Xenomai 3 – An Overview of the Real-Time Framework for Linux

Agenda

Project introduction

Co-Kernel technology, now and then

Xenomai 3 for native Linux

Improving co-kernel integration

Summary
What is Xenomai?

- Old-style real-time extension for Linux?
- Something like / a fork of RTAI?
- Requires real-time applications to be kernel modules?
- ...?
Xenomai is an RTOS-to-Linux Portability Framework

It now comes in two flavors

• As co-kernel extension for (patched) Linux
• As libraries for native Linux (including PREEMPT-RT)
Xenomai History

**Xenomai 1.0**
- Announced in 2001 – as portability framework for RTOS applications
- Required a real-time basis
- Development of ADEOS layer for Linux and RTAI
- Merged with RTAI => RTAI/fusion

**Xenomai 2.0**
- Departed from RTAI in 2005 – incompatible design goals
- Evolved ADEOS to I-pipe layer (also used by RTAI)
- Ported to 6 architectures

**Xenomai 3.0**
- Released in 2015 after >5 years of development
- Rework of in-kernel core (now POSIX-centric)
- Support for native Linux
### People behind Xenomai

<table>
<thead>
<tr>
<th>Name</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Philippe Gerum</td>
<td>Project founder and maintainer</td>
</tr>
<tr>
<td>Gilles Chanteperdrix</td>
<td>ARM, x86 archs, Xenomai 2 &amp; 3 core, RTnet</td>
</tr>
<tr>
<td>Alexis Berlemont</td>
<td>Analogy stack</td>
</tr>
<tr>
<td>Jorge Ramirez-Ortiz</td>
<td>Analogy stack</td>
</tr>
<tr>
<td>Wolfgang Grandegger</td>
<td>Real-time CAN</td>
</tr>
<tr>
<td>Jan Kiszka</td>
<td>RTDM, x86 arch, assorted</td>
</tr>
<tr>
<td>et al.</td>
<td></td>
</tr>
</tbody>
</table>
Xenomai Applications

- Machine control systems, PLCs
- Printing machines (manroland)
- Printers / copying machines
- Network switches (e.g. Ruggedcom)
- Magnetic resonance tomographs (Siemens Healthcare)
- OROCOS (OSS robotics framework)
- Robotic research projects
- … (many, many incognito applications)
Xenomai 3 – An Overview of the Real-Time Framework for Linux

Agenda

Project introduction

Co-Kernel technology, now and then

Xenomai 3 for native Linux

Improving co-kernel integration

Summary
What is a Co-Kernel?

- Kernel
- Hardware
- Scheduler A
  - Preemption
  - Scheduler B
- Dispatching and Collaboration Services
- Hardware
  - IRQ
Interrupt and Trap Dispatching – from Adeos to I-pipe Core

I-pipe Core

Per-CPU Adeos Pipeline

Interrupts & Traps

Highest Priority Domain X

Root Domain

Lowest Priority Domain Y

Linux Kernel
Xenomai 2 Co-Kernel Architecture

- Xenomai Nucleus
- POSIX Skin
- VxWorks Skin
- Native Skin
- RTDM

Syscalls

- Stub Library POSIX
- Stub Library VxWorks
- Stub Library Native
- Stub Library RTDM

Drivers

Kernel

User Space
Cobalt: Real-Time Co-Kernel for Linux

Device drivers (RTDM-based)

Cobalt core

libcobalt (POSIX subset + extensions)

copperplate interface

Applications

Kernel

User

non-POSIX real-time APIs
Cobalt's Application Interface

Dispatching problem
- Both Cobalt and libc provide POSIX implementations
- How do RT application pick the right one?

