JavaOne™

JavaOne™

Getting up to Speed on Oracle Java SE Embedded Performance: Tuning Tips and Tricks

Carlos Lucasius
Software Developer

TUT10155

MAKE THE FUTURE JAVA

ORACLE®

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Program Agenda

- Java SE Embedded Overview
- Java Performance Overview
- Java Embedded Performance Optimizations
- Java Monitoring, Profiling, Tuning
- Demo

Program Agenda

- Inter-System Communication Performance

Java SE Embedded Overview

Motivation for Java Embedded

- The Internet of Things
- Ecosystem with Embedded devices at the Edge
- Huge platform diversity across Embedded devices.
  - Java a must
- Java Performance is key!

Java SE Embedded Overview

Java SE optimized for embedded use

- Java SE implementation optimized for deployment on high-end embedded devices:
  - Memory optimizations
  - Runtime optimizations
  - Power optimizations
- SE compliant: fully implements Java SE specification
  - Promotes application portability across different Java SE platforms
- Unified codebase for Java SE and Java SE Embedded
- Multi-core enabled

Java SE Embedded Overview

Multi-core

- Clock rates have hit a plateau
- Moore’s transistor count now needs more cores, not faster ones
- For *p* processors each with *c* cores each with *h* hardware threads

  we have a total of:

  $p \cdot c \cdot h$ hardware threads, also referred to as virtual processors

Java SE Embedded Overview

JVM request #virtual processors: Runtime.availableProcessors()

```
int os::active_processor_count() {
  int online_cpus = sysconf(_SC_NPROCESSORS_ONLN);
  pid_t pid = getpid();
  psetid_t pset = PS_NONE;
  // Are we running in a processor set?
  if (pset_bind(PS_QUERY, P_PID, pid, &pset) == 0) {
    if (pset != PS_NONE) {
      uint_t pset_cpus;
      // Query number of cpus in processor set
      if (pset_info(pset, NULL, &pset_cpus, NULL) == 0) {
    assert(pset_cpus > 0 && pset_cpus <= online_cpus, "sanity check");
    _processors_online = pset_cpus;
    return pset_cpus;
      }
    }
  }
  // Otherwise return number of online cpus
  return online_cpus;
}
```

Java SE Embedded Overview

Currently Supported Platforms and Releases

Processor	Operating System	Headless or Headful	FPU	Java SE version
ARMv6/v7	Linux	Headful	VFP	JDK 7u6
ARMv5	Linux	Headless	Soft	6u34, 7u6
ARMv6/v7	Linux	Headless & Headful (v7)	VFP	6u34, 7u6
ARMv7 (Server JIT)	Linux	Headless	VFP	7u6
PowerPC e600 core	Linux	Headless	Classic HW FP	6u34, 7u6
PowerPC e500v2 core	Linux	Headless	Embedded FP	6u34, 7u6
x86	Linux	Headless	x86	6u34, 7u6

Quick introduction to Java SE Embedded

Example devices

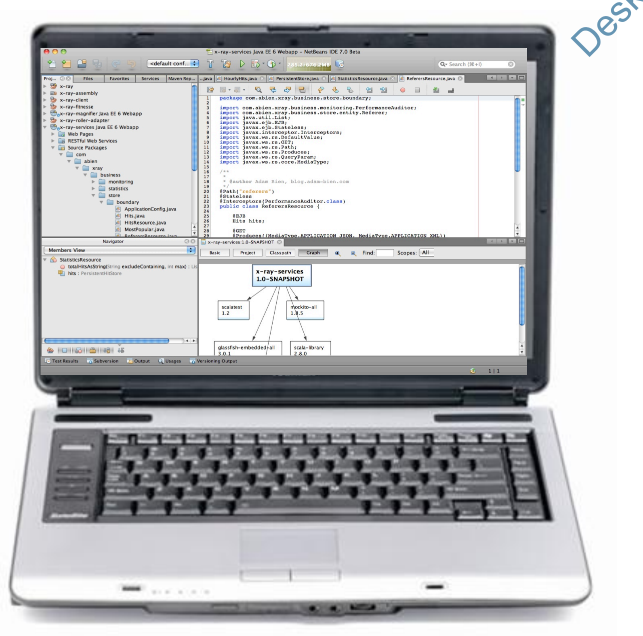
- ATMs
- Parking Meters
- POS Systems
- Lottery/Gaming Systems
- Multi Function Printers
- Intelligent Power Module
- Netbooks

