Java Technology for Embedded Systems

lsi@mes.tu-darmstadt.de

Outline

- Motivation
- Embedded Systems
- Java Technology
  - language
  - application programming interface
  - virtual machine
- Java in Embedded Systems
  - advantages and disadvantages
  - execution possibilities
  - virtual machine options
- Perspectives

Motivation

- „By the year of 2002, it is estimated that more information appliances will be sold to consumers than PCs“ (Business Week)
**Motivation**

- 700 million **cell phones** to be sold worldwide in 2003, about 80% of them offering connections to the Internet” (Gartner Group)

**Embedded Systems**

- Embedded systems:
  - a set of pre-defined, specific functions to be performed
  - the resources available (e.g., memory, power, processor speed, computational functionality) are constrained

- Special-purpose computer built into and integral to a device, used to control, monitor or assist an operation

**Motivation**

- Telecommunication
- Networking
- PDAs
- Automotive Electronics
- Aerospace industry
- Multimedia (digital cameras, digital audio players)
- Home automation (domotics)
- Internet appliances
- Toys and games

**Embedded Systems**

- Until recently the vast majority of the embedded systems used 8- and 16-bit microprocessors, requiring little in the way of sophisticated software development tools

- The breaking of the $5 threshold for 32-bit processors drove an explosion in high-volume embedded applications

- A new trend towards integrating a full system-on-a-chip (SOC) promises a further dramatic expansion for 32-bit embedded applications
Development started in early 90’s under the Green Project at Sun Microsystems
Initial goals included reliable software for consumer electronics
With the popularization of Internet, the main focus moved to web based programs (Applets and Servlets)
As the market for embedded systems grows, and as the embedded system design becomes complex, Java is turning back to its initial objective and it is being used to boost the development of several products

Java Technology may be divided in four building blocks [VENNERS 98]
- Java programming language
- class file format
- Java API (Application Programming Interface)
- Java Virtual Machine (JVM)

Java Language
Features described in the Java Language Whitepaper released in 1995:
- simple, familiar - looks like C++, but without its unnecessary complexities
- object oriented - designed to be OO from ground up, implementing elegantly the OO concepts (inheritance, polymorphism, encapsulation, dynamic linking)
- robust, secure - compile-time and runtime code checking, simple memory management model, exceptions, sandbox security model

Features described in the Java Language Whitepaper released in 1995:
- high performance - JIT compilers, native code interface, garbage collection running as backgorund task
- threaded - allows multiple concurrent lightweight processes and provides synchronization facilities
- dynamic - compiler is strict in its compile-time static checking, but the run-time system is dynamic in its linking stages
- architecture neutral, portable - intermediate code format (bytecodes) can be executed in several platforms through JVM, data types and arithmetic operations behavior are standardized
Java API

- Class libraries covering a wide range of subjects:
  - graphic user interface
  - 2D and 3D graphics
  - threads and sync
  - i/o
  - data structures
  - networking
  - cryptography
  - data compression
  - multimedia
  - e-commerce
  - applets
  - servlets
  - XML
  - distributed systems
  - telephony
  - speech recognition
  - database connection
  - text processing

  and more...

Java API

- The first versions of the Java Development Kit had a single, standard API (JDK 1.0 and JDK 1.1)
- When the Java 2 platform was released (JDK 1.2 and JDK 1.3), different flavors of the API were defined:
  - J2SE™
  - J2EE™
  - J2ME™

Java 2 Platform, Standard Edition (J2SE™) - tools, runtimes, and APIs for developers writing, deploying, and running applets and applications

Java 2 Platform, Enterprise Edition (J2EE™) - architecture with an Application Programming Model and Compatibility Test Suite for building enterprise-class server-side applications

Java 2 Platform, Micro Edition (J2ME™) - highly optimized Java runtime environment targeting a wide range of consumer products, including pagers, cellular phones, screenphones, digital set-top boxes and car navigation systems

Java Virtual Machine

- The JVM is the key point on Java's platform independency
- It lays between the application and the target system, making it possible to run Java code in most combinations of hardware/OS
Java Virtual Machine

- class load subsystem
  - object allocation

- method area
- Java code stack
- heap
- regs. PCs
- native method stack

- runtime memory area
- execution engine
- native methods interface
- native methods library

- program counter

Java Virtual Machine

Java programs comprehend one or more threads, and each of them has its own stack and PC.

