Building High Integrity Systems out of Independently Developed Java™ Components of Unknown Pedigree

OMG® Workshop on Real-time, Embedded and Enterprise-Scale Time-Critical Systems

Kelvin Nilsen, Ph.D., Chief Technology Officer Java
Tom Grosman, Senior Real-Time Consultant

The Appeal of Java for Critical System Development

- Developers are typically two times as productive during creation of new functionality compared with C and C++
- Integration activities (porting, reusing, integrating) are typically 5-10 times less costly
  - This is a critical enabler for third party software component industry
  - This preserves investment in software development, allowing in-house accumulation of valuable software IP
    - Software that cannot be economically reused is not “valuable”
    - Dedicated critical system industry sees greater diversity in platforms
  - Empowers new more cost-effective approaches to dedicated system software development:
    - develop and debug independent components on high-end desktop machines
    - deploy integrated system on dedicated embedded target

Examples of Freely Available J2SE Components

Jetty	Web Server w/ Servlets and JSP
OpenJMS	Java™ Message Service
Felix	OSGi Framework and Bundles
Axis2	Web Services with SOAP
JXTA	Peer to Peer Network Protocols
SNMP4J	SNMP Manager and Agent
Hibernate	Persistence and Object-Relational SQL
JacORB	CORBA ORB
Eclipse/SWT	Eclipse IDE and SWT Graphics
Derby	Small Footprint SQL Database

Over 14,000 open source Java projects at sourceforge.net

Sample Critical System: Traffic Signaling

- Signalbau Huber (now SWARCO) uses Real-time Java to control traffic lights
- Local traffic engineers customize behavior for local circumstances by adding Java into the open framework
- Independent software components:
  - Sysgo partitioned PikeOS
  - Perc Ultra VM
  - Swarco-developed safety-critical software prevents conflicting green signals
  - Customer-supplied Java code optimizes traffic flows

Sample Critical System: Financial Services Platform

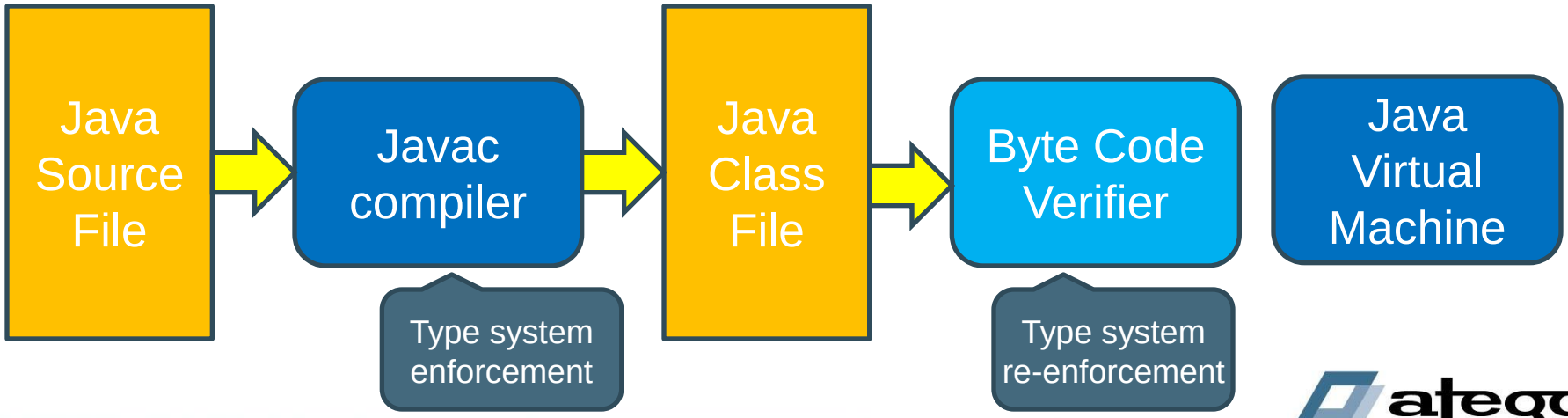
- Financial transaction processing
  - Standard services provided in a secure environment
  - Customizable with customer's own code
  - Tight control over access to system resources
  - Severe memory constraints
- Independent software components:
  - Real-time operating system
  - Perc Ultra VM
  - Basic functionality (C++) delivered in-house
  - Customer code (Java) added ***after*** delivery of the system

