A New kind of real-time: Enterprise Real-time

Theodore Ts'o
IBM Linux Technology Center
January 16, 2007

Agenda

- Introduction to real-time and Enterprise real-time
- IBM's Real-Time Linux Extensions
- The CONFIG_PREEMPT_RT patch
- Real-time Java(tm)
- Conclusion

Definitions

- Real-Time Operating Systems (RTOS) are systems which provide guarantees about the time between when an event happens and code which is waiting for that event starts executing.
  - This time is often referred to as “latency”
  - Different types of latency
    - Scheduler latency
    - Interrupt latency
- Hard versus Soft Real-time
  - Hard Real-Time systems have maximum latencies that can be proven (via longest code path analysis)
  - Soft Real-Time: “best effort” latencies

Traditional Real-Time

- Generally used in small embedded systems
  - Uniprocessor (SMP systems too complex for max. code path analysis anyway)
  - Restricted CPU speeds
- Often used for data collection and immediate “real-time” analysis
- Everyone wants “hard real-time”
  - “Soft realtime” has a (unfair) connotation of “not real-time at all”
  - What does “best efforts” mean, anyway?
- TCP/IP is generally too complex for max. code path analysis, so traditional RTOS's generally didn't have TCP/IP support at all, or did TCP networking outside of the real-time subsystem

The Times They Are Changing....

- Computers have gotten much faster
  - The Cray-1 (1976) had 160MFLOPs and 8MB of memory
    - A modest embedded system, by today's standards
    - Not even capable of running Mozilla or OpenOffice!
  - Now possible to execute many more instructions in 10us.
- Expectations are now much higher than before. TCP/IP is a must. (Some customers require real-time CORBA.)
- Multi-core processors means that even “small” embedded machines will be SMP
- Changing requirements in enterprise software: high throughput not enough; need latency guarantees as well!
  - Complex multi-tier architectures cause latencies to be additive

The result? A new kind of real-time

- Let's call it “Enterprise real-time” to distinguish it from traditional real-time applications
- Characteristics
  - Supports large SMP systems
  - TCP/IP a requirement
  - Ability to support commercially available middleware products
    - Databases, Web servers
    - Perhaps not with real-time characteristics...
  - Supports higher level programming language
  - May not have “hard real-time” proof of maximum non-determinism

