Java Environment for Parallel Realtime Development

Technology Vision

Dr. Fridtjof Siebert
CTO, aicas GmbH
Project Goal

- Provide a platform independent software development environment for
  - complex,
  - safe,
  - real-time,
  - multicore systems.
Project Work Structure

- Multi-Core Java Processor (JOP)
- Java/FPGA Interfacing

WP 2: Architecture Layer
Project Work Structure

- Parallel Partitioning RTOS

WP 2: Architecture Layer

WP 3: OS Layer
Project Work Structure

- Parallel Real-Time JVM
- Parallel Real-Time GC

WP 2: Architecture Layer
WP 3: OS Layer
WP 4: Virtual Machine Layer
Project Work Structure

- Multicore APIs
- Standardization

WP 5: API Layer
WP 4: Virtual Machine Layer
WP 3: OS Layer
WP 2: Architecture Layer
Project Work Structure

- Static Analysis
- Unit Testing

WP 6: Analysis Tools
WP 5: API Layer
WP 4: Virtual Machine Layer
WP 3: OS Layer
WP 2: Architecture Layer
Project Work Structure

- Radar Application
- SW Radio
- Aircraft Operational Communications

WP 2: Architecture Layer
WP 3: OS Layer
WP 4: Virtual Machine Layer
WP 5: API Layer
WP 6: Analysis Tools
WP 7: Validation
Real-Time Guarantees
Real-Time Guarantees

Blocking GC

cycle 1  cycle 2
Real-Time Guarantees

Blocking GC

- cycle 1
- cycle 2

Incremental GC

- [Bar graph showing incremental GC process]
Real-Time Guarantees

Blocking GC

- cycle 1
- cycle 2

Incremental GC

Concurrent GC

- CPU 1: Application
- CPU 2: GC
- CPU 3: Application
Real-Time Guarantees

- **Blocking GC**
  - cycle 1
  - cycle 2

- **Concurrent GC**
  - CPU 1: Application
  - CPU 2: GC
  - CPU 3: Application

- **Incremental GC**
  - cycle 1
  - cycle 2

- **Parallel GC**
  - CPU 1: cycl1 cycl2
  - CPU 2: cycl1 cycl2
  - CPU 3: cycl1 cycl2
Real-Time Guarantees

**Blocking GC**
- Cycle 1
- Cycle 2

**Concurrent GC**
- CPU 1: Application
- CPU 2: GC
- CPU 3: Application

**Incremental GC**

**Parallel GC**
- CPU 1 cycles 1 and 2
- CPU 2 cycles 1 and 2
- CPU 3 cycles 1 and 2

**Parallel & Concurrent GC**
- CPU 1: Application
- CPU 2: GC
- CPU 3: GC
Real-Time Guarantees

Blocking GC

- Cycle 1
- Cycle 2

Concurrent GC

- CPU 1: Application
- CPU 2: GC
- CPU 3: Application

Parallel & Concurrent GC

- CPU 1: Application
- CPU 2: GC
- CPU 3: GC

Incremental GC

- Cycle 1
- Cycle 2

Parallel GC

- CPU 1: Cycl1
- CPU 2: Cycl1
- CPU 3: Cycl1

Parallel Incremental GC

- CPU 1
- CPU 2
- CPU 3
Real-Time Guarantees

Blocking GC
- CPU 1: Application
- CPU 2: GC
- CPU 3: Application

Concurrent GC
- CPU 1: Application
- CPU 2: GC
- CPU 3: Application

Parallel & Concurrent GC
- CPU 1: Application
- CPU 2: GC
- CPU 3: GC

Incremental GC

Parallel GC
- CPU 1
- CPU 2
- CPU 3

Parallel & Incremental GC
- CPU 1
- CPU 2
- CPU 3
Real-Time Guarantees

Blocking GC

- cycle 1
- cycle 2

Concurrent GC

- CPU 1: Application
- CPU 2: GC
- CPU 3: Application

Parallel & Concurrent GC

- CPU 1: Application
- CPU 2: GC
- CPU 3: GC

Incremental GC

- cycle 1
- cycle 2

Parallel GC

- CPU 1 cycl1 cycl2
- CPU 2 cycl1 cycl2
- CPU 3 cycl1 cycl2

Parallel Incremental GC

- CPU 1 cycl1 cycl2
- CPU 2 cycl1 cycl2
- CPU 3 cycl1 cycl2

aicas
Real-Time Guarantees

Blocking GC

Concurrent GC

CPU 1: Application

CPU 2: GC

CPU 3: Application

Parallel & Concurrent GC

CPU 1: Application

CPU 2: GC

CPU 3: GC

Incremental GC

Parallel GC

CPU 1: cycl1

CPU 2: cycl1

CPU 3: cycl1

Parallel Incremental GC

CPU 1

CPU 2

CPU 3

Parallel GC

CPU 1: cycl2

CPU 2: cycl2

CPU 3: cycl2
Real-Time Guarantees

**Blocking GC**
- CPU 1: Application
- CPU 2: GC
- CPU 3: Application

**Concurrent GC**
- CPU 1: Application
- CPU 2: GC
- CPU 3: Application

**Parallel & Concurrent GC**
- CPU 1: Application
- CPU 2: GC
- CPU 3: GC

**Incremental GC**
- CPU 1: Application
- CPU 2: GC
- CPU 3: Application

**Parallel GC**
- CPU 1: cycl1
- CPU 2: cycl1
- CPU 3: cycl1

**Parallel Incremental GC**
- CPU 1
- CPU 2
- CPU 3
Scalability

- Theoretical Limits

Utilization

CPUs

- ideal
- check
- compress
- jess
- raytrace
- db
- javac
- mpegaudio
- mtrt
- jack
- non-parallel
Minimal Added Complexity