Sample Critical System: Aegis Warship Software Modernization

- Aegis warship weapons control software
  - Includes ballistic missile defense
  - 500,000+ lines of code
- Independent software components:
  - Red Hawk Linux
  - GoAhead's SelfReliant high availability framework
  - Perc Ultra VM
  - RTI DDS middleware
  - Multiple independently managed projects developed by Lockheed Martin and their subcontractors

Java Language Strengths

- Stronger type system than C and C++
  - Locals must be initialized before first use
  - Fields are initialized to zero or null
  - Only safe coercions are allowed
  - Forced consistency between actual and formal arguments
  - Strong encapsulation of data abstractions (privates are private)



Java Security Model

- Byte-code verifier prevents malicious or accidental violation of strong type system
- All array subscripts checked
- Stack overflow checked
- No dangling pointers because no object deallocation
- Reduced likelihood of memory leaks (garbage collection)
- Extensive cryptography services (assure integrity of class files and data)
- Security manager enforces “virtual sand box”
  - Every security sensitive operation performed by an “untrusted” application must be approved by the security manager

Benefits of Object Orientation

- Object orientation encourages reuse and extend rather than discard and rewrite
  - Java portability facilitates software reuse across different platforms
  - “Experienced” software is less likely to have bugs than newly authored software
  - Abstraction and generic types allow a single algorithm implementation to be safely reused in many different contexts
    - Only one implementation of the algorithm to be reviewed, maintained
  - Portability and popularity of Java have resulted in abundance of reusable COTS and FOSS components
  - Reuse with refinement is implemented by inheritance, without requiring any changes to original (field proven) capabilities

Benefits of Object Orientation

Example of software reuse with refinement:

```
public class EmergencyStopHandler extends BoundAsyncEventHandler {
    public void handleAsyncEvent() {
        stopMotor();
        soundAlarm();
        coordinateWithLineManager();
    }
}

// inherits implementations of addIfFeasible(), addToFeasibility(),
// getAndClearPendingFireCount(), getAndDecrementPendingFireCount(),
// getAndIncrementPendingFireCount(), getMemoryArea(),
// getMemoryParameters(), getPendingFireCount(),
// getProcessingGroupParameters(), getReleaseParameters(),
// getScheduler(), getSchedulingParameters(), isDaemon(),
// removeFromFeasibility(), run(), setDaemon(), setIfFeasible(),
// setMemoryParameters(), …
```

Risks of Integrating Independent Software Components

- Denial of service
  - Memory allocation pool is shared: misbehaving thread may prevent a critical thread from timely allocation
  - Pool of processing cores is shared: misbehaving thread may enter an infinite loop at a high priority
  - Critical synchronization locks may not be appropriately hidden: misbehaving thread may acquire and retain a critical lock
- Compromised information hiding
  - Programmers may fail to restrict visibility of fields and methods that should be “hidden”
  - Reflection services allow applications to subvert traditional access restrictions (e.g. to see hidden fields and methods)
- Native code
  - Generally ignores traditional Java security model and may even crash the Java virtual machine

Use of java.lang.SecurityManager

- Many of Java's standard libraries "consult" the system-wide security manager before acting on sockets, threads, class loading, files, properties, networking, and windowing
- A SecurityManager decides whether to approve particular operations depending on which thread is running, the current call stack for that thread, and the operation's proposed arguments
- If the SecurityManager objects to a proposed operation, it throws a SecurityException
- Restrict the privileges of 3rd party software components by implementing your own SecurityManager which extends java.lang.SecurityManager

