Understanding the Role of Real-Time Java in Aegis Warship Modernization

Dr. Kelvin Nilsen, Chief Technology Officer Java, Atego Systems

Aegis Weapons System Overview

- The shield of the fleet, providing area air defense for a “carrier battlegroup”, as well as long-range ballistic missile defense
- Computers monitor data from Radars, provide situational awareness to human operators, and automate launching of interceptor missiles
- About 100 computers, many multi-core, providing end-to-end timing guarantees of less than 100 ms on computations spanning many computers
- Jitter constraints on “local” computations are well under one ms

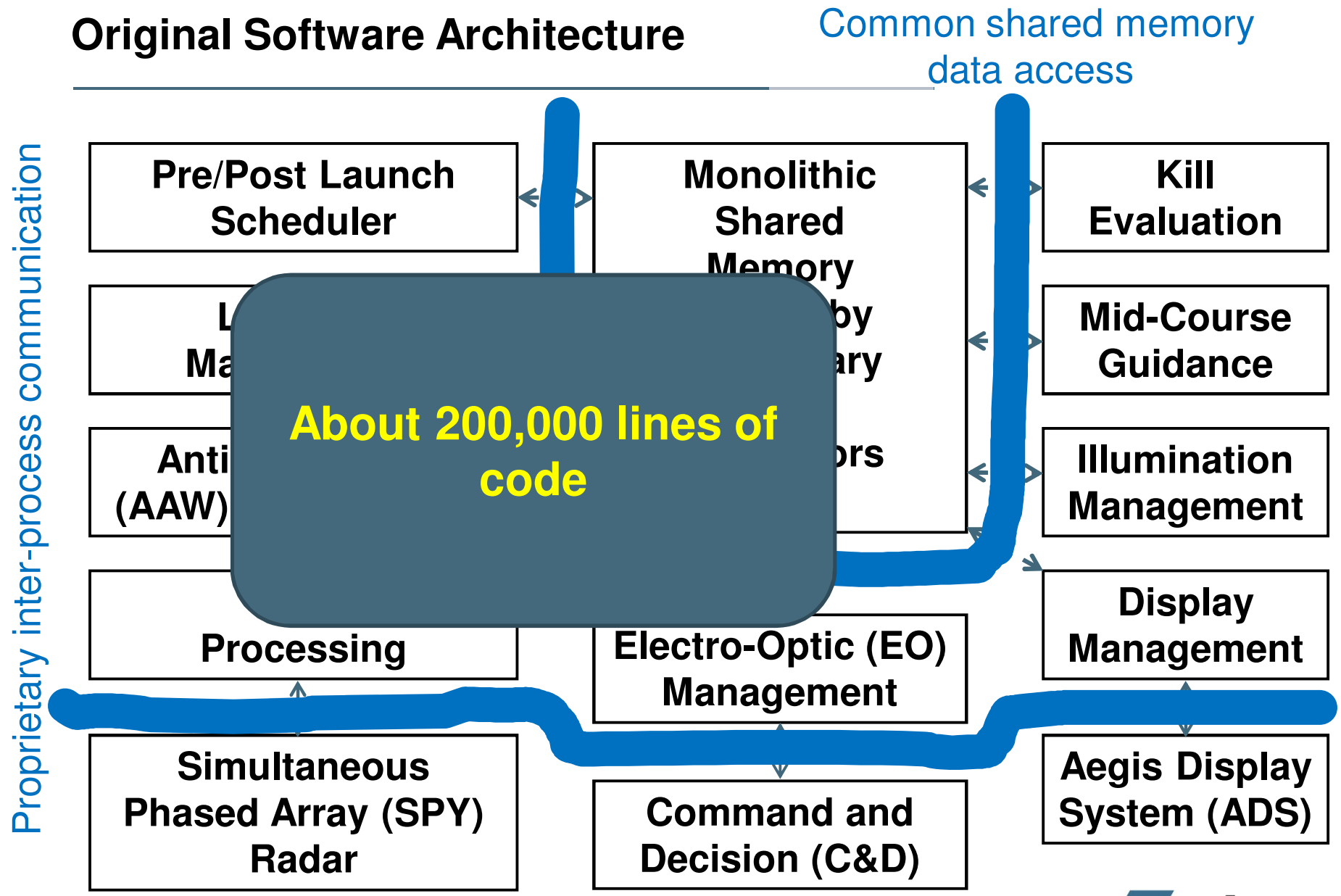
Why "Modernize" the Aegis Weapons Software?

