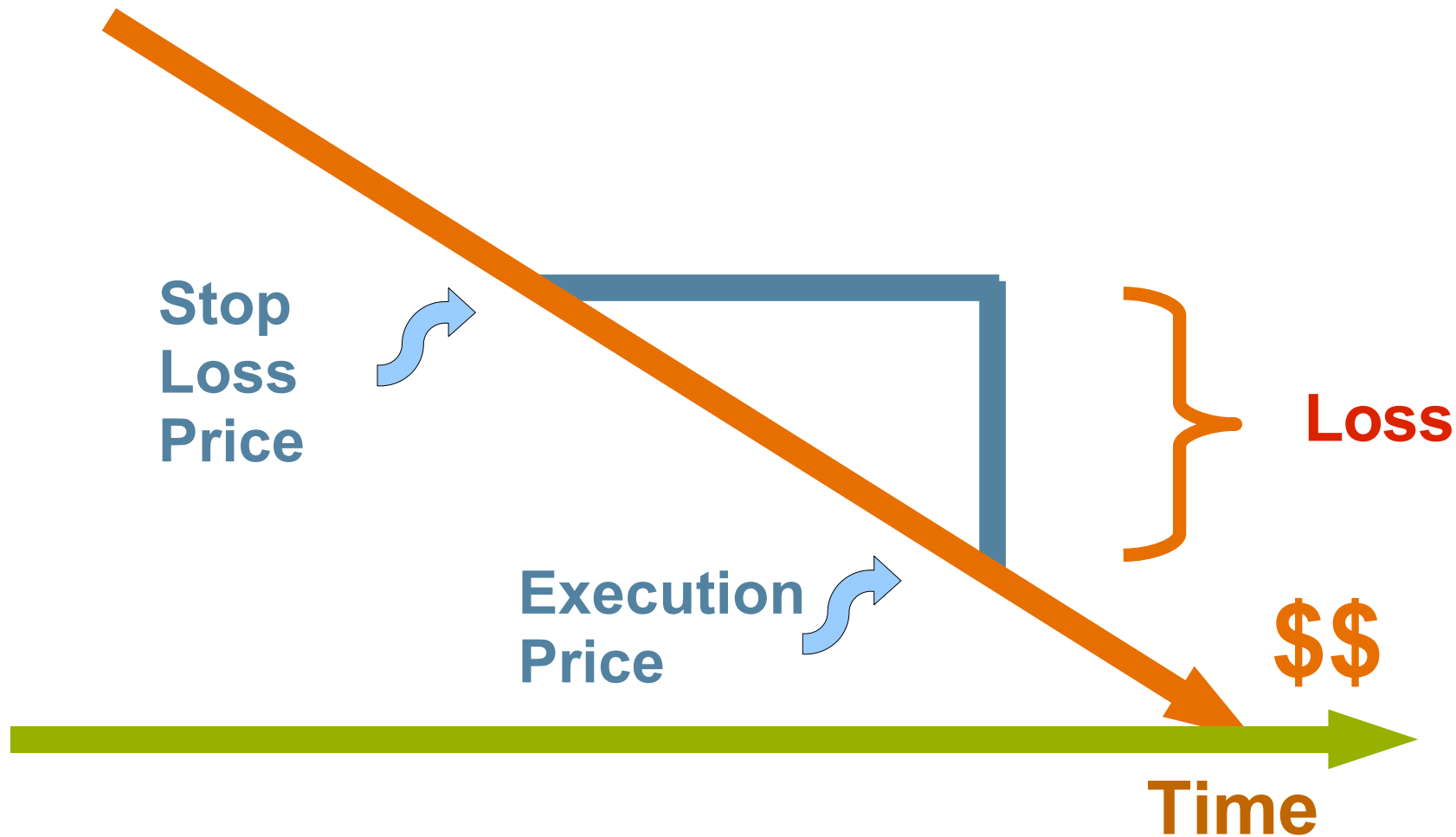
- Java GC and *other* interrupts cause latencies when executing trades.



Impact – Limit Buy Order



Impact – Stop Loss Order/Limit Sell



Why Real-time Java?