Static Analysis Tools

- A static analyzer derives information about software behavior and structure by examining Java source or class files
  - Enforce compliance with particular programming style guidelines
    - Checkstyle (open source)
  - Understand control and data flows and class relationships
    - Klockwork Insight Pro (Architect)
    - Scitools Understand
  - Aid in the search for common programming errors
    - Checkthread (checkthread.org)
    - Findbugs (findbugs.sourceforge.net)
    - Jlint (sourceforge.net/projects/jlint)
    - Klockwork Solo
    - Parasoft Java Quality Solution
    - Coverity Prevent for Java
    - SofCheck Inspector for Java
    - PMD (pmd.sourceforge.net)

Static Analysis Tools

- A static analyzer derives information about software behavior and structure by examining Java source or class files

Strong type checking of Java enables static checkers to perform more rigorous analysis than with C and C++

Popularity of Java has resulted in a variety of static analysis tools

Each tool delivers different benefits – many organizations apply combinations of these distinct tools

Recommended applications:

1. Facilitate an understanding of imported software components
2. Enlighten as to flows of information within software of “unknown pedigree”
3. Enforce proper use of ‘private’ and ‘package’ restrictions

- PMD (pmd.sourceforge.net)

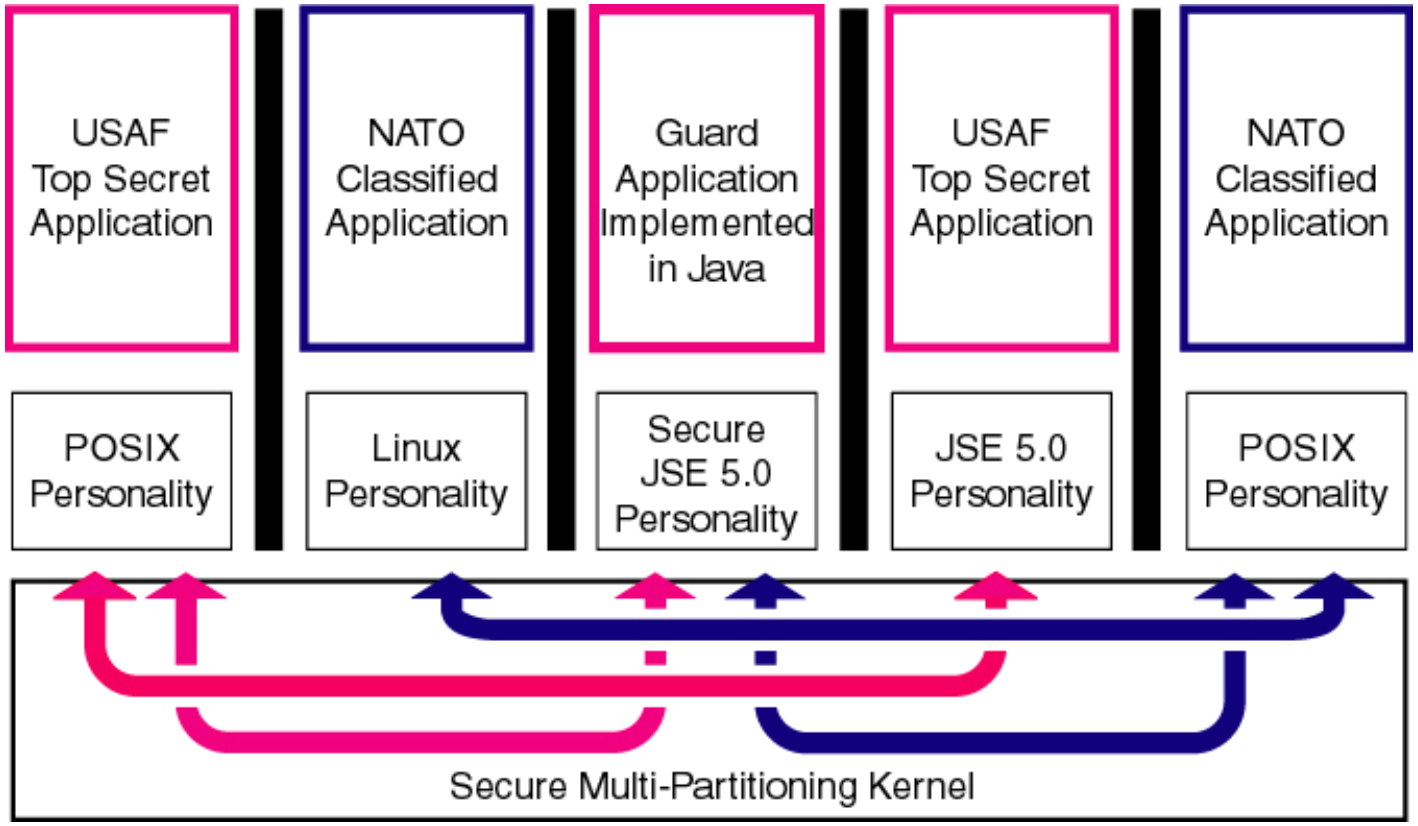
Partitioned Execution

- When mixing Java software components with different assurance (trust) levels, there are risks that less assured software will compromise higher assurance software
  - Run an infinite loop at a very high priority
  - Allocate huge amounts of memory in an infinite loop
  - Acquire and retain a mutual exclusion lock on a critical shared resource

- Separation of concerns is a common approach to managing system complexity
  - Run multiple JVM instances in virtually isolated partitions
  - Each instance of a JVM has its own heap memory budget
  - Each instance of a real-time JVM can map its Java priorities to a different range of operating system priorities

Partitioning Options

- Use RTOS processes or a “secure partitioning kernels” (MILS, ARINC 653) to enforce isolation of both time and memory



