From Parking Meters to Netbooks: Java™ Platform, Standard Edition 6 for ARM-Based Embedded Devices

Bob Vandette
Sun Microsystems, Inc.
Java SE Embedded Technical Lead
Introduction
Java SE 6.0 for Linux ARM Devices

> ARM Processors
> Example Embedded Devices Using ARM
> Challenges supporting ARM CPU's
> Java SE Embedded Enhancements
> Currently Supported Configurations
> Tips for Developing Java on Java SE Embedded
> Future Java SE Embedded Directions
ARM Processor

> ARM CPU History
  • First design in 1983.

> Why is ARM popular?
  • Low power, most popular mobile embedded CPU

> Multiple Vendors
  • ARM, Qualcomm, TI, Marvell, many more

> What's new in ARM processors
  • Higher Clock Speeds
  • Multi-CPU and Multi-Core designs coming
  • High Level of Integration with video, 3D, wireless, bluetooth, gps
Qualcomm Snapdragon Chipset

- Fully integrated solution including modem, Cortex-A8 ARMv7 applications processor, GPU, multimedia, peripherals
- CDMA and UMTS modems
- First Superscalar 1GHz application processor for mobile devices
- Advanced hardware accelerated codecs, 2D and 3D graphics
- Sub-500mW applications processor
Java on ARM History

**J2ME**

2001  CLDC – Connection Limited Device Configuration  
      Used primarily in Cell Phones

2002  CDC – Connected Device Configuration

**JAVA SE**

2007  Java 2 Standard Edition for Embedded 1.4.2  
      Linux ARM Headless

2009  Java Standard Edition for Embedded 6.0  
      Linux ARM Headless and Headful Configurations
Sampling of Embedded Applications using Java SE Embedded on ARM

Technical Challenges for Hotspot Support of ARM CPU's

> Several popular ARM architecture versions
  • ARMv5, ARMv6, ARMv7

> Many sub features to choose from
  • Thumb, Thumb2, Jazelle, RCT
    • Not available on all versions
  • Big Endian and Little Endian

> Floating Point Support
  • Optional (Requires Software Floating Point Support)
  • Different implementations (VFPv2, VFPv3)
Java SE for Embedded ARM Processor Support

> Goals

- 100% compatible with Desktop Java SE 6.0
- Broadest coverage with least number of binaries
- Take advantage of specific performance improvement features on each architecture

> Implementation

- Four different little endian binaries supporting the most common use cases

<table>
<thead>
<tr>
<th>armv5te</th>
<th>Software Floating Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>armv5te</td>
<td>VFP Hardware Floating Point</td>
</tr>
<tr>
<td>armv6</td>
<td>Software Floating Point</td>
</tr>
<tr>
<td>armv6</td>
<td>VFP hardware Floating Point</td>
</tr>
</tbody>
</table>

The armv6 binaries auto detect armv7 (Cortex-a8) and take advantage of v7 specific features
Java SE for Embedded ARM Processor Support

cont.

> armv5te specific support
  
  • atomic ops and Java level locking use mutexes after attempting a swap hashed on the address of the lock
  
  • Use literal pools for loading 32 bit pointers and constants with relocation support. The JIT allocates literal pool in jit'd method memory.

> armv6 support
  
  • Uses ldrex/strex for atomics and locking rather than slower armv5 support
  
  • Use rev instruction for performing byte swap

> armv7 (Cortex-A8)
  
  • All armv6 improvements plus, movw/movt instruction combinations are used for faster literal loading
Java SE Embedded Specific Features

> Headful and Headless builds
> Software Floating Point
> Low Memory Detection
> Power Conservation
> Gstreamer Audio Support
> Alternate UI and Window Manager Support
  • xfwm4*, Matchbox, ICEWM, metacity
> Browser, Plug-in Support for Mobile Device Browsers
  • Firefox*, Coolfox
> Tested Linux Distributions
  • Xandros*, Redflag Midinux, Fedora

* Full Certification Planned for First Release
The Linux community has been searching for solutions to handle low memory conditions.

