HPI Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Realtime Java

Christoph Neijenhuis

Agenda

- RT Problems in Java
- Solutions: RTSJ and others

- Guiding Principles
- Memory Management
- Threads
- Asynchronous Event Handling
- Scheduling
- Time
- Synchronization

- Implementations

Problems in Java

- Garbage collection can occur at any time
- VM is optimized for through put
  - Lazy initialization, e.g. for class loading
  - Compilation of code at run-time
- VM doesn't cooperate with realtime OS
- Threads can not be scheduled or prioritized reliably
- APIs are not tailored for real-time scenarios
  - e.g. time is only represented with millisecond precision

Solutions

- Real Time Specification for Java
  - Defined as JSR 1
  - Released in 2002
  - Reference implementation from TimeSys
  - Commercial implementations from Sun and IBM
- Other proprietary solutions exists
  - Support only a subset of standard Java
  - Tailored for embedded systems with limited memory
  - E.g. PERC Pico from aionix
  - Or support a subset of RTSJ
  - E.g. JamaicaVM from aicas GmbH

Guiding principles

- No syntactic extension
- Backward compatibility
- Applicability to particular Java environments

- Current practice and advanced features

- Allow variation in implementation decision

Memory Management

- Additional memory areas that are not garbage collected
- Immortal memory
  - Objects exist until end of application
- Scoped memory
  - Exists for a limited lifetime
  - Reference counting, finalization when it reaches 0
  - Scopes can be nested
  - No references into scope from heap or immortal memory
- Physical memory
  - For direct communication with hardware

Memory Management

Reference to Stored In	Heap	Immortal	Scoped
Heap	**Permit**	**Permit**	**Forbid**
Immortal	**Permit**	**Permit**	**Forbid**
Scoped	**Permit**	**Permit**	**Permit**, if the reference is from the same scope or an outer scope

Memory Management

- Garbage collection algorithm must be described
- “precisely characterize a implemented GC algorithm's effect on the execution time, preemption, and dispatching of real-time Java threads”
- Must be preemptable by Threads that do not use heap
- May be preemptable by Threads that use the heap

Memory Management

- Memory usage of threads can be limited with MemoryParameters
  - Define maximum memory usage for scoped or immortal memory
  - Define maximum allocation rate (in bytes) for heap
- If more memory is used, a OutOfMemory Exception is thrown

Threads

- RealtimeThread extends java.lang.Thread
- Reuses most concepts, but not priority
- ReleaseParameters, SchedulingParameters and MemoryParameters may provide information to the scheduler

- NoHeapRealtimeThread extends RealtimeThread
- Can only use scoped and immortal memory
- Therefore runs in preference to the garbage collector

Examples


```
PriorityParameters priority = new PriorityParameters(15);
RelativeTime relative = new RelativeTime(20 /* ms */,0 /* ns */);
PeriodicParameters period = new PeriodicParameters(relative);
RealtimeThread rt = new RealtimeThread(priority, period);

ScopedMemory memory = new LTMemory(1024); // size in bytes
NoHeapRealtimeThread nhrt =
    new NoHeapRealtimeThread(priority, period, memory);
```

Asynchronous Event Handling

- AsyncEvent class can be:
  - bound to external, implementation specific event
  - fired by application
- One or more AsyncEventHandlers can be assigned
- Released when the event occurs
- Execution inside a (NoHeap-)RealtimeThread

Scheduling

- RealtimeThreads, AsyncEventHandlers, but also standard Java threads need to be scheduled
- Specification requires one scheduler (PriorityScheduler)
- Implementation may define further schedulers
- PriorityScheduler is fixed-priority, preemptive with at least 28 priorities
- Subclasses of SchedulingParameters (e.g. PriorityParameters) provide values for scheduling decisions
- Subclasses of ReleaseParameters are Periodic-, Aperiodic- and SporadicParameters
- Define release time and may define a deadline and cost
- May define handlers for deadline miss or cost overrun

Scheduling

- Scheduler must implement feasibility algorithm
- IsFeasible() returns true if all known schedulable objects described by their SchedulingParameters and ReleaseParameters are feasible for the current system
  - Implementation must define feasibility
  - Standard implementation for PriorityScheduler:
    - Return true if no AperiodicParameters is set
- Optional: Cost enforcement of CPU utilization

Examples


```
PriorityParameters priority = new PriorityParameters(15);

AperiodicParameters aperiodic = new AperiodicParameters();
aperiodic.setCost(new RelativeTime(2, 500000));
aperiodic.setDeadline(new RelativeTime(5, 0));
aperiodic.setDeadlineMissHandler(missHandler);

AsyncEventHandler handler = new AsyncEventHandler(
        priority, aperiodic, null, null, null, false);

AsyncEvent externalEvent = getExternalEvent();
externalEvent.addHandler(handler);
```

Time

- HighResolutionTime describes time with nanosecond accuracy
  - Actual accuracy depends on underlying system
  - long getMilliseconds()
  - int getNanoseconds() (min: -9999999 max: 999999)
- Subclass AbsoluteTime defines absolute point in time
- Subclass RelativeTime represents a time interval