- new APIs, e.g.:
  - CPU affinity:

    ```
    AffinitySet.setProcessorAffinity(
        affinitySet, thread);
    ```
Find Errors

- Classical Race Conditions
  - **Thread 1**
    - `obj.i++;`
  - **Thread 2**
    - `obj.i++;`
Find Errors

- Classical Race Conditions
  - Thread 1
    obj.i++;
  - Thread 2
    obj.i++;

Static Analysis
Conclusion

- JEOPARD enables
  - real-time applications to profit from
  - upcoming multi-core architectures

- Technology now part of aicas' products
AICAS CASSIDIAN- SYSGO Webinar
December 8th, 2010

Multi-core and Virtualization:
A Safe & Secure Solution with PikeOS
Agenda

- Company Overview
- PikeOS concept
  - Safe and Secure Virtualization
  - Multi-core support
  - General benefits
- PikeOS and real-time Java
  - Aicas’ Jamaica integration
- References
  - CASSIDIAN project
SYSGO Company Overview

- A software technology leader addressing a global market
- Founded in 1991
- 90+ employees
  - 15% growth in 2008, 20% growth in 2009
- Continuous product revenue growth since 2001
- Offices in Germany (Mainz, Ulm, Rostock), France (Paris), The Czech Republic (Prag) and North America (Chicago)
- Distributors in Japan, Korea, Italy, UK, Austria, India, ...
Safe & Secure Virtualization RTOS

- Designed from ground-up for safety & security
  - Modularity and compactness
  - MILS compliant architecture
- Virtualization for embedded/real-time
  - By design (no hypervisor add-on)
  - From ARINC-653 to full featured Linux
- Scalable and flexible
  - Can be used just as small and fast RTOS
  - Widest range of supported personalities in the market (Posix, Arinc-653, Linux, RTEMS, Ada, RTJava,...)
- Hardware independent
  - x86, PPC, ARM, MIPS, SPARC/LEON, SH-4...
  - Not depending on hardware virtualization
- Certification for safety & security
  - DO-178B, EN50128, IEC61508, EAL,...
Modular Virtualization Platform

- Guest Operating System
- Guest Runtime Environment
- PikeOS Native

PikeOS System Software

PikeOS Separation Microkernel
- Architecture Support Package (ASP)
- Platform Support Package (PSP)

Hardware

User Mode
Kernel Mode

up to 62 partitions
Resource Partitioning

- Static allocation of all system resources
- Application has guaranteed access to assigned resources
- Applications cannot access resources of other partitions if not explicitly configured otherwise
- No error propagation throughout other partitions
- Memory protection enforcement using Hardware (MMU)
- All partitions execute in user mode
Time Partitioning

- Static configuration of execution order and duration of partitions
- Deterministic hard real-time operation

Advanced features*

- Multiple resource partitions can be assigned to one time partition
- Best possible CPU usage
  - Dedicated threats can be scheduled whenever the current time partition becomes idle
- Shortest response time
  - Optional: Dedicated threads can have superior priority to any normal partition (if used this will violate strict determinism and might have impact on hard real time behavior of normal partitions)

* using PikeOS’ patented scheduler
PikeOS Multi-Core Support

- Asymmetric Multi Processing (AMP)
  - Adaptations are limited to the PikeOS PSP
  - PikeOS can coexist with other operating systems or executives
  - Multiple instances of PikeOS can run in parallel

- Symmetric Multi Core Support (SMP)
  - Cores are allocated to resource partitions
  - Cores can be shared between partitions
  - One partition can have multiple cores assigned
Safe & Secure Virtualization Benefits

- On a single device the software virtualization platform can ensure certifiable coexistence of any:
  - safety critical applications
  - non critical applications
  - hard real-time applications
  - non real-time applications

- Modular certification:
  - applications can be certified partition wise
  - no need to certify a non-critical application just because it runs on the same hardware

- Future proof long-term solution:
  - code can be reused as standard API are available
  - implementation can be vastly independent of hardware

- Centralized fault management (Health Monitoring)
PikeOS Benefits

- **Reduce costs**
  - Decrease hardware deployment costs
    - Reduce number of computers to be used
    - Optimize CPU use
  - Decrease development costs
    - Allow legacy code reuse
    - Special tools to manage time & space partitioning
  - Decrease certification costs
    - Very modular architecture
    - ASP and PSP concept