- Killing processes is NOT the answer (SIGABORT, ouch!)
- We chose to support /dev/mem_notify
  - Linux kernel driver monitoring available memory

```
/dev/mem_notify avail?

YES
  create monitoring thread
  Open /dev/mem_notify

NO
  Feature Unavailable

Block waiting for input from device

Full GC

*Release additional VM memory (JIT buffers, etc)
```
Java SE Embedded Feature
Power Conservation

> **Goal**

- Reduce CPU usage to 0% when Java applications are idle in order to allow CPU clock to be reduced or suspended

> **Modifications Performed**

- Hotspot Virtual Machine
  - Eliminated all Polling Threads
- Java SE AWT/Swing Changes
  - Eliminated Polling for X Events when idle
Java Tips for Netbook devices

- Don't assume overlapping Windows
- Don't assume 640x480 resolution or higher
- Eliminate polling for I/O in your Java and Native code
- Only blink cursor when activity present
ThinkFree Java Application Experience
Optimizing ThinkFree Office Suite for Netbooks

> Netbook is ....
  • Lower in Price
  • Lower Power CPU
  • Less RAM and Disk Space
  • Lower Screen Resolution
  • Consumer Oriented Device
  • (Optionally) Touch Screen

Thanks to Geon Goo (accent@thinkfree.com) for input on ThinkFree optimizations
ThinkFree Java Application Experience
ThinkFree Mobile Viewer

> Optimized for Smaller Screen (Simplified UI)

> Faster Startup
  • Split Viewer functionality from Document Editor
    • Document Editor is one click away
  • Less Java Code Loading
  • Class Data Sharing Enabled helped by 25%
  • Used Native Splash Screen Support from SE 6

> Issue GC operations during idle times to reduce interruptions but eventually block to conserve power

> Reduced Cursor Polling by Decaying timer
Java SE Embedded on ARM Futures

- MVM Support
  - Multi Tasking Virtual Machine
- Multi-Processor ARM Support
- OpenGL ES Support
- Thumb2 Support Evaluation
- Java FX Graphics Acceleration
Multi-Tasking Virtual Machine
memory savings

Multiple Java Processes

<table>
<thead>
<tr>
<th>Java App 1</th>
<th>Java App 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM Native Libs</td>
<td>VM Native Libs</td>
</tr>
<tr>
<td>static VM data malloc'd data</td>
<td>static VM data malloc'd data</td>
</tr>
<tr>
<td>Java Heap</td>
<td>Java Heap</td>
</tr>
<tr>
<td>JIT Buffer</td>
<td>JIT Buffer</td>
</tr>
<tr>
<td>Class Data</td>
<td>Class Data</td>
</tr>
<tr>
<td>Linux Process 1</td>
<td>Linux Process 2</td>
</tr>
</tbody>
</table>

MVM Java Processes

<table>
<thead>
<tr>
<th>Java App 1</th>
<th>Java App 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM &amp; Native Libs</td>
<td>VM &amp; Native Libs</td>
</tr>
<tr>
<td>static VM data malloc'd data</td>
<td>static VM data malloc'd data</td>
</tr>
<tr>
<td>Java Heap</td>
<td>Java Heap</td>
</tr>
<tr>
<td>JIT Buffer</td>
<td>JIT Buffer</td>
</tr>
<tr>
<td>Class Data</td>
<td>Class Data</td>
</tr>
</tbody>
</table>

Sharing

- None
- Some
- All

Single Linux Process
Demonstrations

> Linux ARM Netbook
  - Full Java SE 6.0 Compatibility on ARM Processor
    - Plug-In Support Launching FX Application
    - Java Office Productivity Application (ThinkFree)

> Multi Tasking Virtual Machine
  - Startup Improvements
  - Memory Savings
Web addresses for more information on Java SE for Embedded

Java SE Embedded home page:  
http://java.sun.com/javase/embedded/index.jsp

Java SE Embedded downloads page:  
http://java.sun.com/javase/downloads/embedded.jsp

Java SE Embedded system requirements page:  
http://java.sun.com/javase/embedded/system_requirements.jsp

Java SE Embedded customers page:  
http://java.sun.com/javase/embedded/customers.jsp
Bob Vandette
bob.vandette@sun.com
javase-embedded-interest@sun.com