
JavaOne℠
Sun's Worldwide Java Developer Conference

JavaOne℠

Sun's Worldwide Java Developer Conference

JavaOS™ – Java™ on the Bare Metal

Peter Madany

Outline

- Goals
- Hosted Java
- JavaOS
  - Technical overview
  - Advantages
  - Target systems
  - Status
- Demo
- Future

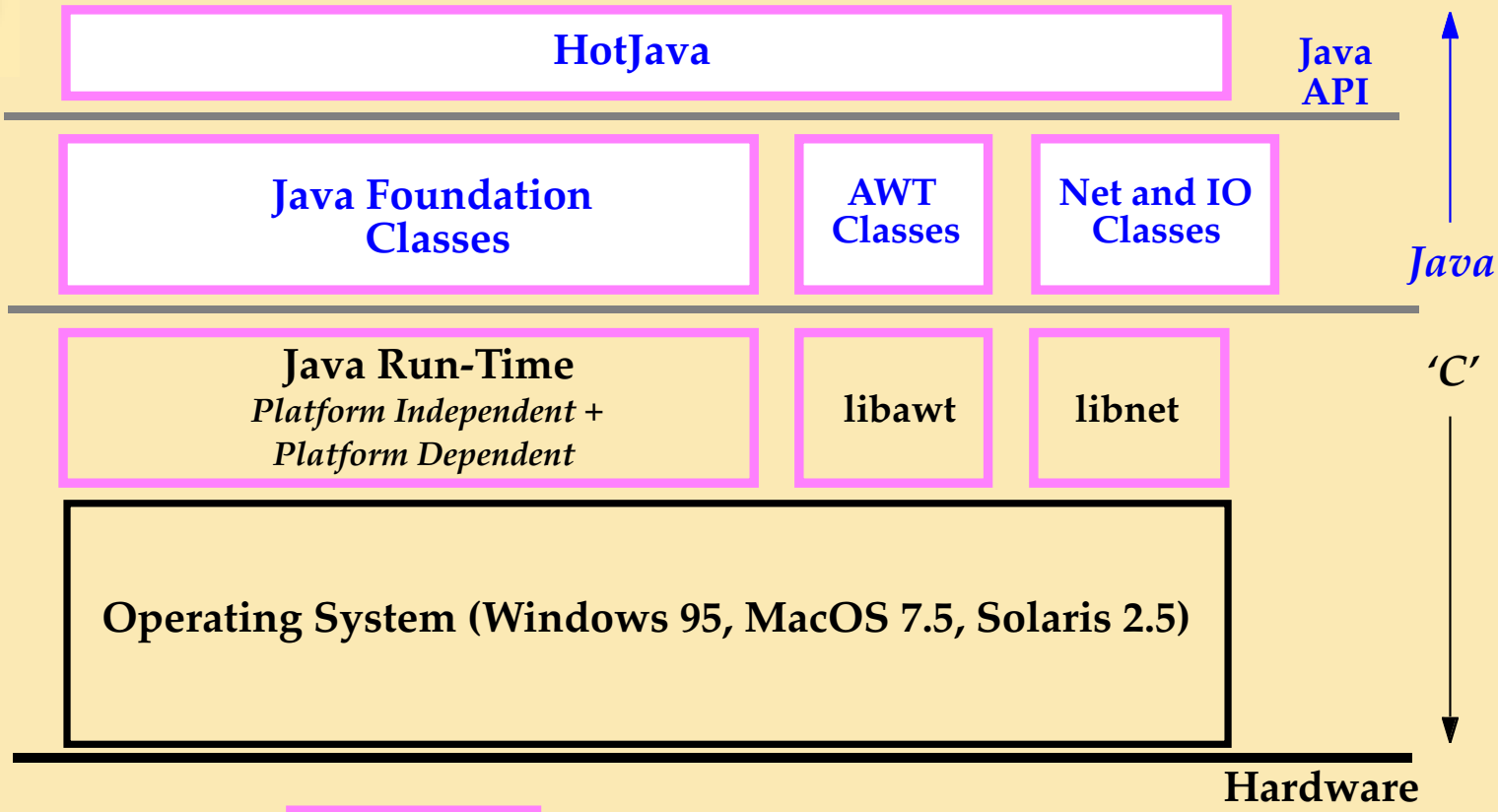
Goals

- Enable new types of network devices
  - Intelligent
  - Dynamic
  - Simple to install, administer, and use
- Run Java in systems with limited resources
  - Java-enabled devices
  - Cannot assume high-end hardware and software
- Limited Hardware:
  - Limited Memory: RAM, ROM
  - Optional items display and disks
- No host operating system
  - Minimize memory requirements

How Features Are Delivered

- Today: the Java language is embedded in applications
  - e.g. browsers
- Soon: embedded within operating systems
  - e.g. in Solaris, Windows, Mac OS
- Both rely on a host OS
- Conclusion: today Java requires a host OS

Java on a Conventional OS



Major Features

- Java Virtual Machine
  - Byte-code interpreter
  - Class loader
  - Garbage collector
- Language and Utility Classes
- AWT Classes
- Net Classes
- IO Classes

Host OS Requirements

- Support for threads
  - Context switching
- Memory Allocation
- Window system
- Network Protocols
- File Systems
- Device drivers
  - Keyboard
  - Mouse
  - Display