- Original Aegis Weapons software ran on a mix of government-specified UYK computers and some other systems, running custom software mostly written in CMS-2
- The reliance on custom technology limited access to competition: could not leverage economies of scale
- Step one: replace some UYK computers with COTS processors
  - Replaced some CMS-2 code with Ada, C++, and Jovial applications
- But porting software to each new COTS hardware configuration required considerable time and effort
  - Aegis "Baseline 7" selected hardware in 1998, but first (of many planned) deployment(s) was delayed until 2005
  - Cannot afford the calendar time, nor the costs associated with continual porting of the software to new platforms

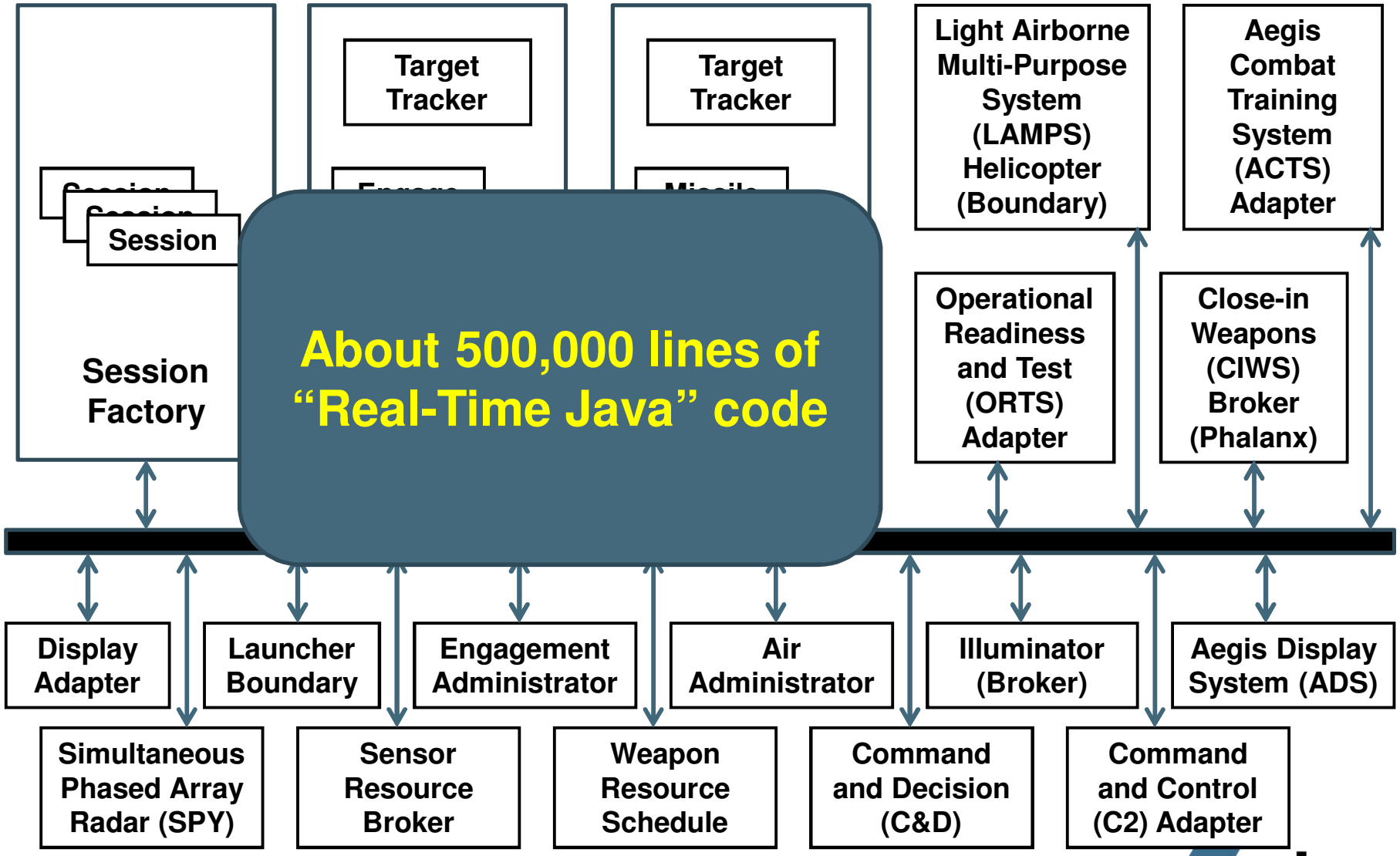
Key Objectives of Aegis Weapons System Modernization