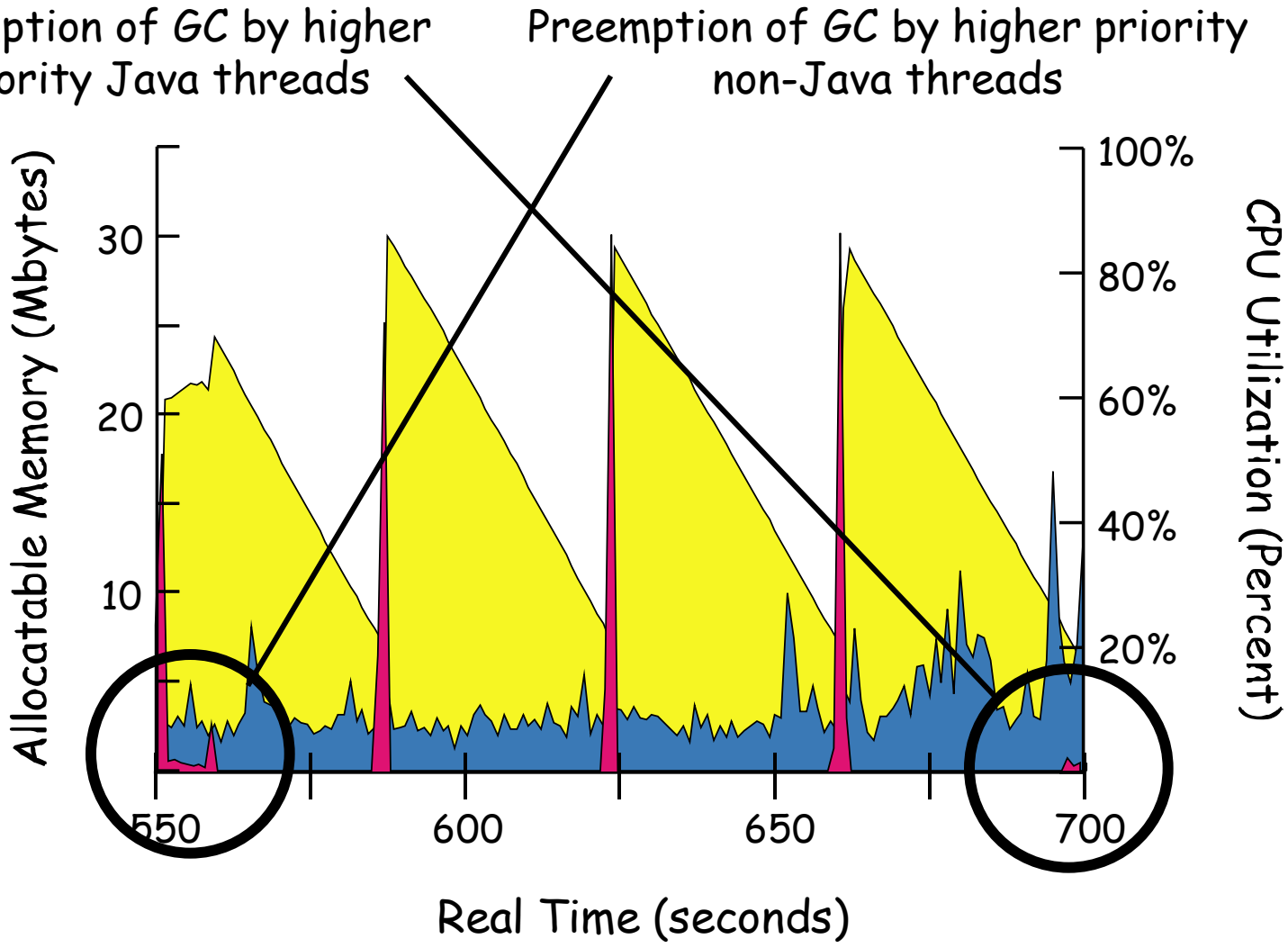
Real-Time Virtual Machines

- Many mission critical high assurance systems have real-time constraints
- Activities that occur at the wrong time have reduced or even negative value
- Developers of real-time software perform theoretical schedulability analysis of workload to prove compliance with deadlines
- The virtual machine needs to behave in predictable ways to support schedulability analysis
  - Traditional Java Virtual Machines fail this requirement due to JIT compiler interactions, garbage collection pauses, unpredictable priority boosting, and unbounded priority inversions