- Routers & Switches
- Storage Appliances
- Network Management Systems
- Medical Imaging Systems
- Radar Systems
- Industrial PCs
- Factory Automation Systems
- Geo-Imaging Devices

- Smart Meters
- RFID Readers
- Video Conferencing Systems
- In-Flight Entertainment Systems
- Video Streaming Systems
- Electronic Voting Systems
- Voice Messaging Systems
- Security Systems

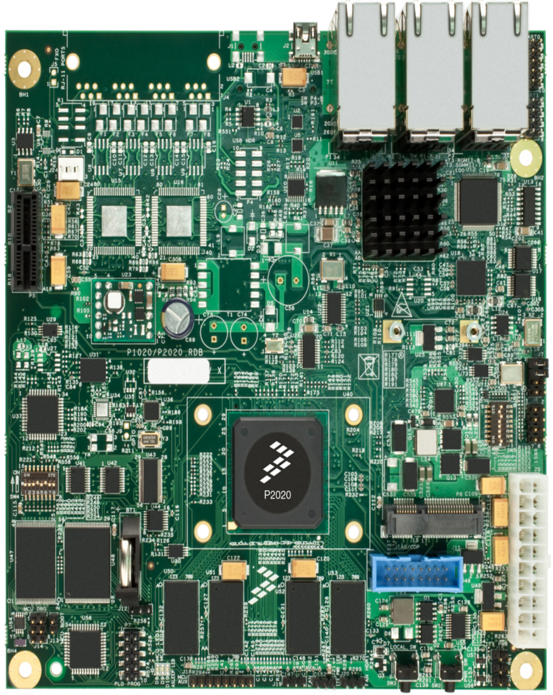
Java SE Embedded Overview

Development, initial troubleshooting



```java
package com.abien.xray.business.store.boundary;

import com.abien.xray.business.monitoring.PerformanceAuditor;
import com.abien.xray.business.store.entity.Referer;
import java.util.List;
import javax.ejb.EJB;
import javax.ejb.Stateless;
import javax.interceptor.Interceptors;
import javax.ws.rs.DefaultValue;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;

/**
 *
 * @author Adam Bien, blog.adam-bien.com
 */
@Path("referers")
@Stateless
@Interceptors(PerformanceAuditor.class)
public class ReferersResource {

    @EJB
    Hits hits;

    @GET
    @Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
```

Desktop

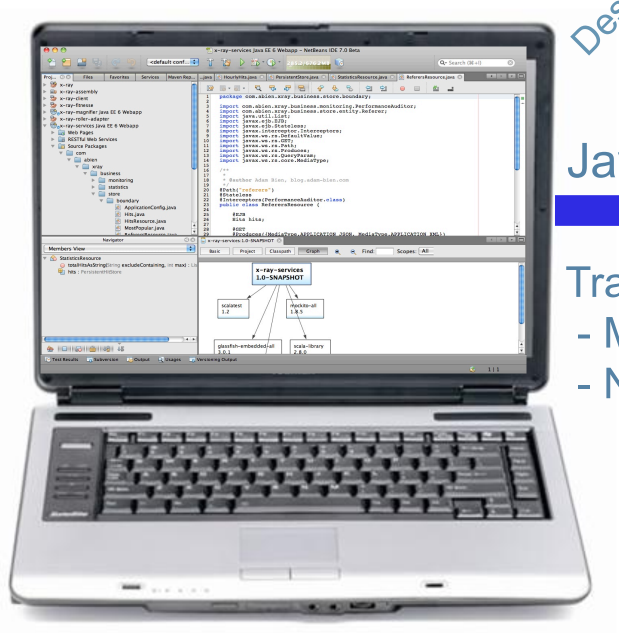


Board

Use Java SE (JDK) here to develop and initially test/troubleshoot embedded Java App