- To decouple hardware from software, enabling all deployed Aegis Combat Systems to undergo hardware upgrades every 4 years and software upgrades every 2 years, on a staggered schedule (over 100 ships, plus new land-based deployments)
- To enable competitive bidding for new software capabilities, with new software components easily integrated into the existing software infrastructure without compromising existing software capabilities

Original Software Architecture



Modernized Software Architecture



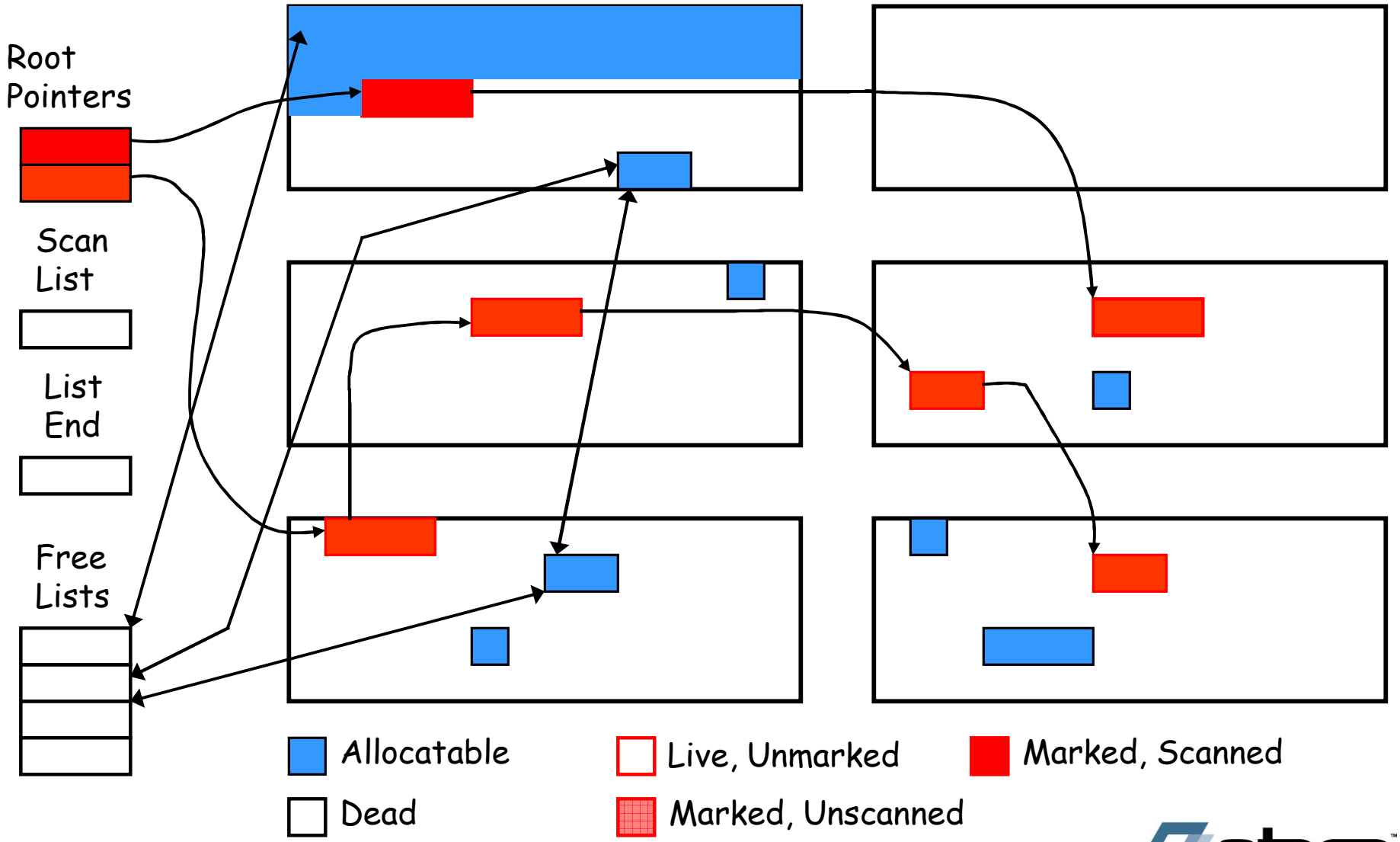
What do we mean by “Real-Time Java”?

