Issues with Java for Real-Time

Kelvin Nilsen

Chief Technical Officer, NewMonics Inc.

NewMonics Vision

- **Java programmers should be able to write real-time programs**
  - Deterministic memory usage
  - Deterministic tasking behavior (execution time, task interaction)
  - Expressive power (priorities, signals, semaphores, messages)
- **Real-time Java programs should be portable!**
  - Fast or slow computers
  - Big or small computers
  - Whether dedicated to single task, or serving multiple needs

Writing Real-Time Applications in Java

- **Unpredictable task execution times**
  - Complicated control flow; absence of analysis tools; JIT translation; method caching, inlining, other optimizations
- **Unpredictable memory requirements**
  - Conservative garbage collection, failure to defragment, complex library services
- **Unpredictable task interaction**
  - Poorly defined priority system, priority inversion, non-incremental garbage collector, complex dynamic workload
- **Weak vocabulary**
  - Messages, semaphores, interrupts, IO, signals, timeouts

Writing Portable Real-Time Applications

- Applications need to be able to determine their resource (time and memory) needs.
- API must allow applications to request dedicated resources.
- Virtual machine must commit resources to particular real-time workloads.
- API must allow applications to manage resources dedicated to their execution.

Writing Embedded Applications

- Severe memory and power constraints
- Reliability requirements
- Execution from ROM
- Hardware devices, interrupt handling

What NewMonics Does for Real-Time

- **Fixed priority tasks**
- **Priority inheritance**
- **Priority-ordered wait queues**
- **Real-time garbage collection**
  - accurate
  - defragmenting
  - incremental
  - aggressive

How We Make Real-Time Portable

- **Real-time activities configure themselves**
  - API supports measurement of resource usage
  - API and run-time environment support analysis of resources
- **System executive negotiates resource budgets and then enforces them**
- **Each activity takes responsibility for efficient utilization of budgeted resources**
  - Timed and atomic statements
  - Awareness of ongoing resource consumption and availability

Example of Timed and Atomic

```
computeApproximation();
refinements = 0;
try {
  timed (Time.ms(N)) {
    for ( ; ; ) {
      refineApproximation();
      atomic { updateApproximation(); refinements++; }
    }
  }
}
catch (TimeOutException t) {
  adjustN(refinements);
}
```

What We Do For Embedded

- **ROMizer™**
- **picoPERC (64 Kbyte subset of JVM)**
- **MWT (One third the size of AWT)**
- **Special API libraries:**
  - Persistent
  - IOPort
  - InterruptVector

(Mix in real-time capabilities as desired...)

What Might This Group Accomplish?

- **Definition of terms**
  - ROMable, JIT, AOT, flash, soft-real-time, hard-real-time, “real time”, response time, etc.
- **Specification of testing methods, benchmarks, compliance tests**
  - How to measure response time. How to demonstrate “soft real time”. Etc.
- **Define standard APIs:**
  - Core libraries, standard optional components, application-specific libraries
- **Define language, subsets, and supersets**