UNC CHARLOTTE

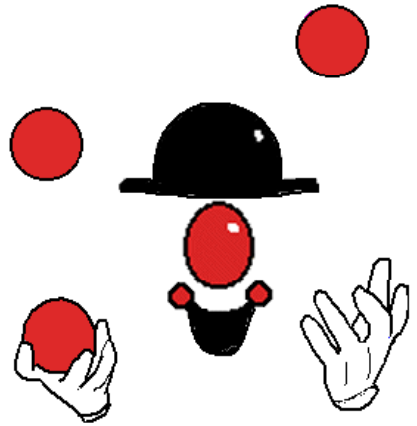
Design of a Real-Time Virtual Machine (RTVM)

Published at the
Canadian Conference on Electrical and Computer Engineering
at Saskatoon, Saskatchewan Canada

May 2005

A gist of Real Time OS

- RTOS does multitasking (juggling) and multithreading
- Gives a good response time and allows addition of new tasks to be added easily.
- Provides valuable services like semaphores, mailboxes, data structures.



➢ The biggest problem of an RTOS is that it is proprietary, expensive (commercial versions) and may not meet real-time constraints such as hardware changes.

What is an Real Time Virtual Machine ?

- A Virtual Machine emulates hardware.
- Virtual Machines are independent of underlying hardware. (e.g.: VMware, Microsoft Virtual PC)
- Provide a generic independent platform instead of reinventing the software.

The paper speaks of a Java paradigm for implementing RTVM for embedded RTOS

C versus Java

C/C++

- Compiled
- Prone to crashes
- From Embedded POV, platform dependent
- Only C++ can be considered as an OOP language
- Memory model: stack

- FAST but tends to be BUGGY

Java

- Interpreted
- Sandbox model ensures lesser crashes and better security
- Inherently platform independent
- Allows for Object Oriented Programming standards
- Memory model: heap

- SLOWER but SECURE

About Java

- Philosophy: Write Once, Run Anywhere
- Java is interpreted not compiled.
- Java compiles code to byte code
- JVM interprets the byte code at runtime

Advantages:

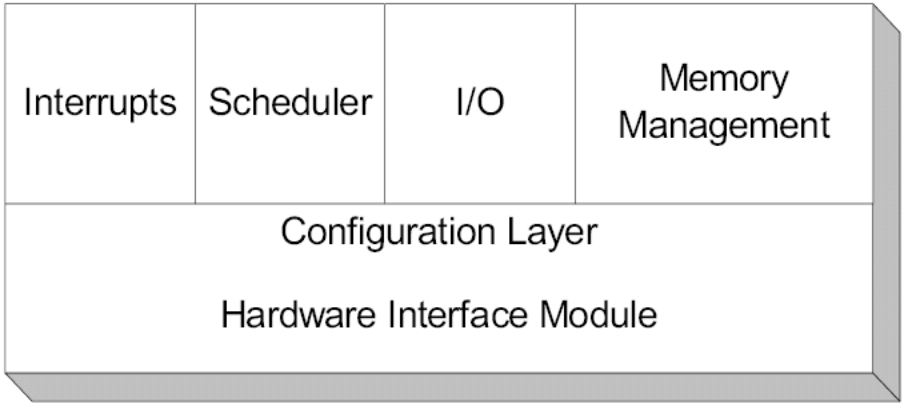
- Platform independent.
- More secure compared to C/C++
- Sandbox model

Disadvantages:

- Slower
- From Embedded POV, no pointers.

Redrawn & Referenced from java.sun.com and med.harvard.edu

The RTVM Architecture



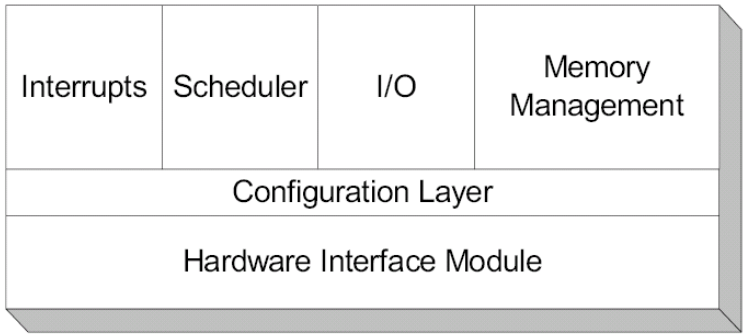
General Ideas discussed:

- Abstraction
- Hardware Interface Module
- Configuration of Real Time Virtual Machine
- Portability

RTVM Architecture (contd.)

Hardware Interface Module:

- Lies just above the Hardware platform
- Performs Abstraction from Hardware
- "Plug-in-able" and adaptable to other RTVMs



RTVM Core Services:

- Interrupt Handling
- Event Handling
- Memory Management
- Scheduler

Configuration Layer:

- Defines behavior
- Helps control process memory allocation, scheduler policy, etc.

Application Portability

- HIM abstracts “hardware idiosyncrasies” from the RTVM core services. (similar as the HAL implementation in Windows and Linux)
- Applications in RTVM uses shared libraries for the C/C++ languages which are built into RTVM
- Applications compiled on any native compiler can run on RTVM as long as shared libraries are compatible.

Java Virtual Machine in Real Time

- Java Real-Time Specification (JRTS) used to address response and predictability. (Includes many compile methods)

- Memory Management: Garbage collection (reclamation of memory used by objects) system for improving response, current area of research.

- Scheduler: Many predictive scheduler systems are being researched, most popular being a rate-monotonic scheduler. The RTOS must provide a pre-emptible mechanism.

- Hardware Interface: HIM creates a configuration pattern to support the core services.

Thoughts about the paper

- Innovative idea.
- Tackles a real problem and proposes a good solution.
- Even though shared libraries are available to be used by GCC, details of the abstraction would still need to be known.
- Is implementing an RTVM cheaper than upgrading an RTOS ?