Solution: symbol wrapping
- Example pthread_mutex_lock → __wrap_pthread_mutex_lock
- libcobalt provides __wrap_* , forwards unhandled invocations to libc
- No source code changes to POSIX applications required
- Some additional services available (*_np)

Supported architectures
- ARM (32 bit, 64 bit upcoming)
- Blackfin
- PowerPC (32 bit, 64 bit)
- x86 (32 bit, 64 bit, 32-on-64 bit, x32)
Migrating Threads between Cobalt and Linux

Preserve Linux service for Cobalt threads

- Linux syscalls
- Fault and trap handling
- Handling of asynchronous signals

Solution: every cobalt thread is also a Linux task

- Share thread states
- Only one can run at a time
- Migration to RT: suspend Linux task, resume Cobalt thread
- Migration to Linux (on syscall, fault/trap, signal): suspend Cobalt thread, resume Linux task
Real-Time Driver Model (RTDM)

Goals and principles

• Provide environment for co-kernel real-time drivers
  • Service interface towards applications and other drivers
  • Low-level primitives from implementing drivers
• Reuse Linux for non-RT purposes
  (setup / shutdown, resource discovery and claiming, etc.)

Two types of RTDM devices

• Character device (open/close, read, write, ioctl)
• Protocol device (socket, bind, send, recv, etc.)

Device profiles

• Character: UART, UDD (analogous to UIO), Memory, ...
• Protocol: UDP/TCP (RTnet), CAN, IPC, ...
Tooling with Cobalt

Debugging
• gdb works
• Improvements on synchronous stop/resume are work in progress

Tracing
• ftrace (tracecmd & Co.)
• I-pipe latency tracer (low-level latency hunts)

Valgrind / Helgrind
• No support because of unknown syscalls
• Alternative: Mercury (native support)
• Limited suitability for RT applications in general
Hardening Your RT Application with Cobalt

Cobalt fosters clear RT/non-RT split
• RT = everything that runs against cobalt, non-RT = all the rest
• Migrations can trigger debug signal

SIGDEBUG (SIGXCPU)
• Usage: enable when RT thread enters time-critical phase
• Signal reasons
  • SIGDEBUG_MIGRATE_SIGNAL (Linux signal pending)
  • SIGDEBUG_MIGRATE_SYSCALL (Linux syscall invoked)
  • SIGDEBUG_MIGRATE_FAULT (page fault etc. triggered)
  • SIGDEBUG_MIGRATE_PRIOINV (RT thread waits for migrated thread)
  • SIGDEBUG_WATCHDOG (RT thread starves Linux)
• Instrumentation of lazily migrating malloc/free
Xenomai 3 – An Overview of the Real-Time Framework for Linux

Agenda

- Project introduction
- Co-Kernel technology, now and then
- Xenomai 3 for native Linux
  - Improving co-kernel integration
- Summary
Mercury: RTOS API Emulation for Native Linux

- Device drivers (RTDM-based)
- native RTDM
- Glibc
- copperplate interface
- non-POSIX real-time APIs
- Applications
- User

Kernel
Mercury Details

Vision and goals

• Run API emulation over standard Linux/POSIX
• Enable seamless migration between co-kernel and native Linux deployments

Status

• 3 real-time APIs available
  • VxWorks
  • pSOS
  • Alchemy (former “native skin”)
• Native RTDM layer yet missing
  • Preexisting work by Wolfgang Grandegger, need update
  • Shall enable usage of all RTDM drivers under Linux (RTnet, Analogy, ...)

© Siemens AG 2015, et al. Creative Commons BY-SA 3.0 license.
Do We Still Need a Co-Kernel?

Functional limitations of Mercury

• Emulation of RTOS scheduling behavior limited by Linux scheduler
• Not all kernel+libc code paths used by Mercury are necessarily hard real-time under PREEMPT-RT
• Application use of non-RT services harder to identify

Performance limitations of Mercury / PREEMPT-RT

• Co-kernel usually more light-weight on low-end platforms (limited caches vs. code path lengths)
• PREEMPT-RT can have unwanted impact on co-located non-RT workloads
Xenomai 3 – An Overview of the Real-Time Framework for Linux

Agenda

- Project introduction
- Co-Kernel technology, now and then
- Xenomai 3 for native Linux
- Improving co-kernel integration
- Summary
The Dark Side of the Co-Kernel: Patch Maintenance

Limited availability of patches
- Current (release 3.0.2) support available for 3.10.32, 3.14.44, 3.18.20, 4.1.18
- Patches usually do not target latest stable
- Self-made updates (e.g. 4.1.18 → .20) often feasible but not broadly tested
- And then there are “nice” vendor trees...