- Predictability
 - > *Real-Time* does not mean “real fast”
 - > It means “respond within a predictable time”
- Better design/architecture choices
 - > Discriminate between more and less important events
 - > Handle most important events first, allow others to complete when possible – all on one system

RTSJ and the Java RTS

- JSR-001: The Real-Time Specification for Java
 - > A standard that defines how real-time behavior must occur within Java
 - > True real-time Java implementations adhere to this specification (i.e. Sun's Java RTS)
- Attributes
 - > 100% Java
 - > Soft and hard real-time support
 - > High-resolution timers
 - > Latency: 20 microseconds (+/- 5 microseconds)
 - > Developed by many experts from:
 - > RT-scheduling, embedded systems design, Ada design, Java design, embedded processor design, real-time systems design, academia, RTOS design, etc.

Using Java RTS

- Real-time garbage collector
- New threading models
 - > RealtimeThread – almost no code changes
 - > NoHeapRealtimeThread – some code changes
- New memory models
- Scheduling options

Soft Real-time

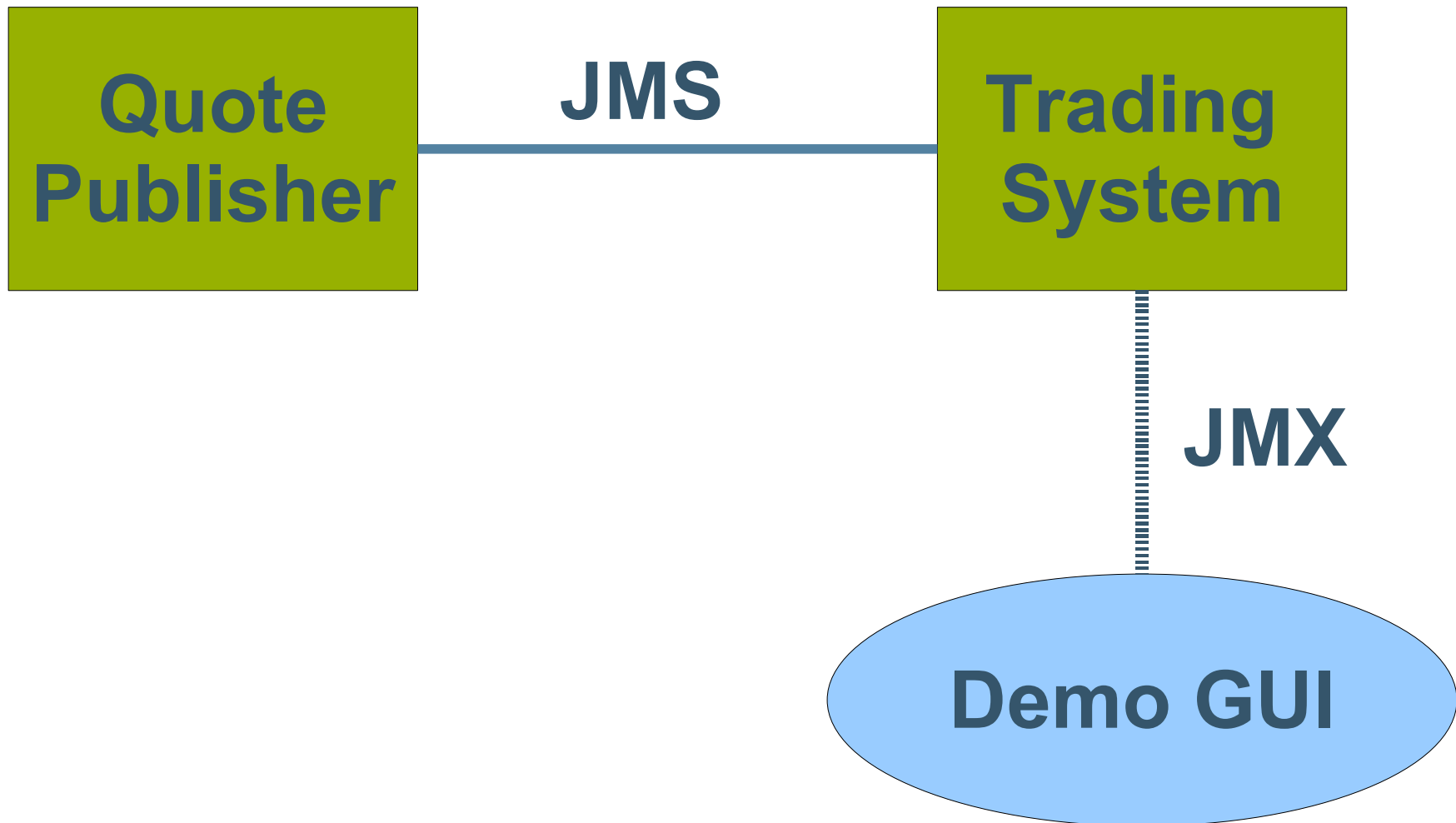


Hard Real-time

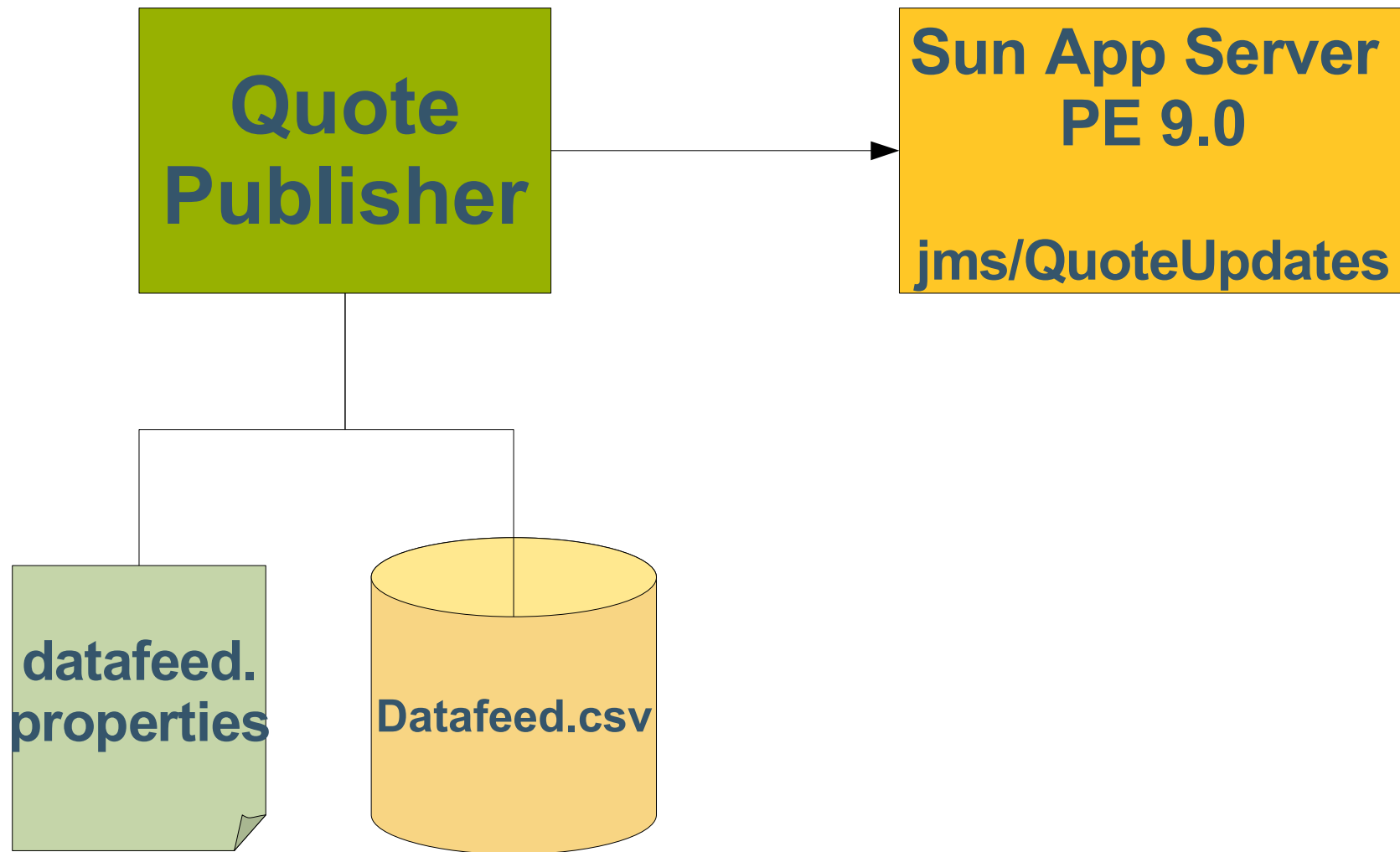
Solaris Real-time Basics

- Fixed priority real-time processes
- High resolution timers
- Pre-emptive scheduling
- Deterministic and guaranteed dispatch latency
- Process priority inheritance
- Memory Management
- Processor Sets
- Interrupt shielding
- Arbitrary Granularity Interval Scheduling