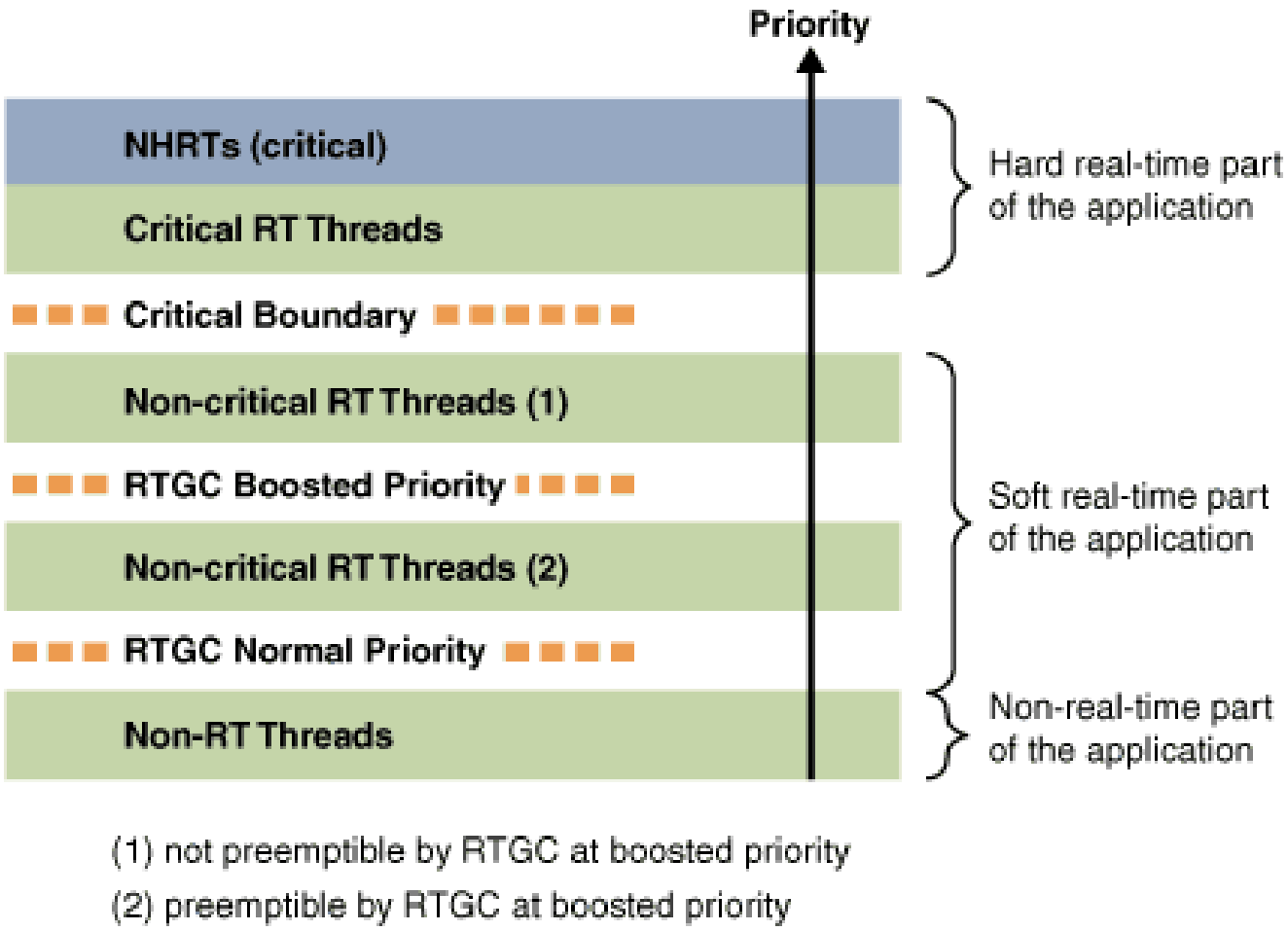
Synchronization

- Java provides synchronized keyword for synchronization but no prevention of priority inversion
- PriorityInheritance is default synchronization policy
- PriorityCeilingEmulation is defined, but optional

Implementations

- Reference Implementation from TimeSys (2002)
  - Runs on X86/Linux
  - Not commercially available
- Sun Java SE Real-time (2005)
  - Runs on Solaris 10, SUSE & RedHat Realtime Linux
  - Based on Java SE 5
  - Min requirements: Dual core or dual CPU system with 512 MB
- Apogee RTJRE (2006)
  - Runs on Linux, based on IBMs JVM
  - Targets embedded systems, includes version for Java ME
- IBM WebSphere Real Time (2006)
  - Runs on X86/Linux

Garbage Collection in the Sun VM



Source: http://java.sun.com/javase/technologies/realtime/reference/doc_2.2/release/figures/ThreadPriority.gif

References

- http://jcp.org/en/jsr/detail?id=1
- http://java.sun.com/javase/technologies/realtime/index.jsp
- http://www.rtsj.org
- http://www.apogee.com/products/rtjre
- http://www.dcl.hpi.uni-potsdam.de/teaching/EmbeddedOS/Slides/05
- http://java.sun.com/javase/technologies/realtime/reference/doc_2.2/