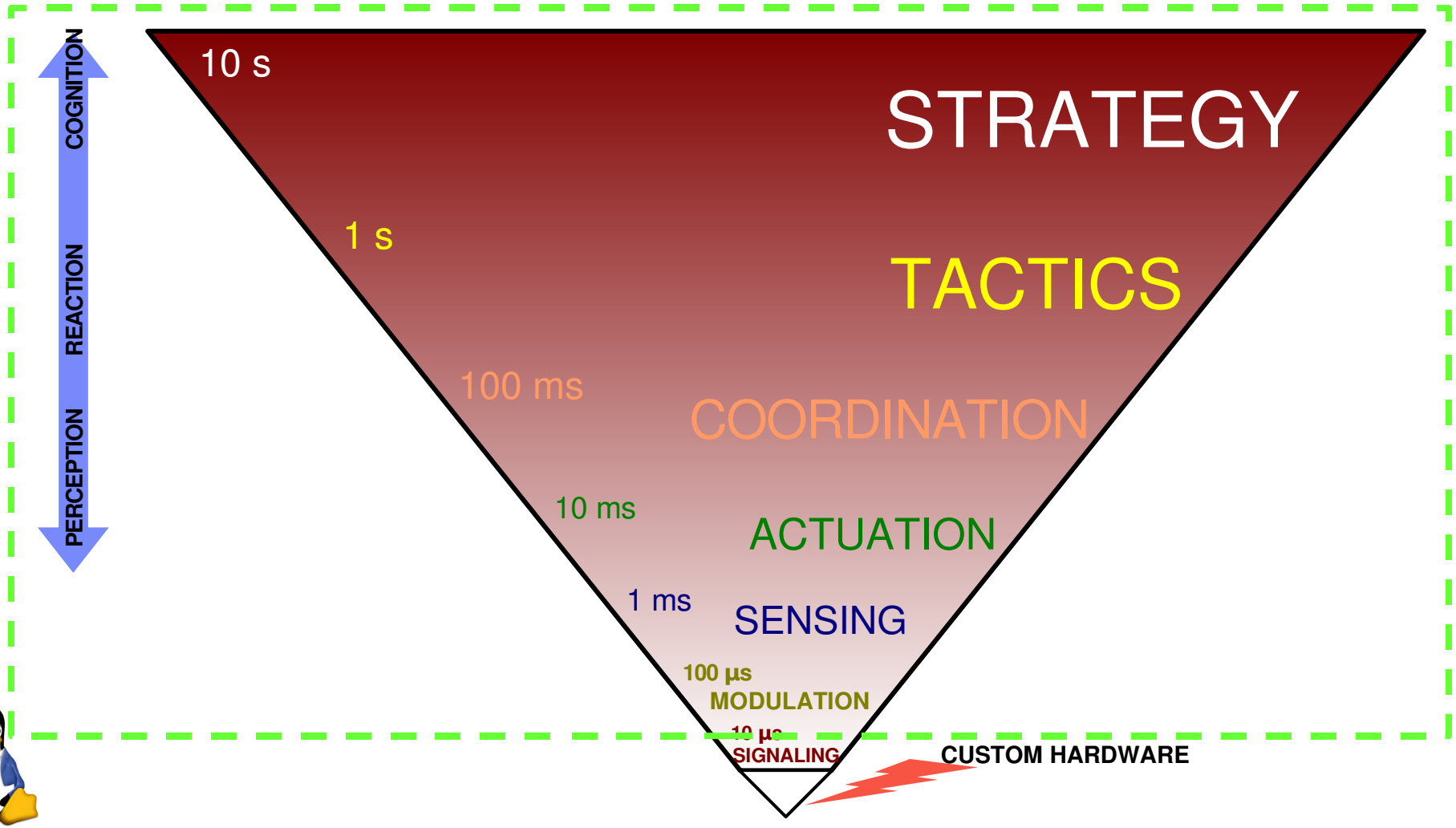
Of code path analysis and mathematical proofs...

- Modern systems are so complex that mathematical proofs of maximum latency are impractical.
  - (This has been true for proofs of security as well)
  - Replaced with empirical testing
  - Hardware failures can cause latency misses, so a statistical "proof" isn't so bad --- after all, MTBF is a statistical concept!
- Some could argue this is just a fancy way of saying "soft real-time"....
  - But 10-20us latencies while running 10 kernel compiles, several disk-to-disk transfers, and while the system is being ping flooded is a lot better than what most people associate with "soft real-time"!

A Word About Latency Numbers....

- Real-time truism: ask a real-time programmer what latency number is required, and they will always tell you “as small as possible” (or they will give you a numerical equivalent to that).
  - This may be because even for hard real-time systems, application programmers mistrust the “guarantee” and so they add engineering padding to their requirements
- Depending on the application, it may make sense to talk about real-time systems with a 1 second latency.
  - There is more to the real-time application space than just data acquisition!

Latency Time Scales



IBM's Real-Time Linux Extensions

- Provide bleeding-edge latest development code to early adopters in form that has been stabilized and supported by IBM.
  - Standard kernels from Red Hat or SuSE are of necessity 9 to 24 months older than the kernel.org development stream
  - Allows IBM clients to gain a first-mover advantage over their competitors
- Restricted set of hardware platforms which are supported
- Utilizes a RHEL4U2 for its user space environment
  - Only two interfaces added to glibc; no other changes
  - Allows most 3rd party software products to work without modification, so long as they do not depend on binary/proprietary kernel modules.