Java SE Embedded Overview

Transfer embedded Java App to embedded system



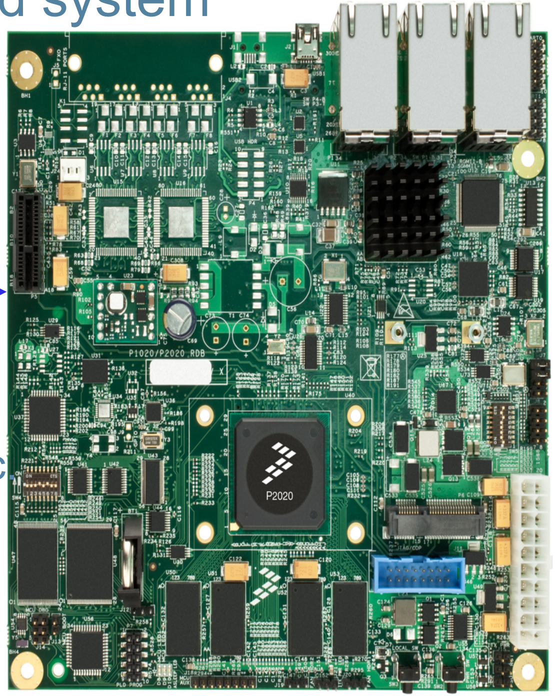
```
package com.abien.xray.business.store.boundary;

import com.abien.xray.business.monitoring.PerformanceAuditor;
import com.abien.xray.business.store.entity.Referer;
import java.util.List;
import javax.ejb.EJB;
import javax.ejb.Stateless;
import javax.interceptor.Interceptors;
import javax.ws.rs.DefaultValue;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;

/**
 *
 * @author Adam Bien, blog.adam-bien.com
 */
@Path("referers")
@Stateless
@Interceptors(PerformanceAuditor.class)
public class ReferersResource {

    @EJB
    Hits hits;

    @GET
    @Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
```

Desktop

Java App binary

Transport:

- Media: memstick, etc.
- Network: scp, sftp, etc.



Board

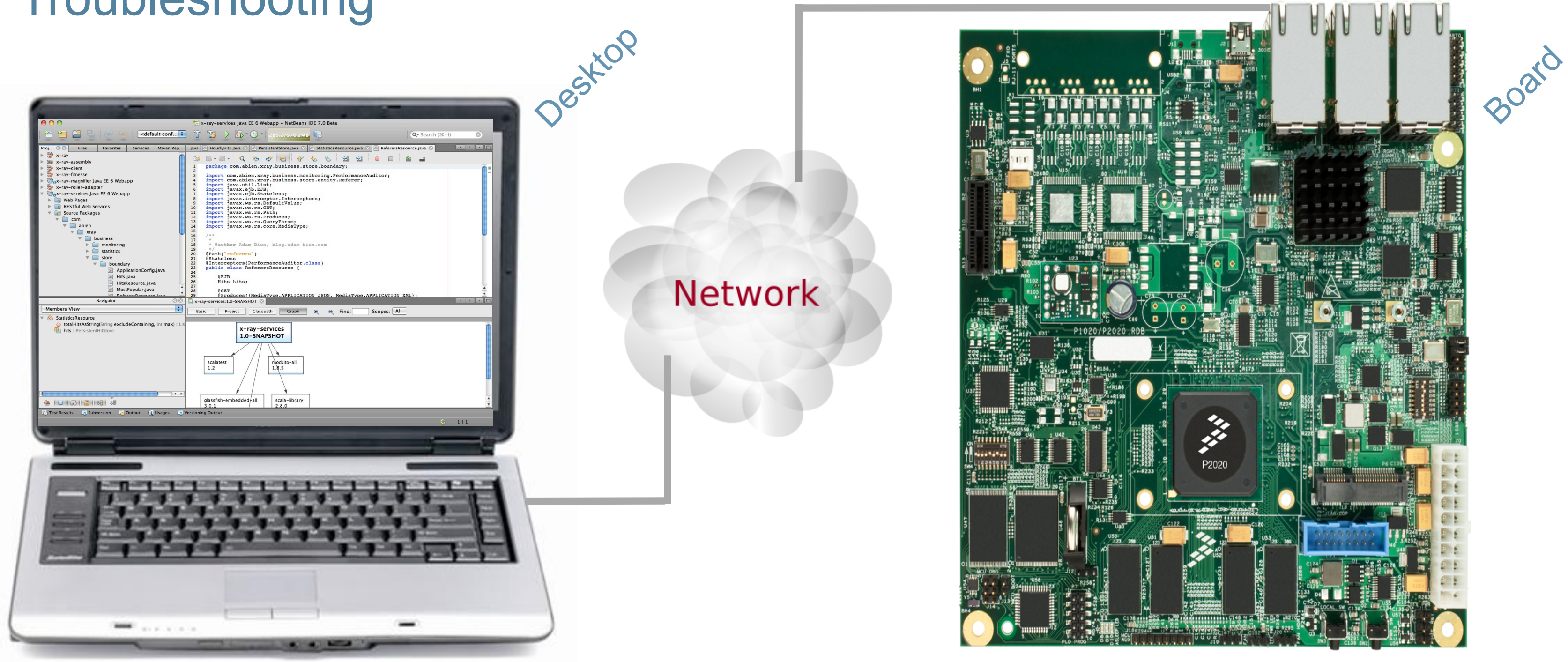
Java SE (JDK) here
version *X* update *Y*