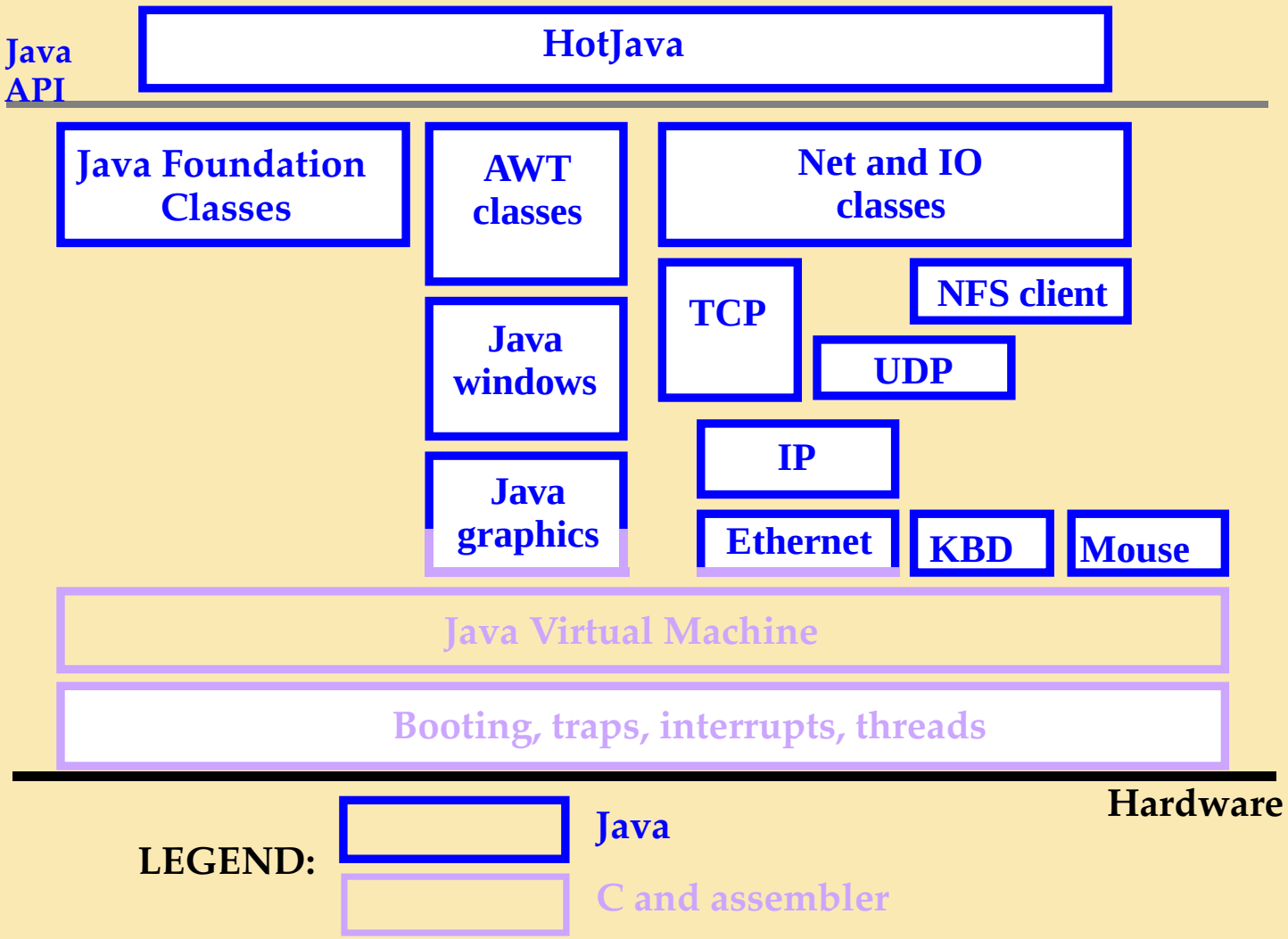
Matching Java to a Host OS

- Map AWT to window system
- Map Net classes to native networking
- Map file-related IO classes to file system
- Port platform-dependent part of VM

Java Without a Host OS

- Provide just enough kernel features:
  - To implement the Java VM
- Just enough Java™-based code for:
  - AWT, Net, and File IO
- Necessary device drivers
  - Display and network
  - Mouse and keyboard
- Still support the full Java Platform API

JavaOS™ Architecture



JavaOS™ Details: JavaKernel

- Bootstrap
  - Memory allocation
  - Device mapping
- Trap and interrupt handling
- Thread support
  - Simple context switching
  - Single address space

JavaOS Details: Virtual Machine

- The virtual machine implements “Java”
  - Bytecode interpreter loop
  - Java exception handling
  - Memory management
  - Threads
- JavaOS mostly uses standard VM
  - Tuned the malloc and Java™-compatible heaps
  - Class garbage collection

JavaOS™ Details: ROMability

- Java™-based byte codes are modified when run
  - Cannot be placed as is in ROM
- The ROMizer tool
  - Converts set of class files into ROMable, bootable Java™-based images
  - Performs some optimizations on the image

JavaOS Details: Device Drivers

- All device drivers written in Java
  - Use two small “C” support classes
  - For “memory objects” and interrupt dispatch
- We have working device drivers for:
  - Ethernet