Contents of the Real-Time Linux Extensions

- Patches and scripts to update a base 32-bit x86 RHEL4U2 system
  - 2.6.16 Linux kernel
  - 2.6.16-rt22 CONFIG_PREEMPT_RT patch from Ingo Molnar
  - 2 new interfaces added to glibc to support priority inheritance
  - enhancements to the PAM libraries to allow non-root users access to real-time facilities
  - direct physical memory access required for RTSJ conformance
  - bug fixes and stabilization patches
- Support provided via IGS Supportline and the IBM Linux Technology Center

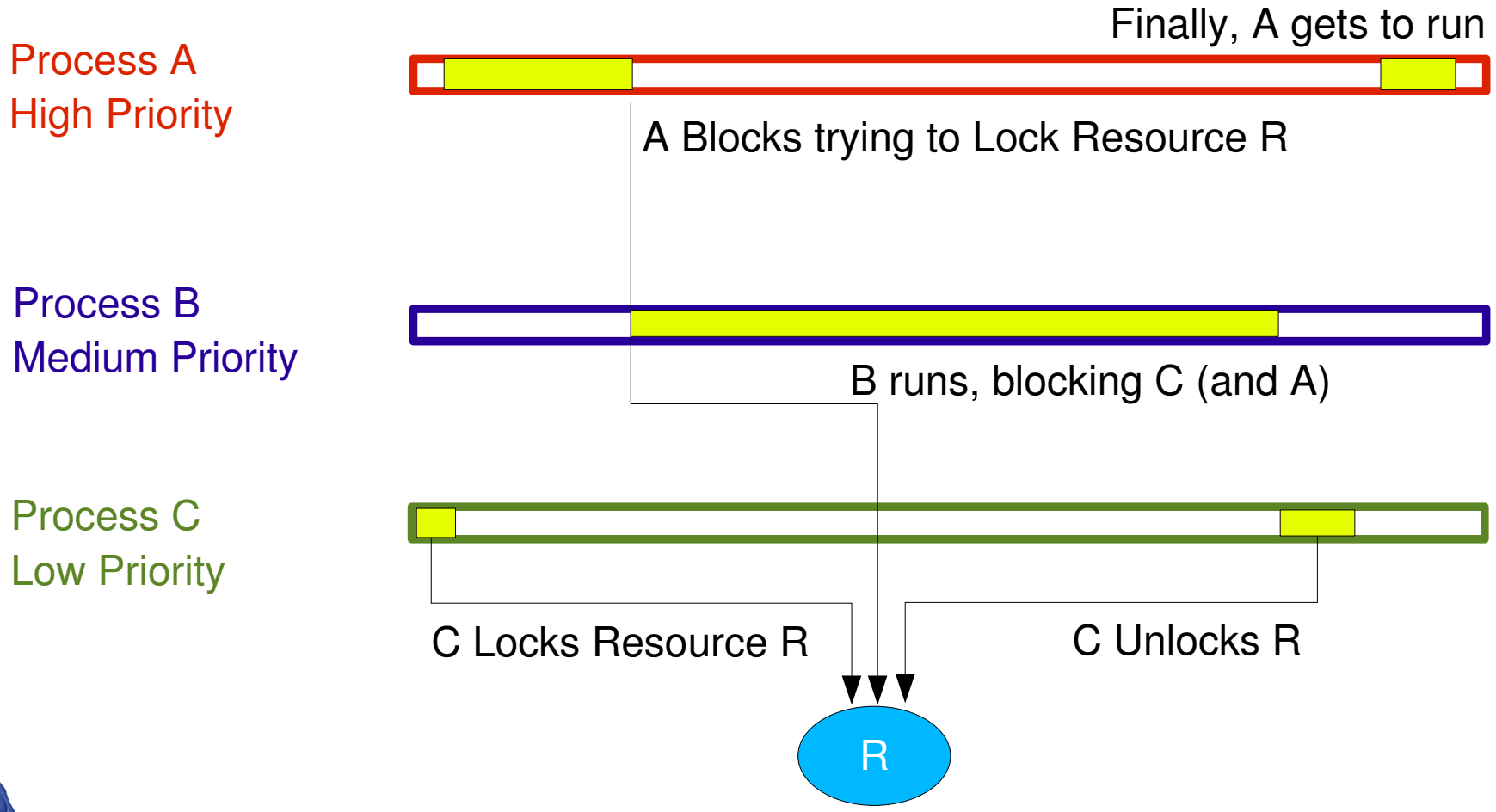
Technical Features of CONFIG_PREEMPT_RT

- High-resolution timers
- Improved kernel preemption functionality
- Full kernel and userspace priority inheritance
- Hardware and software interrupt handlers run in kernel threads