same version / update level
promotes same bug fixes

Java SE-Embedded here
version *X* update *Y*

Java SE Embedded Overview

Troubleshooting



```java
package com.abien.xray.business.store.boundary;

import com.abien.xray.business.monitoring.PerformanceAuditor;
import com.abien.xray.business.store.entity.Referer;
import java.util.List;
import javax.ejb.EJB;
import javax.ejb.Stateless;
import javax.interceptor.Interceptors;
import javax.ws.rs.DefaultValue;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;

/**
 *
 * @author Adam Bien, blog.adam-bien.com
 */
@Path("referers")
@Stateless
@Interceptors(PerformanceAuditor.class)
public class ReferersResource {

    @EJB
    Hits hits;

    @GET
    @Produces({MediaType.APPLICATION_JSON, MediaType.APPLICATION_XML})
```

Java SE (JDK) runs here to remotely troubleshoot embedded Java App

Java SE Embedded runs here, embedded Java App on top of it

Java Performance Overview

Introduction

- Java performance involves much more than just Java: OS too
- Broad field, also many performance criteria (benchmarks)
- Java performance tuning is more of an Art than a Science
- Available Java SE performance tuning documentation basically also applies to Java SE Embedded performance tuning
  - Just situations often different: Embedded vs. Non-Embedded
  - Same command-line options that control performance
- Java Performance book: Charlie Hunt, Binu George

Java Performance Overview

Source vs Binary Performance Tuning

- Source:
  Improvements at this level continually being made by Oracle and OpenJDK developers community
- Binary:
  You have the binary, the JVM exposes a collection of settable parameters – command-line options – that allow developers to improve performance if feasible:
  standard / non-standard (-X) / developer options (-XX)

Java Performance Overview

Important factors

- OS level (Java can only be as good as underlying OS)
  - Numerous: CPU-related, I/O-related (disk, peripheral), MT-related, etc.
- Java level:
  - Runtime (Interpreter, Multi-Threading system)
  - Garbage Collection
  - JIT compilation
  - Class loading
  - Background threads: GC, JIT, profiling, stats collection, etc.

Java Embedded Performance Optimizations

Considerations

- Situation:
  Disk storage in device usually SD card, can be relatively slow
  - Minimize/avoid swapping, virtual memory use, logging
  - Filesystem choice can be a factor too
- Situation:
  RAM memory a key cost factor usually, hence minimization target
  - Most optional components in Java SE removed in Java SE Embedded
- Situation:
  Power consumption a cost factor usually, hence minimization target
  - Avoid –XX:+UsePerfData in deployment

Java Embedded Performance Optimizations

Considerations

- Situation:
  Loading of .class/.jar files slow, causes long start-up
  - Consider .jar file consolidation
  - Consider Class Data Sharing (CDS)
- Situation:
  Not taking advantage of inherent parallellism

Java Monitoring, Profiling, Tuning

Bottom-Up Approach: Proactive

- Starts at CPU Utilization
  - Utilize cycles as much as possible
  - Minimize cache misses
  - Maximize use of virtual processors: #hardware threads
  - More details: Java Performance, by Charlie Hunt and Binu John
  - Oracle collaborates with various major chip designers/manufacturers
- Works way up to OS, then Java platform, then Java application

Java Monitoring, Profiling, Tuning

Top-Down Approach: Reactive

- Monitor
  - Collect performance data
- Profile
  - Identify possible root causes
- Tune
  - Java level, Java platform level, OS level
    - Change source code: design, implementation
    - Change environment: set values for exposed parameters