- **Secure investment**
  - Increase hardware independence
    - Kernel implementation does not depend on specific hardware
  - Facilitates new execution environment support
    - Personality concept allows easy port to new software/middleware platform
JEOPARD highlights

- SMP built on top of safe and secure POSIX
  - Certifiable POSIX based on core PSE52
  - UDP certifiable stack and certifiable file system
- Jamaica support through POSIX SMP Personality
  - Especially designed and implemented for JEOPARD
- Java threads vs PikeOS threads
  - 1 PikeOS thread for n POSIX threads
  - 1 POSIX thread for 1 Java thread
- Multi-Core management
  - Static assignment for PikeOS threads
  - Load balancing for POSIX /Java threads
- Determinism vs performance
  - Load balancing doesn’t impact real-time behavior
  - Good level of performance despite hard real-time support
JEOPARD industrial applications

- Project objectives
  - System software designed for developing predictable high performance embedded systems that exploit the power of new multiprocessor platforms and higher level programming languages.

- Avionics Application – GMV Skysoft
  - AOC (Airline Operational Centre): IMA application, compliant to ARINC 653 and developed according to DO-178B/EC-12B Level C. It communicates with a number of on-board & ground systems, like Flight Control, Navigation, Cockpit Display, AGP.

- SDR Application - RadioLabs
  - Software Defined Radio application: designed in order to simulate the operations performed in a UMTS system at the physical layer.

- Radar Application – CASSIDIAN (formely EADS Defense & Security)
  - Radar application: based on performance requirements typically related to Airport Surveillance Radar (ASR). The ASR digital subsystem is called a Radar Processor (RAP).
SYSGO’s users
Java Environment for Parallel Real-time Development

Radar Use Case
Cassidian

- Formerly EADS Defence & Security
- Revenues of € 5.4 billion in 2009
- 21000 employees
Radar use case

- 2 Demonstrators: jRSP and jTracker
  - jRSP requires high processing and data throughput
  - jTracker handles a huge amount of objects
- Jeopard tools fulfill most of the radar use case evaluation criteria
- jRSP was demonstrated at ICT 2010 in Brussels
- jRSP uses all the Jeopard runtime tools
Jeopard Radar Signal Processor

Radar Scenario Generator

Radar Signal Processor

Plan Position Indicator (DISPLAY)
Jeopard Radar Signal Processor
Jeopard Radar Signal Processor

uc Use Cases Evaluation jRSP

- «system» jRSP
  - receive ADC data
  - detect and output targets
  - perform high resolution spectrum estimation (HRSE) at selected position
  - provide HRSE results to LCD Display

User

- Radar Scenario Generator

- PPI Display
  - reception picture
  - HRSE position

- LCD Display

User interaction:
- User provides radar picture, HRSE position
- PPI Display receives and displays radar picture and HRSE results
- LCD Display receives and displays HRSE results

Cassidian
Jeopard Radar Signal Processor

- Pulse Compression (PC)
- Spectrum Estimation (SE)
- Detection (DET)
- CONTROL
- Jamaica VM / OpenJDK
- PikeOS/Linux
- 6 Core CPU
- PC Mainboard
- FPGA
- High Resolution Spectrum Estimation (HRSE)
- LCD Display Control
- JOP
- Show HRSE results
- HW Methods
## Evaluation criteria (1)

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Expected Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV001 Application scalability over all cores</td>
<td>All processing cores have about the same load and processing time inversely proportional to the number of cores</td>
</tr>
<tr>
<td>EV002 Minimum input sample rate or input data rate with application running</td>
<td>2.5 MHz 10MB/s</td>
</tr>
<tr>
<td>EV003 Minimum processing throughput without data loss</td>
<td>60000 detections per scan 600 tracks per scan</td>
</tr>
<tr>
<td>EV004 CPU idle time</td>
<td>More than 25%</td>
</tr>
<tr>
<td>EV005 HWMethods overhead (latency)</td>
<td>Less than 0.5ms</td>
</tr>
</tbody>
</table>
### Radar use case

#### Evaluation criteria (2)

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Expected Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV006 Ethernet maximum throughput (TCP) without application running</td>
<td>More than 50 MB/s</td>
</tr>
<tr>
<td>EV007 UDP Package loss</td>
<td>None for data rates lower than 15MB/s</td>
</tr>
<tr>
<td>EV008 Overall application latency</td>
<td>Less than 150ms</td>
</tr>
</tbody>
</table>
Radar use case

Pulse compression 4 threads

- jRSP_nRT average
- jRSP_nRT min
- jRSP_nRT max
- jRSP_RT average
- jRSP_RT min
- jRSP_RT max

Batch number (1 batch = 1000 executions)
Radar use case

What the Jeopard tools have achieved

- Show the strength of real-time Java on parallel architectures
  - Predictability
  - Good scalability
  - Data processing performance
- Application overall latency can be better with the Jeopard tools than with standard Linux and standard Java. (jRSP)
- Application overall execution performance with PikeOS and Jamaica can be very similar to performance on Linux and OpenJDK (jTracker)
- Add FPGA power to Java (HWMethods) with minimal added latency
- Java-Java information sharing within heterogeneous systems (JOP on FPGA)