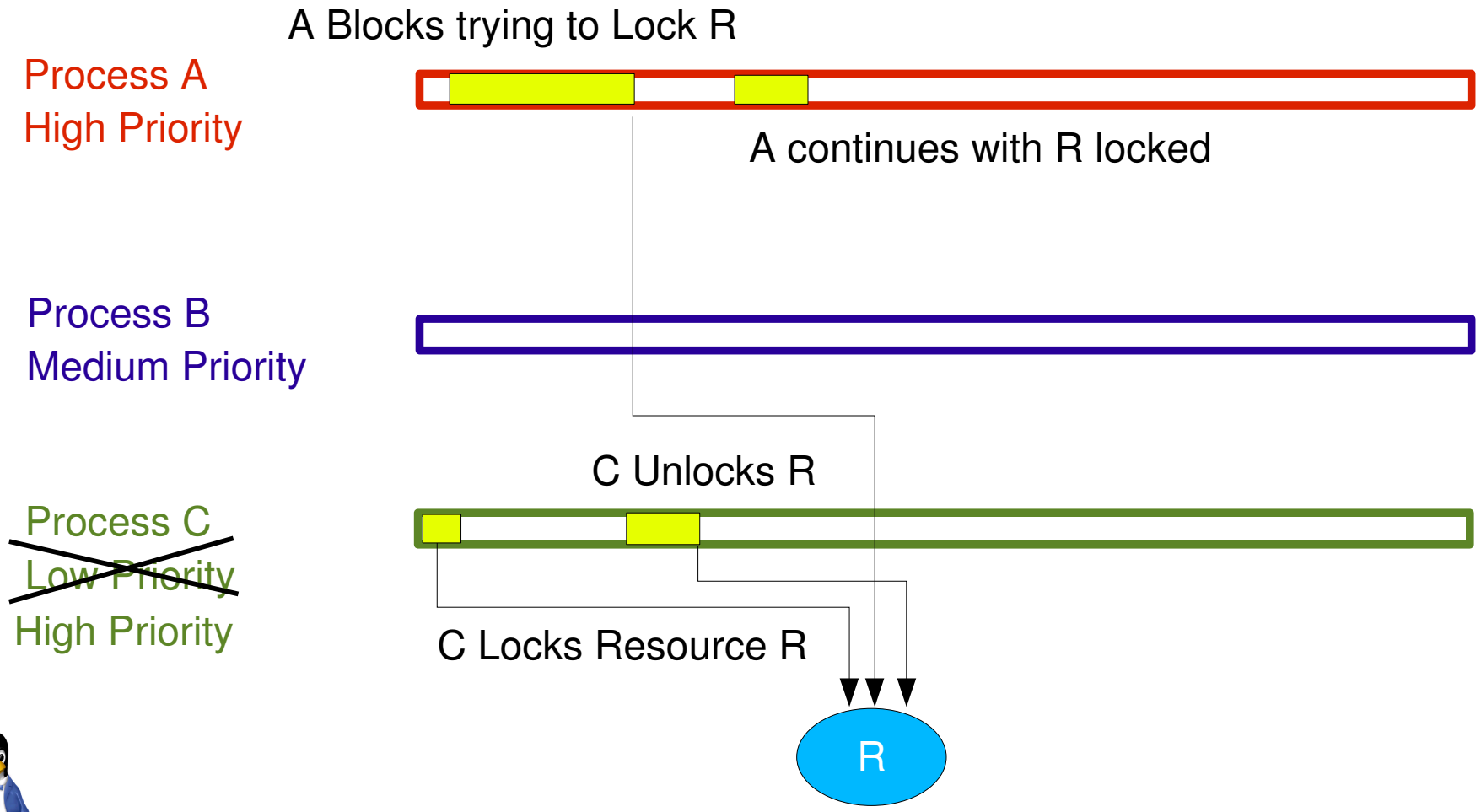
Kernel Preemption Changes

- Normal Kernel
  - Top-half kernel code only gives up control when:
    - schedule() is called
    - reference to user memory --> page fault
- CONFIG_PREEMPT
  - Kernel code can be preempted by higher priority process whenever not in a spinlock critical section
- CONFIG_PREEMPT_RT
  - Spinlocks become mutexes with priority inheritance to prevent priority inversion problems

Priority Inversion



Priority Inheritance



Hardware/Software Interrupts in Kernel Threads

- Hardware interrupts wake up a kernel thread
- Threads are scheduled and compete with other processes
  - Can be given a priority lower than userspace processes
  - “With great power comes great responsibility”

Why did we do all of this?

- In August 2006, IBM announced a real-time JVM/SDK product (IBM Websphere Real-Time v1.0) which requires a productized version of the CONFIG_PREEMPT_RT patches.
- The “Linux Real-Time Extensions” is distributed as “patches and scripts” which extend a RHEL4 32-bit x86 installation.
  - Support is provided by IGS Supportline with L3 support from the IBM Linux Technology Center

Features of the real-time JVM

- RTSJ: Real-Time Specification for Java
- Metronome: A real-time garbage collector
- AOT: Ahead of Time compilation

RTSJ – Real-Time Specification for Java

- First extension to the Java language
- Architecture did not define a real-time garbage collector
- Instead defined NonHeapRealTime threads
  - NHRT threads can only use scoped and immortal memory
  - NHRT threads may not reference normal “Heap” memory, since the must stop all normal threads in case it needs to move Java objects, and that would be unacceptable for real-time threads
    - Any attempt to reference normal memory from a NHRT thread causes a uncatchable exception
    - Means that that NHRT can not use most Java libraries unless they are carefully audited not to read, write, or allocate objects on the heap. Most Java libraries are not NHRT-safe.