- It is not the “Real-Time Specification for Java” (RTSJ)
- Real-Time Garbage Collection
  - Typical applications enforce time constraints of 1-100 ms
- Portable Real-Time Scheduling and Synchronization
  - Global dispatching always runs the N highest priority ready Java threads on N available cores
  - Implements priority inheritance on all Java synchronization locks
  - Maintains all thread queues in priority order
  - Optional use of extended priority range (1-32)
- Embedded integrations (RTOS, processors, ROM deployment)
- VM Management Services (monitor, control resource utilization)
- Improved Timing Services
  - Monotonically increasing clock for global synchronization

Aspects of Real-Time Garbage Collection

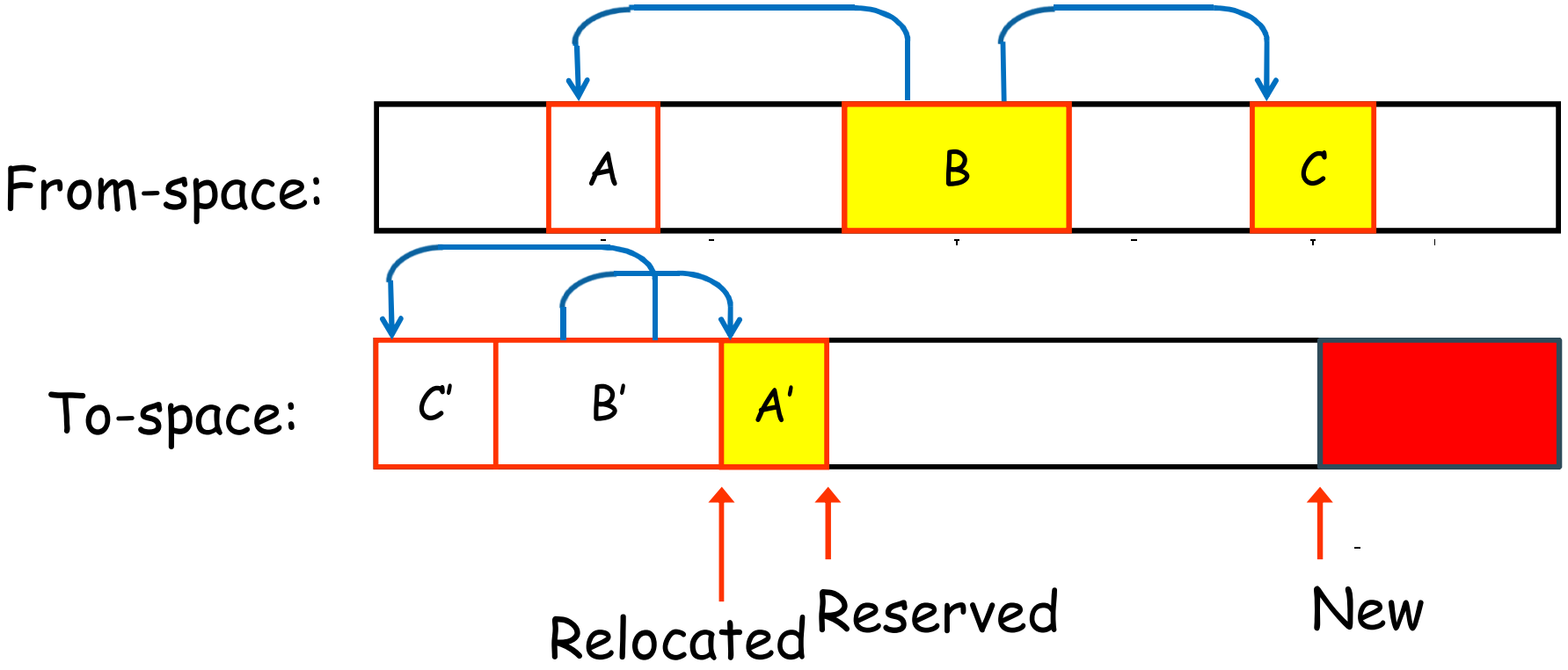
- Key attributes of Perc Ultra real-time GC for SMP Java:
  - Preemptive
  - Incremental
  - Accurate
  - Defragmenting
  - Paced
  - Parallel and Concurrent
- Garbage collection is a service provided on behalf of application
  - Generally runs at “low priority” due to long deadlines
  - May inherit priority of any high-priority task that needs to allocate
- Threads that need to run at higher priority than garbage collection need to avoid allocating memory (in order to avoid priority inversion)