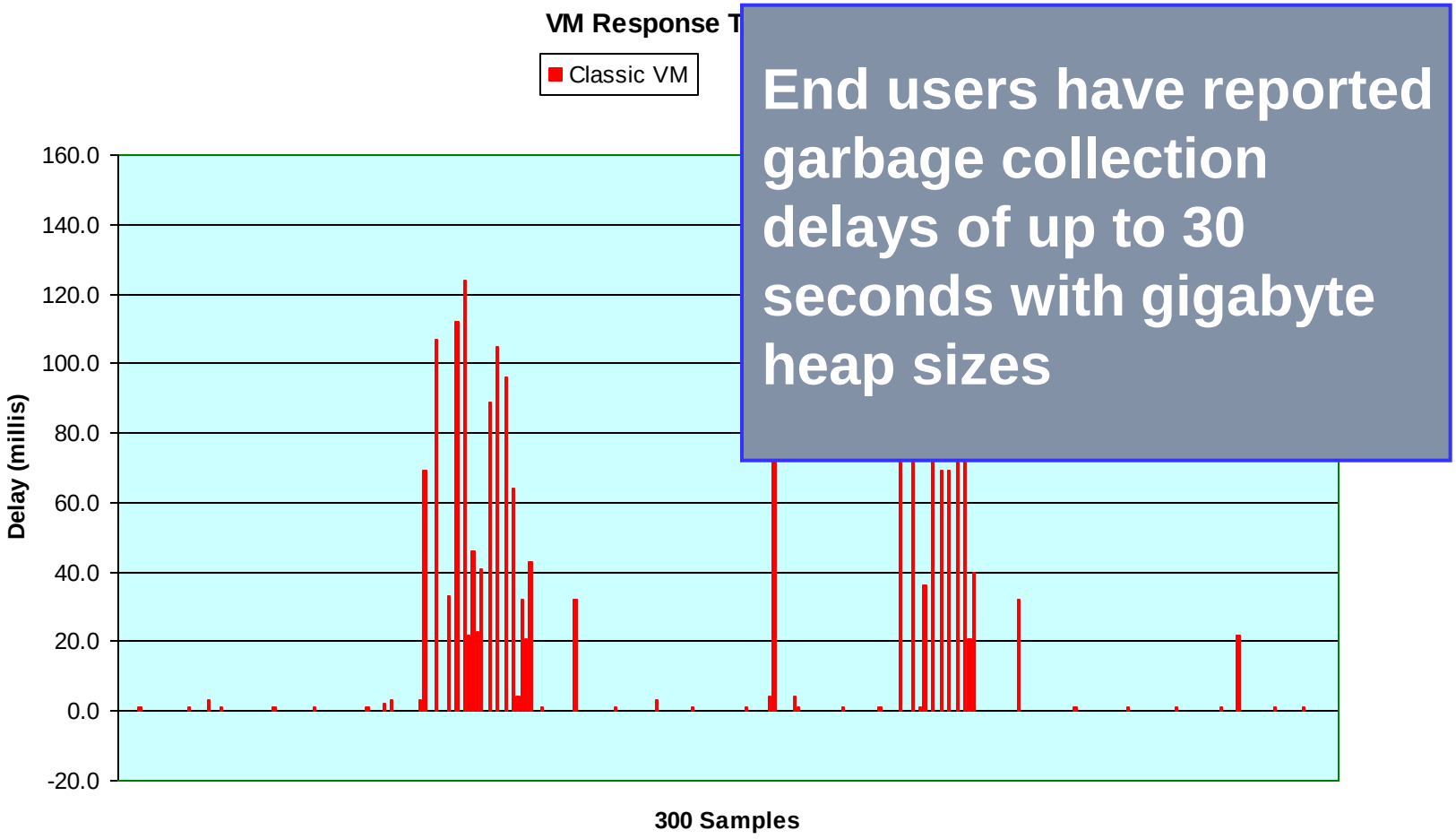
How To Do Soft Real-Time with J2SE APIs

- Real-Time Garbage Collection
  - Preemptive
  - Incremental
  - Fully accurate
  - Defragmenting
  - Paced
- Fixed priority scheduling
  - With extended priority range explicitly mapped to OS priorities
- Priority inheritance synchronization with all thread queues maintained in priority order
- VM management services
  - To calibrate and monitor resource consumption and adjust resource budgets