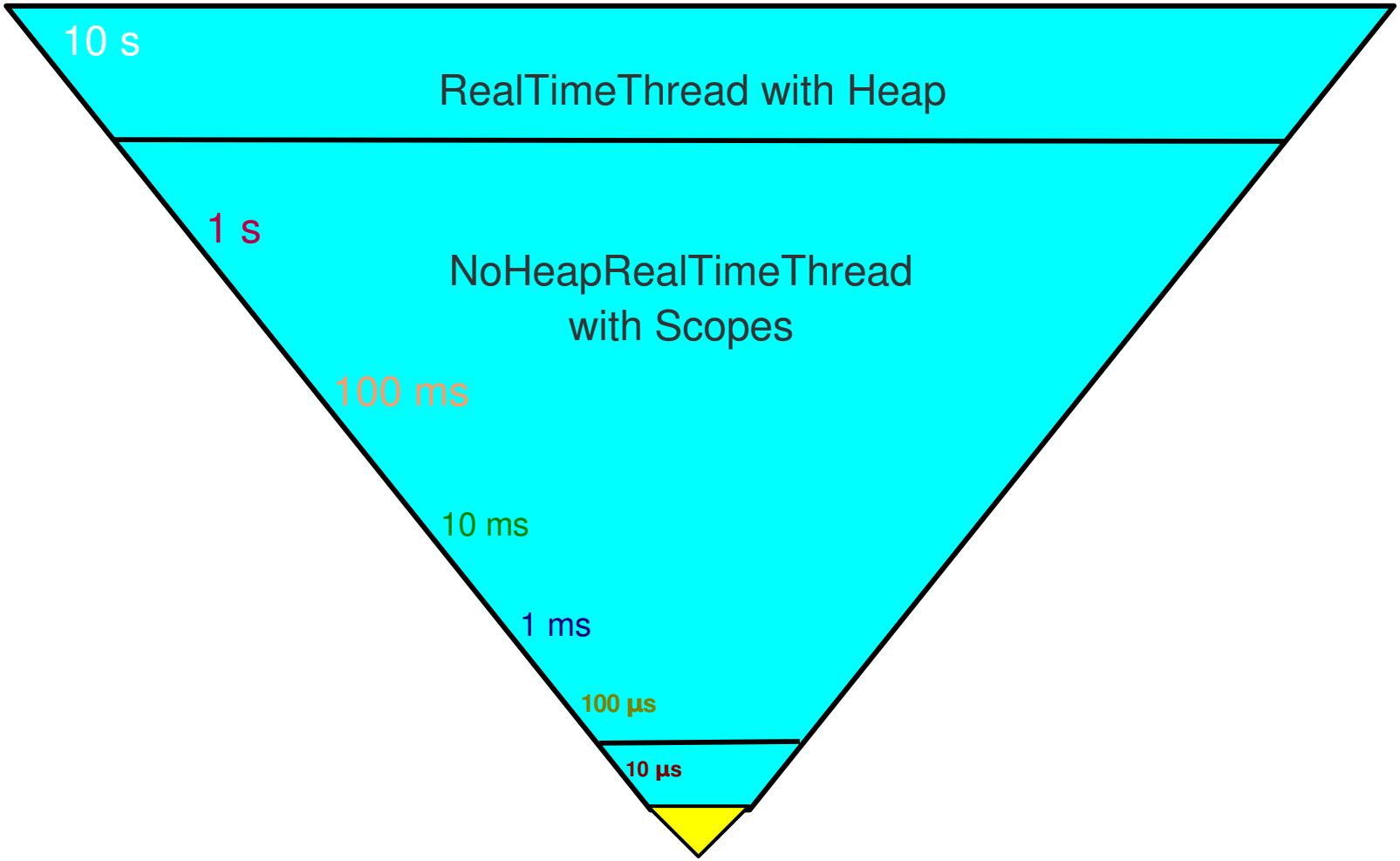
Metronome – a real-time garbage collector

- An incremental garbage collector designed by David F. Bacon from IBM Research for real-time workloads. Tuning knobs:
  - Max. time slice "stolen" by the GC (e.g., 100us)
  - Max. % of CPU time used by the GC (e.g., 30% over 1ms)
  - What to do if application generates more garbage than the GC can handle:
    - Print a warning message and do a "stop the world" GC
    - Abort the program since real-time guarantees cannot be sustained
  - Max. heap memory used by the JVM
    - Increasing this can help if the application has a "bursty" garbage generation pattern
- IBM unique value: no other JVM has this!

