Java for Real-Time Programming

Jan Lindström (jplindst@cs.helsinki.fi)

Seminar on Object architectures

Fall 1998

Contents

Introduction

- Java originally designed for embedded system development.
- Java offers important high-level benefits to developers of traditional software.
- It would be desirable for real-time dialect of Java to improve upon the state of the practice in real-time development.
- Java lacks capabilities for the reliable development of portable real-time systems.

Real-Time Programming

- Software that is constrained to operate within specified time constraints is said to be “real-time”.
- Soft real-time requirements are such that a slightly late action has greater value than no action at all.
- Firm real-time requirements are such that a late action has zero value.
- Hard real-time requirements are such that a late action has negative value.

Types of Real-Time Tasks

- Periodic: executes regularly according to a precomputed schedule.
- Sporadic: triggered by external events, whith a predetermined maximum execution frequency.
- Spontaneous: optional real-time task that are executed in response to external events, if resources are available.
- Real-time threads: ongoing tasks.

Designing Real-Time Software

- Designers specify real-time constraints.
- Engineers implement routines to provide the necessary services and analyze these routines to determine their worst-case execution times (WCET).
- Engineers analyze the combined workload to determine whether individual real-time tasks can be proven to not interfere with each other's real-time requirements.

Writing Portable Real-Time System

- Applications need to be able to determine their resource 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.

Problems

- Writing real-time software is difficult.
- Lack of automated tools.
- Every programmer must act globally
- Programming languages lack real-time semantics.
- Over 50 real-time operating systems.
- Tens of incompatible “popular” embedded microprocessors.
- Architecture upgrades not upward compatible.

Results

- “black magic” C and C++
- Compile time analysis tools
- pre-run-time measurements of application resource requirements.
- run-time interaction with non-standard, real-time operating system services.

CASE: RODAIN

- Real-Time Object-Oriented Database System for Intelligent Networks.
- Firm real-time tasks.
- Chorus Classix3.1r real-time operating system.
- Main memory database.
- Priority based scheduling.
- Optimistic concurrency control.
- C and C++ implementation.

Java Problems I

- Over-sold, creating unrealistic expectations.
- Development toos are less mature than C and C++ equivalents.
- Require more memory and CPU horsepower.
- Java lacks determinism of resource management
  - Time to execute programs is unpredictable
  - Memory requirements are unpredictable
  - Task priorities and syncronization are unclear

Java Problems II

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

PERC Real-Time API I

- Superset of Java 1.1 in that it offers additional syntax and additional time- and memory-related semantics.
- Subset of Java in that it forbids certain legal Java practises.
- Source code is translated to Java byte codes by **percolate** compiler.
- Percolate recognized and translates the special **timed** and **atomic** control structures.
- Percolate does additional analysis of the source code and encode certain of the results of this analysis in generated byte code files.

PERC Real-Time API II

- Fixed priority tasks
- Priority inheritance
- Priority-ordered wait queues
- Real-time garbage collection
- Resource configuration.
- Resource negotiation.
- Reconfiguration.
- Resource budgeting.
- The **Embedded** and **RealTime** Toolkits.

PERC Real-Time API III

- Ability to analyze code segments to determine worst-case execution times.
- Support for measurement of CPU-time and dynamic memory needs.
- Real-time executive that supports reliable timeliness of periodic task execution and enfroces time and memory budgets for each real-time activity.

Sample **timed** statement

```
try{
   timed(Time.ms(20)) {
        Method()
   }
} catch (TimeOutException t) {
   DoSomething(t) // Error
}
```

Sample **atomic** statement

```
try {
        atomic{
                BeginTrans();
                WidthDraw(FromAccountId,Sum);
                Deposit(ToAccountId,Sum);
                Commit(); // thows AbortException
        }
} catch (AbortException t) {
        DoSomething(t) // Error
}
```

Portable Real-Time Systems

- Real-time activities configure themselves.
- System executive negotiates resource budgets and then enforces them.
- Each activity takes responsibility for efficient utilization of budgeted resources.

Summary

- Java offers important high-level benefits to developers of traditional software.
- It would be desirable for real-time dialect of Java to improve upon the state of the practice in real-time development.
- In its current form, Java is not appropriate for the development of real-time software.

Links

- http://java.sun.com/features/1997/july/hubble.html
- http://java.sun.com/features/1997/july/mars.html
- http://java.sun.com/features/1998/09/jtrack.html