Incremental Mark and Sweep GC



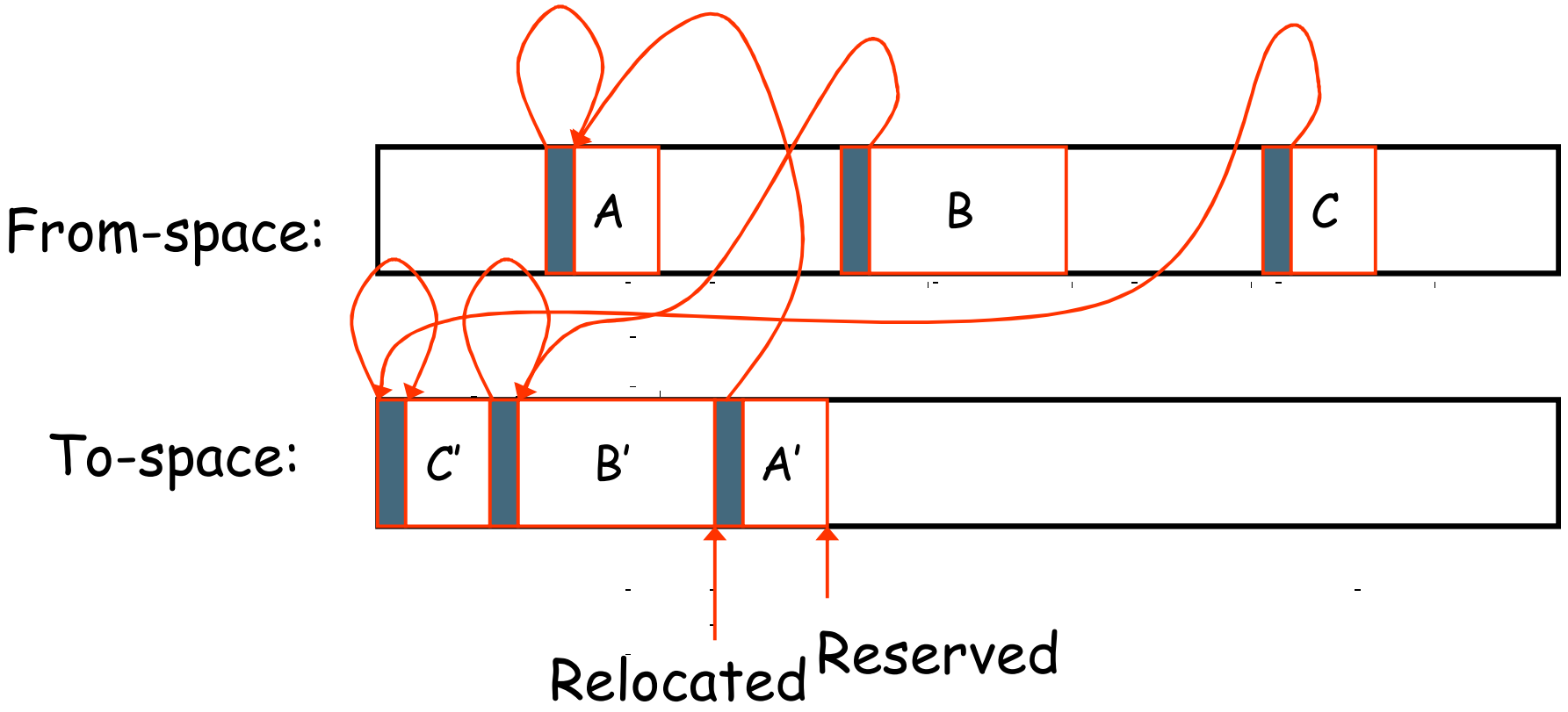
Fully Copying Garbage Collection

- Incrementally copy objects from from-space into to-space
- Redirect memory accesses between Relocated and Reserved