  - Keyboard, mouse, serial ports, and clock
  - Several frame buffers, including VGA
  - Audio
- Designing a Java™-compatible Device Driver Interface

JavaOS Details: Networking

- Written in the Java language
- We currently have working versions of:
  - TCP, UDP, IP and ICMP
  - ARP for address resolution
  - DNS or NIS for hostname lookup and login
  - DHCP or RARP for address discovery
  - NFS client-side, including SunRPC/XDR
  - SNMP agent for network management
  - Socket support
  - Network time

JavaOS Details: Windows & Graphics

- Uses the “Tiny AWT” widget library
- X-windows would be too heavy-weight
- Window system
  - Small window system
  - Written in the Java language
  - Optimized for limited memory
- Graphics rendering package
  - Mostly written in Java
  - Designed for AWT
  - Lowest level uses native methods written in “C”
  - Bitmapped fonts

JavaOS Details: Applet API

- Same code
- Same API

HotJava As “Desktop”

- HotJava
  - HTML browser
  - Written in the Java language
  - Supports multiple windows
  - Can run multiple applets per window
  - Dynamically extendable
- Could run other Java™-based programs
  - Possibly customized versions of HotJava

JavaOS Is Not an OS

6. It doesn’t need a file system
5. It doesn’t need virtual memory
4. It doesn’t need separate address spaces
3. It doesn’t support more than one language
2. It doesn’t have its own set of system calls
1. ...