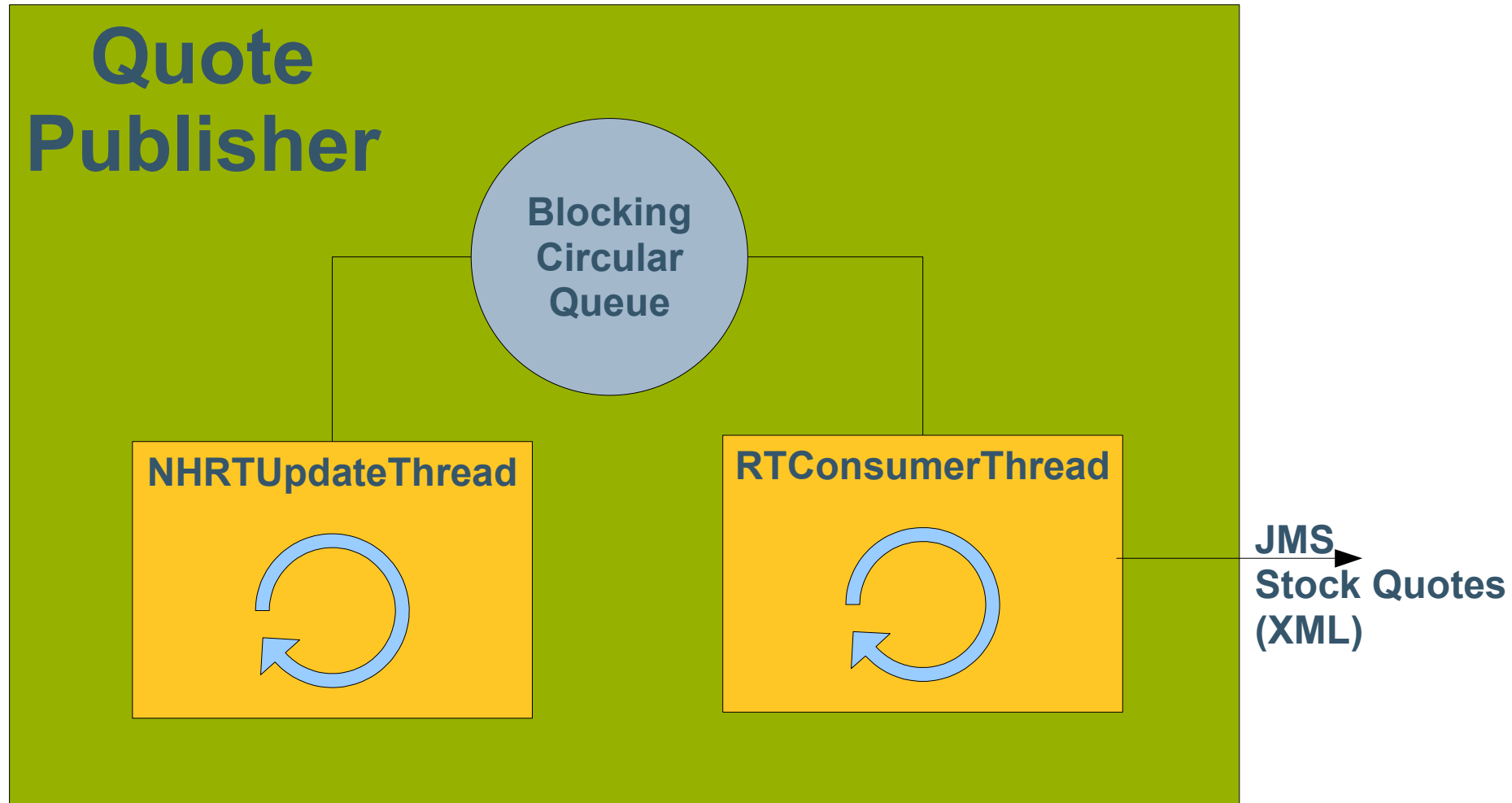
Demo Architecture



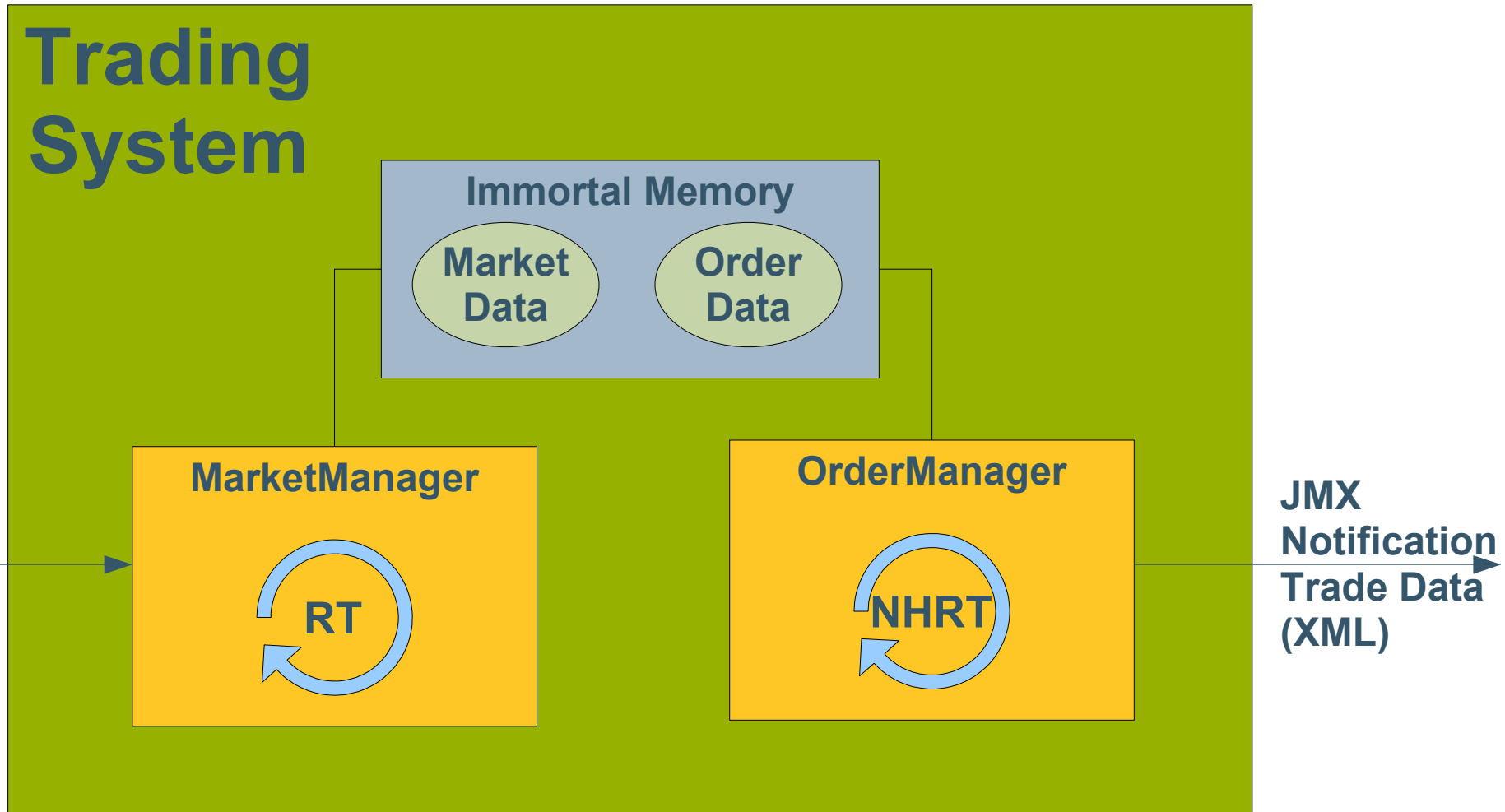
Demo Architecture - Quote Publisher



Demo Architecture - Quote Publisher



Demo Architecture - Trading System



The Financial Demo

- Three main components
 - > Real-time Data Feed (application) **
 - > Real-time Trading Engine (application) **
 - > Performance GUI (application)
- Market data
- Trading data

Demo Results (Non-real-time version)

- The market moves fast and continuous
- 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 continuous
- 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**
This is good!

To the demo...

Java RTS and Solaris – A Summary

- Java RTS is built on Solaris to provide a true real-time application environment
 - > The power of Solaris:
 - > Real-time scheduling
 - > Mission-critical reliability
 - > Proven
 - > The power of Java:
 - > Developer productivity; large knowledge base
 - > No need to build low-level real-time code
 - > Java RTS leverages Solaris' built-in RT facilities so you don't have to

More Java RTS Information

- *Sun's Java RTS Page:*
 - <https://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>
 - https://java.sun.com/developer/technicalArticles/Interviews/Bollella_qa2.html
- *The RTSJ Specification:*
 - http://www.rtsj.org/specjavadoc/book_index.html
- eric.bruno@sun.com
- tim.kay@sun.com



Thank you!

Eric.Bruno@sun.com