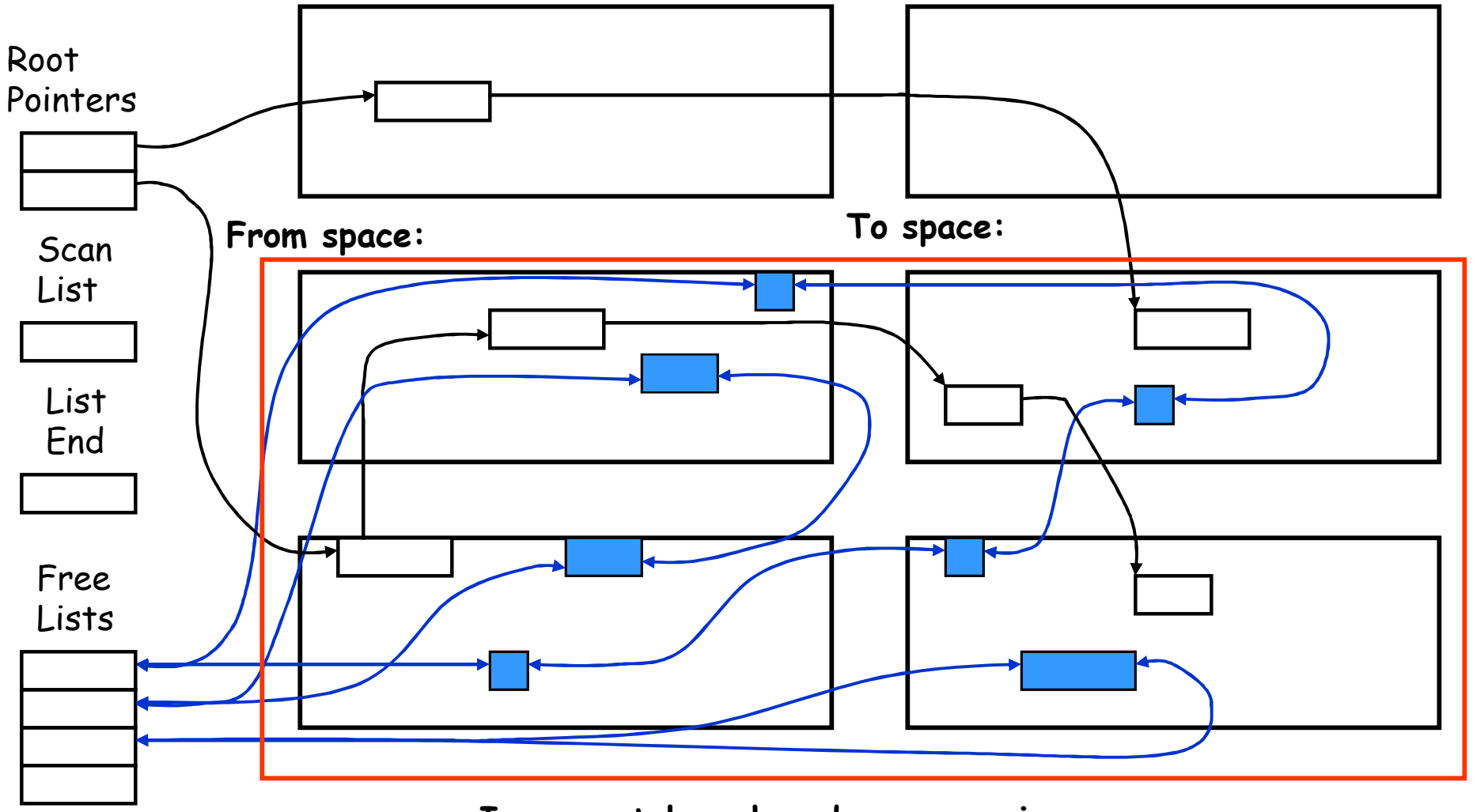


Implementation Approach

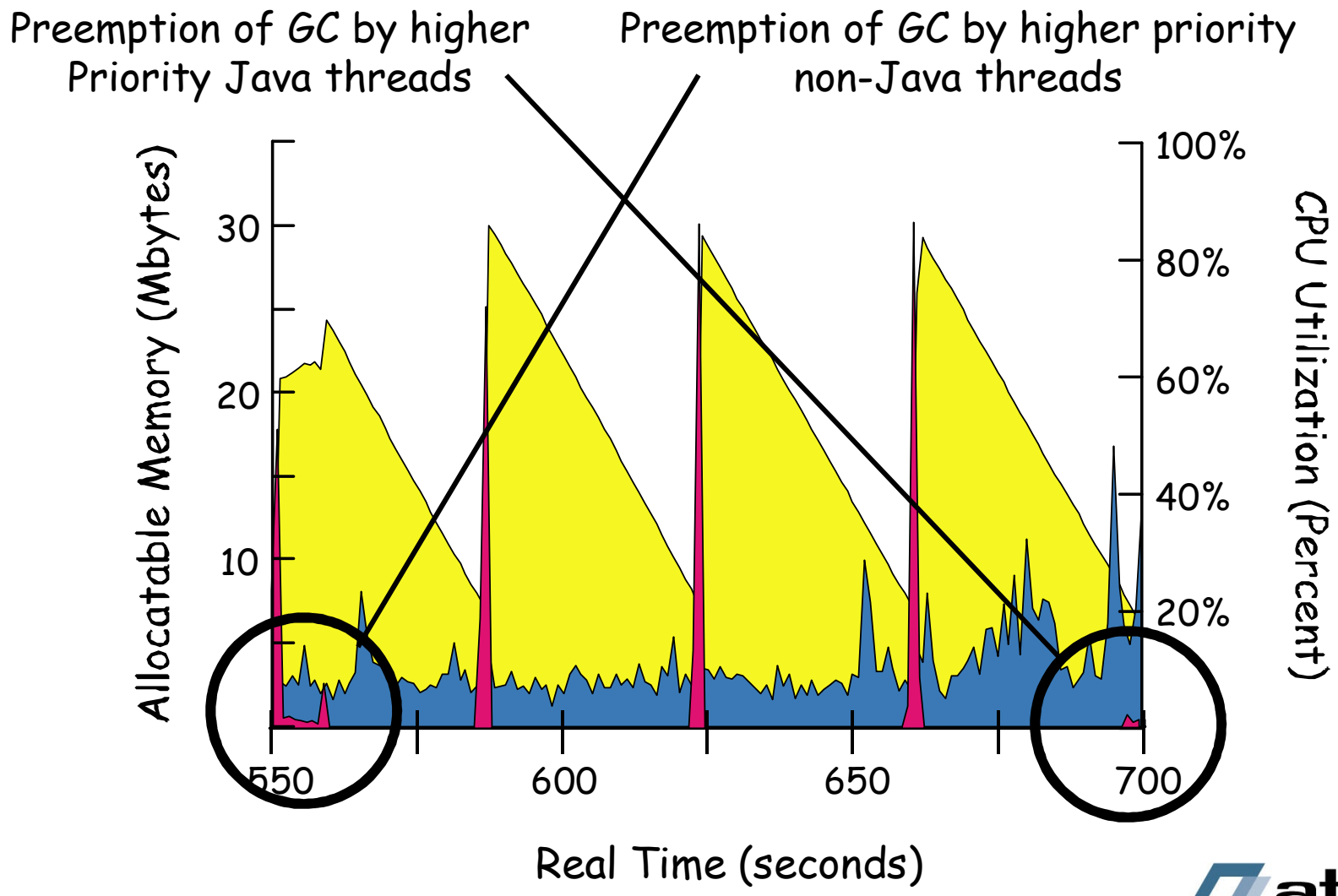
- Every object has a valid-copy pointer contained within its header