Trace of Garbage Collection Pacing

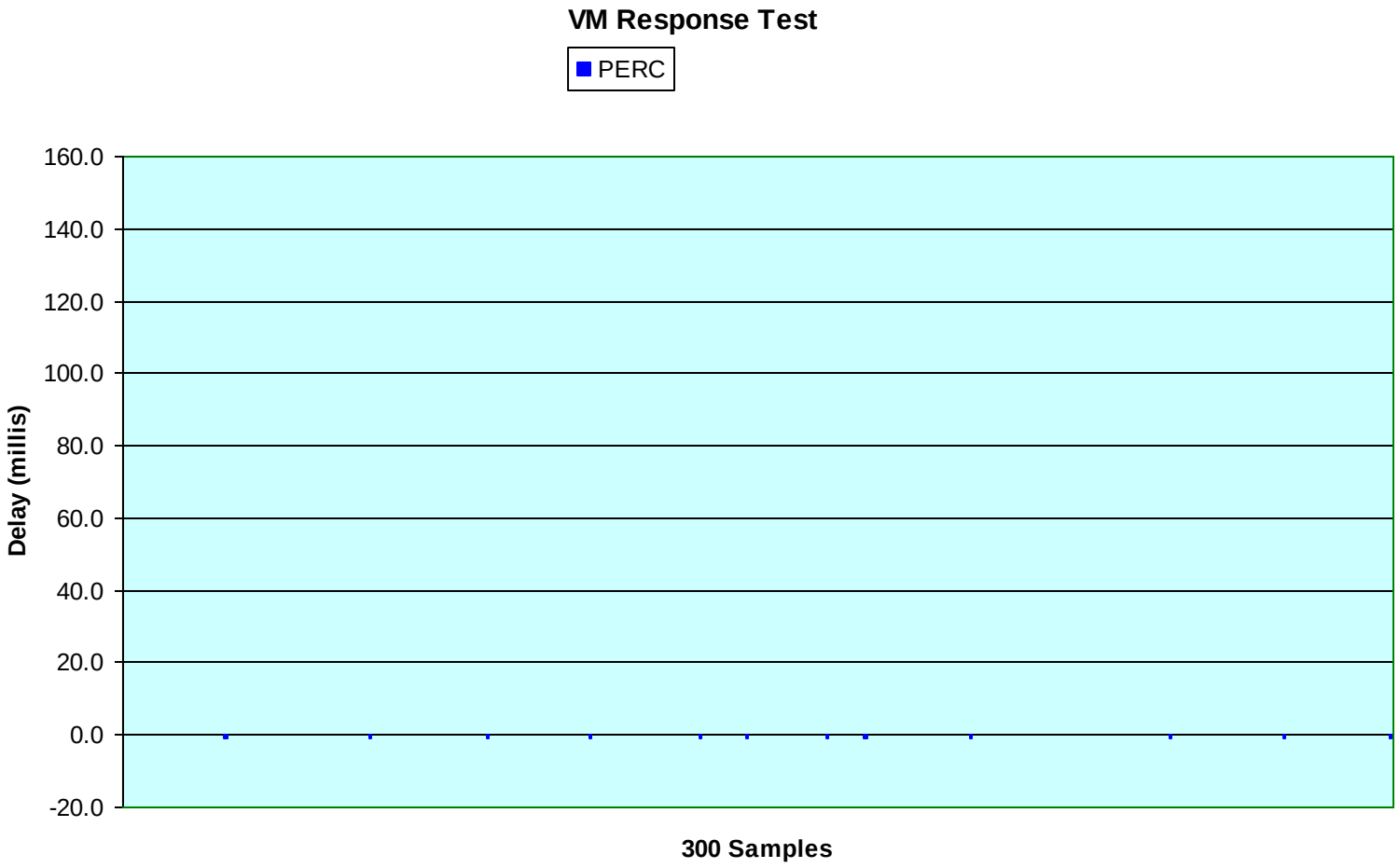


Classic VM running VM Response Test



End users have reported garbage collection delays of up to 30 seconds with gigabyte heap sizes

Real-Time VM running VM Response Test



Integration with Legacy (non-Java) Software Components

- Most modern software systems include a combination of new functionality combined with decades of accumulated legacy capabilities
- JNI (Java Native Interface) allows C, C++, and Ada code to be linked into a JVM to enable calls to and from Java
  - But be very careful: high run-time overhead and circumvents the Java security model
  - JNI native methods supplied by 3rd parties deserves extra scrutiny
- When feasible, use virtual partitions to separate legacy and Java code, communicating by CORBA, NDDS, network sockets, or shared files
- More efficient integration may be realized by use of hard real-time Java “gateway” between legacy and traditional Java

Summary

- Java provides many benefits to the developer of high assurance software:
  - Strong type enforcement, security model, object-oriented software reuse, static analysis tools
- Developers must use Java effectively to derive the greatest potential benefit
  - Follow object-oriented design, apply static analysis tools, partition components with different assurance levels
- Special Java technologies are required for real-time systems