Java Monitoring, Profiling, Tuning

Interfaces exposed by the JVM

- Java VM Tools Interface: JVMTI
- Java Monitoring and Management Interface: JMX
- Both interfaces fully implemented in Java SE Embedded
- An agent thread is typically started inside the Java VM process – the agent interacts directly with the Java VM interface and provides a front-end interface for tools, hence acts like a proxy
- Many agents networking-enabled, thus allowing remote troubleshooting

Java Monitoring, Profiling, Tuning

HPROF

- One of various tools available to generate heap dumps
- Heaps contain a wealth of useful information on Java execution, this applies at dump time: current state, some history
  - Great in helping find root causes of Java performance issues
    - Example: excessive GC caused by Java memory leak
- How to use: `java -agentlib:hprof[=option1,option2,...] Main`
  - Example: `java -agentlib:hprof=heap=dump,format=b`

Java Monitoring, Profiling, Tuning

NetBeans Profiler

- Powerful profiler, originates from Research Labs
- Implementation on ARM currently being developed and already accessible:
  - Already available NetBeans 7.3 dev version, check nightly builds: http://bits.netbeans.org/dev/nightly/

Java Monitoring, Profiling, Tuning

Java Management and Monitoring

- How to use:

```
java  -Dcom.sun.management.jmxremote.port=9999 \
        -Dcom.sun.management.jmxremote.ssl=false \
        -Dcom.sun.management.jmxremote.authenticate=false \
        Main
```

- com.sun.management.HotSpotDiagnosticMXBean supported

  Represents yet another way to dump heap on demand (HPROF-Format)

Demo

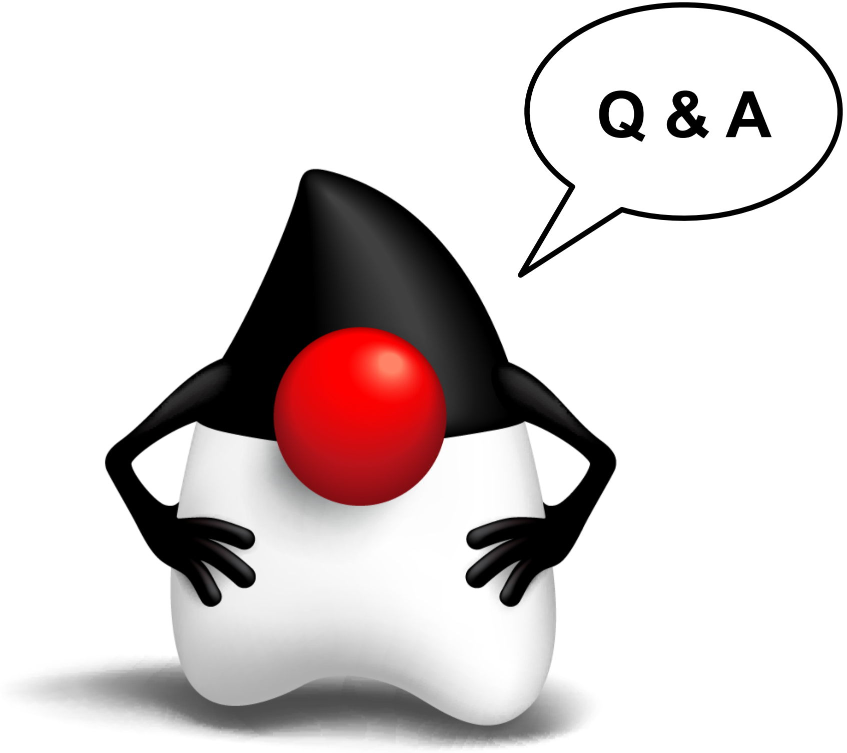
Java Remote Profiling on ARM

- HPROF
- NetBeans Profiler
- Management & Monitoring

Inter-System Communication Performance

SDP: Sockets Direct Protocol

- SDP provides high throughput, low latency
  - Example where useful: streaming applications
- Essentially by-passes TCP part of standard TCP/IP protocol
- Implemented in Java since JDK 7
- Transparent: no Java API changes needed to take advantage
- Developed to support stream connections over InfiniBand fabric
  - Useful in for example distributed filesystems, such as Hadoop



- Downloads page:
  - http://www.oracle.com/technetwork/java/embedded/downloads/javase/index.html
- Available for ARM, PPC, x86

JavaOne™