Mostly Stationary Garbage Collection



Pacing of Garbage Collection



Project Results

- **"I can say it was a roaring success for us."** – Andrew Winkler, Chief Software Architect, Lockheed Martin Aegis Open Architecture
- Existing software engineers (previously proficient in Ada or CMS-2) quickly adapted to Java
- The Eclipse Integrated Development Environment, debugging support, and COTS Java tools represented huge improvements
- Implemented Aegis Air Defense Warfare software (150K SLOC), including test, evaluation, and requirements verification, in 18 months – **"Phenomenal"**
- Verified 3,500 requirements to Navy satisfaction in 5 months **(9-fold improvement over previous best practice)**

Project Results (part 2)

- First deployment based on new Open Aegis Architecture required only 3 years from hardware selection (vs. 7 years)
- Support for a traditional deployment style (AoT compile and link) satisfied Navy export restrictions (Aegis is shared with many allies)
- After “completing” the Open Aegis Architecture modernization, support for “Standard Missile 6” was added in only 3 months (vs. 1 full year previously)

Conclusions

- "Real-time Java" can provide important value to large and complex dynamic soft real-time systems
- Soft real-time systems require empirical techniques within a theoretically sound analysis and implementation architecture
- Object-oriented encapsulation and modular composition are critical to the success of this effort
- Portability across diverse and evolving hardware and operating system platforms was perhaps even more critical to the success of this project
- Use of COTS software technologies can leverage high-volume economies of scale to the benefit of mission-critical and even safety-critical systems

Acronyms

- AoT: Ahead-of-Time
- CMS-2: Compiler Monitor System 2 (a programming language)
- COTS: commercial off-the-shelf
- CPU: Central Processor Unit
- GC: Garbage Collection
- J2SE: Java 2 Standard Edition
- ms: millisecond
- ROM: Read-Only Memory
- RTOS: Real-Time Operating System
- RTSJ: Real-Time Specification for Java
- SLOC: Significant Lines of Code
- SMP: Symmetric Multi-Processor
- VM: (Java) Virtual Machine