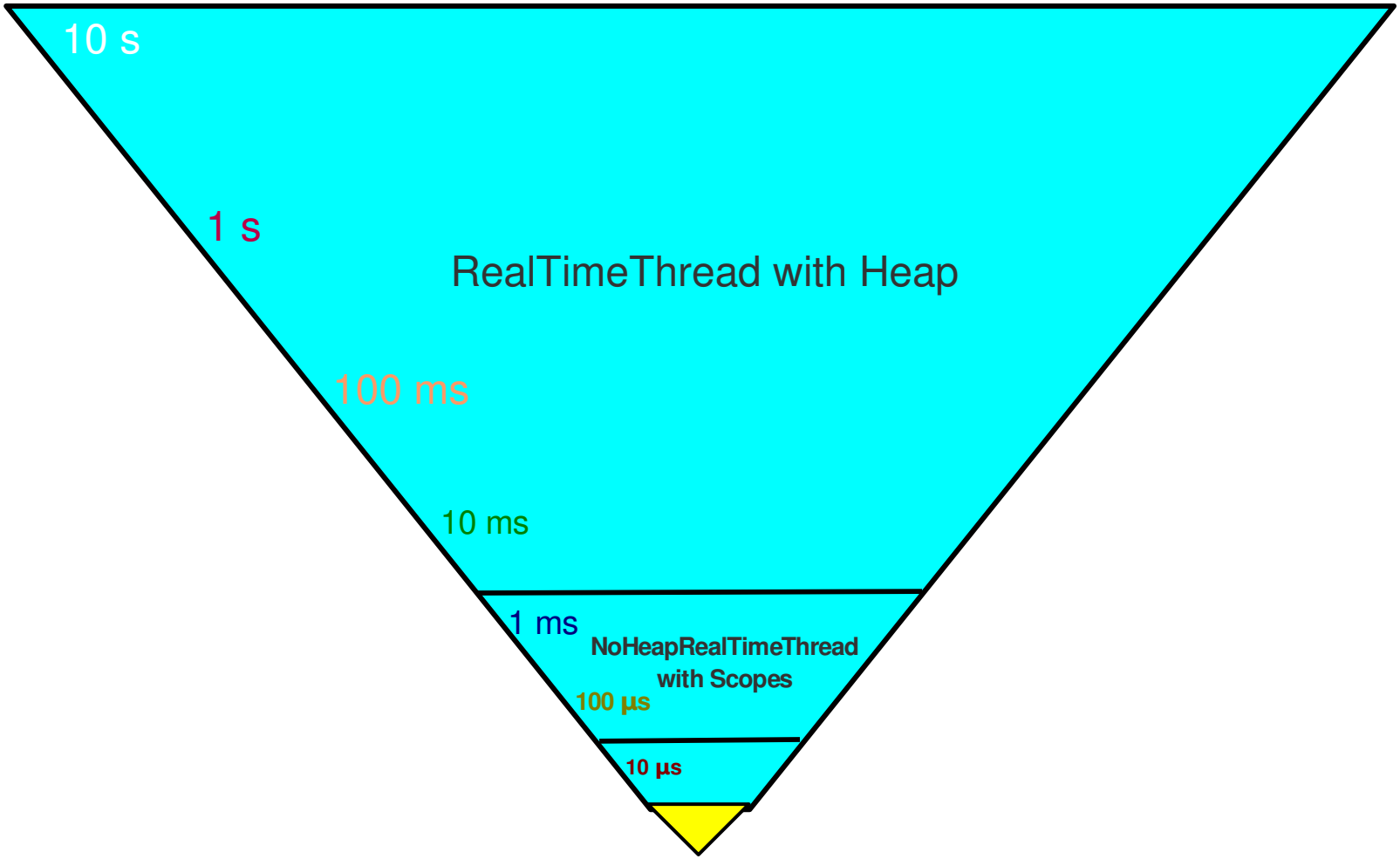
AOT – Ahead of Time compilation

- Reduces pause time caused by the JIT compiler
- Difficult because Java has language features that require a JIT compiler for performance
  - e.g., subclasses defining new types can be loaded dynamically at run-time
- AOT's performance midway between interpreted code and fully optimized JIT-generated code.

Without Metronome and AOT



With Metronome and AOT



Why Real-time Java?!?

- For the US Military/Defense Contractors, Java is the new ADA
  - For some reason, hard to find ADA programmers...
  - Weapons systems
    - need real-time
    - are becoming more integrated with each other and more complex
- Financial Sector
  - Automated trading and automated market-making applications have real-time requirements
- Web sites with response-time guarantees
  - In an multi-tier architecture, an individual tier may require millisecond response times to meet a sub-second page load time requirement

Conclusion

- Enterprise Real-Time: a new way of thinking of real-time
  - More powerful and feature rich than traditional hard real-time
  - Better determinism that generally expected of soft real-time
- IBM's Websphere Real Time v1.0 and Real-Time Linux Extensions as the first realization of the enterprise real-time concept
  - Available to lead adopters today
  - Metronome and AOT make use of real-time in Java programs much easier and more palatable.

Legal Statement

- IBM is a registered trademark of International Business Machines Corporation in the United States and/or other countries.
- Linux is a registered trademark of Linus Torvalds.
- Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- Other company, product, and service names may be trademarks or service marks of others.