The stack is divided in frames, which contain the status of the invoked methods.

The frame has separate areas for storing the method local variables, pointers for the invoker of the method and the operand stack.

- bytecode instruction execution engine implemented in software, hardware or both

- local variables
- invoker info
- operand stack
- local variables
- invoker info
- operand stack
Java Technology Overview

Java in Embedded Systems

Based on its motto for Java - „Write once, run everywhere“ - Sun Microsystems is pushing Java back to the embedded software development.

One particular API (J2ME) and two specifications (Embedded Java and Personal Java) address the subject.

Another API (JavaCard API) may also be considered suitable for embedded systems with limited processing and memory resources - it is designed to run in 100kb.

Java in Embedded Systems

Advantages on using Java as embedded software

- simpler programming, when comparing to C/C++ or assembly
- OO paradigm behind Java language enforces modularity, data abstraction and thus design reuse
- rich and useful APIs (one standard and several custom)
- portability
- pointer safe
- array indexing check
- exceptions
- code compactness

Disadvantages on using Java as embedded software

- resources for runtime environment
- garbage collection
- no direct access to hardware
The execution possibilities of Java code depends on several architectural options:

- integration with the rest of the system
  - native method call
  - separate set of threads
- storage and execution of Java bytecodes
  - host machine
  - target
- executable code generation
  - interpreter
  - AOT compiler
  - bytecode execution

- storage and execution of Java bytecodes
  - bytecodes stored in target device, but executed in host (not actually Embedded Java, but Embedded Server)
  - bytecodes stored in host, but executed in target device
  - bytecodes stored and executed in target device - there might not even be a host
Java in Embedded Systems

- Executable code generation
  - Interpreter - bytecodes translated at runtime
    - Slower
    - Requires memory and processing resources
    - Keeps with platform independency
  - AOT compiler - native code generated at compile time
    - Faster
    - Don’t need runtime environment
    - Platform dependent

- JIT compiler - bytecodes translated at runtime, but with optimizations
  - Mid-term solution
  - Currently being used as default for Sun's JDK

- Bytecode direct execution - requires hardware-implemented Java Virtual Machine
  - Faster
  - Some hardware solutions provide execution of both bytecode and regular instruction set

Perspectives

- Picocore - licensed under Sun Community License
- WindRiver RTOS - uses special resources and AOT compilation to allow Java-implemented GUIs
- Esmertec J Bed - RTOS + JVM, no additional software needed
- ARM 9 RISC Core - set of extensions to the ARM RISC architecture lets an ARM9 core directly execute Java bytecodes as if they were native instructions
- Kaffe - free and portable software JVM
- GCJ - Gnu AOT compiler for Java

Motorola iDEN series, Nokia 9210 and DoCoMo’s i503 iMode - first generation of cellular phones supporting J2ME
- Sega develops games for next generation Motorola Phones using J2ME
- JVM for Palm and Symbian OS for handheld computers
**Perspectives**

- TINI Board, Dallas Semiconductors

**Acknowledgement**

- Ito, S.A. Projeto de Aplicações Específicas com Microcontroladores Dedicados. PPGC, UFRGS.
- Bothner, P. Compiling Java for Embedded Systems.
- www.netrino.com
- java.sun.com
- www.samsung.com
- www.visteon.com
- www.teletask.be
- www.motorola.com
- www.embeddedjava.net
- www.embedded.com
- www.sony.com
- www.artima.com