Changes to critical subsystems regularly cause regressions
- Subtle breakages in IRQs, syscalls, memory management possible
- New kernel features have incompatible side effects

Porting efforts consume core developer resources
- Most work done by Philippe and Gilles so far
- Time would be better spent on feature improvements...
Project “Dovetail”

Goals
• Reduce maintenance efforts of co-kernel
• Make hooks/extension separate, more upstream palatable features

Main elements
• IRQ pipeline
• Co-kernel extensions
  • Scheduler transitions for tasks
  • Sharing of CPU traps (incl. Syscalls)
  • Collaborative task management (affinity, context switch, signals, exit)
  • Process memory pinning (eagerly spread ioremap/vmalloc mappings)
  • IRQ muting
  • …
• Extended use of kernel infrastructure for Xenomai core
Xenomai 3 – An Overview of the Real-Time Framework for Linux

Agenda

Project introduction
Co-Kernel technology, now and then
Xenomai 3 for native Linux
Improving co-kernel integration

Summary
Summary

- **Xenomai adds value to Linux**
  - Portability framework from classic RTOS's to Linux
  - Co-kernel approach can be beneficial for low latencies and real-time application architecture

- **Version 3 renovates and expands Xenomai**
  - Support for RTOS API emulation on top of native Linux & PREEMPT-RT
  - New architecture simplifies and improves co-kernel support

- **“Dovetail” aims at easing co-kernel maintenance**
  - Clearer feature separation
  - Better integration with Linux infrastructure
  - Propose for upstream merge???
Any Questions?

Thank you!

http://xenomai.org

Jan Kiszka <jan.kiszka@siemens.com>
Xenomai mailing list <xenomai@xenomai.org>
<table>
<thead>
<tr>
<th><strong>Glossary</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cobalt</strong></td>
<td>Co-kernel variant of Xenomai 3</td>
</tr>
<tr>
<td><strong>Mercury</strong></td>
<td>Native Linux variant of Xenomai 3</td>
</tr>
<tr>
<td><strong>Alchemy</strong></td>
<td>Xenomai-own real-time API</td>
</tr>
<tr>
<td><strong>Copperplate</strong></td>
<td>Library layer for building RTOS APIs</td>
</tr>
<tr>
<td><strong>Boilerplate</strong></td>
<td>Internal utility Library</td>
</tr>
<tr>
<td><strong>Trank</strong></td>
<td>Library to support porting from Xenomai 2 to 3</td>
</tr>
<tr>
<td><strong>RTDM</strong></td>
<td>Real-Time Driver Model, kernel API that enables</td>
</tr>
<tr>
<td></td>
<td>RT drivers, specifically for Cobalt</td>
</tr>
<tr>
<td><strong>Analogy</strong></td>
<td>RTDM drivers for digital/analogue converters</td>
</tr>
<tr>
<td><strong>Adeos</strong></td>
<td>Original interrupt pipeline for Linux,</td>
</tr>
<tr>
<td></td>
<td>used by early Xenomai 2 versions</td>
</tr>
<tr>
<td><strong>I-pipe</strong></td>
<td>Evolution and simplification of Adeos</td>
</tr>
<tr>
<td><strong>Dovetail</strong></td>
<td>New architecture of Linux extensions</td>
</tr>
<tr>
<td></td>
<td>to hook Xenomai 3 into Linux</td>
</tr>
</tbody>
</table>