JavaOS Is Not an OS

1. Marketing tells people it is NOT an OS

JavaOS Is an OS

9. You can boot it
8. You can log in to it
7. It safely runs several applets at a time
6. It drives devices
5. It networks
4. It does windows
3. It has an API – the Java Applet API
2. Tons of applets are being written for it

JavaOS Is an OS

1. Marketing tells people it is an OS

JavaOS Performance: Speed

- System built without using a JIT compiler
- Network performance
  - Current TCP/IP throughput is 500K Bps
  - Little tuning
  - No native methods written in C

JavaOS Performance: Speed

- Pendragon CaffeineMark

Category	Netscape/Solaris	HotJava/Solaris	HotJava/JavaOS
Seive	31	33	33
Loop	27	27	29
Graphics	89	102	122
Image	57	96	265
CaffeineMark	51	64	112

JavaOS Performance: Memory

- 4MB ROM
  - Code for JavaOS includes:
    - Kernel code and drivers, Java VM and classes,
    - JavaOS windows, graphics, and networking
  - Code for HotJava
  - Fonts of various types, sizes, and styles
- 4MB RAM
  - Assuming ROM can be executed in place
  - 2.5MB for JavaOS + Hotjava
  - 1.5MB for pages, applets, and images

JavaOS Advantages

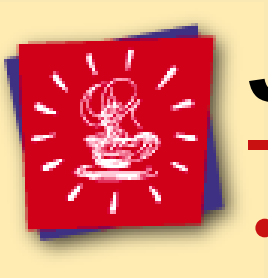
- Eliminates host OS overhead
  - Smaller size
  - No extraneous features
  - Faster startup
- ROMable
- Written in the Java language
  - Safe language
  - Portable
  - Dynamically extendable
- Cost of ownership
  - Ease of installation
  - Ease of administration

JavaOS for Intranet Computers

- Exploit enterprise network infrastructure
  - Ethernet at each desktop
  - Higher speed backbone networks
  - Level of system administration depends on choice of protocols like DHCP or RARP
- Supports Java™-based application environment
  - MIS applications easier to develop, deploy, maintain
- OEMs build JavaOS Intranet Computers

JavaOS for Internet Computers

- Use current or future home networking
  - High-speed modem
  - ISDN
  - Cable modem
- Requires an Internet Service Provider (ISP) to make it behave like an appliance
- Supports Java™-based application environment
  - Suitable for surfing the Web
- OEMs build JavaOS Internet Computers



JavaOS for Network Devices

- Tailoring Java to fit the device
  - No subsetting of:
    - Java VM and language and utility classes
  - Include or exclude based on hardware features:
    - AWT, Net, and file-related IO classes
  - Tune system for soft real time
    - VM and garbage collection
- JavaOS Development Environment
- Embedded JavaOS memory requirements:
  - 128K RAM, 512K ROM
  - Retains ability to download new Java code

JavaOS Status

- Currently runs on two platforms
  - SPARC
  - X86
- Being ported to other platforms
- Available in October 1996

JavaOS Demo

- Demo hardware
  - Prototype “network portal”
- Demo software
  - No host OS
  - Alpha JavaOS
  - Pre-beta HotJava
- Demo applets

JavaOS Future

- New Java API's
- Device Driver Interface
- Window system enhancements
  - Scalable fonts

Summary

- Features
- Performance
- Advantages
- Target systems
- Status

Deep Dive: JavaKernel

- MMU usage
  - Modified only at boot-time
  - Used to make memory ranges contiguous
- JavaOS runs in supervisor mode only
- Java™-compatible interrupt handlers were a little tricky
  - Low-level code uses interrupts-as-threads