Evaluation of real-time operating systems for use in Integrated Modular Avionics

Professor: Dr. Martin Bogdan, Universität Leipzig
Tutor: Thomas Schanne, EADS Deutschland GmbH
Author: Martin Christian

Introduction
Structure

Structure:

1. *Introduction*

   Motivation, problem

2. *Requirements*

   Kernel requirements in Integrated Modular Avionic

3. *Analyses*
   - Linux: Free UNIX for PC
   - Xen: Hypervisor for para-virtualised guest OS
   - Minix 3: µKernel + OS following the „TV model"
   - L4: µKernel providing space, activity and communication abstraction

4. *Implementation*
   - L4 implementations + Linux ports on L4 µ-kernel
   - Implementation steps

5. *Evaluation*

   What's the result? Does it meet the requirements?

Introduction
Problem

Motivation:

- Linux takes hold of the embedded systems market [heise, 2003]
- Linux is used in a plane by Boeing [heise, 2006]
- All real-time OS can't be evaluated within a Diplomarbeit
- EADS Deutschland GmbH provides the development board of the laser range radar project *Hellas*

Problem:

1. Find the most reasonable way to use Linux in Avionics
2. Port Linux to the Hellas-board this way

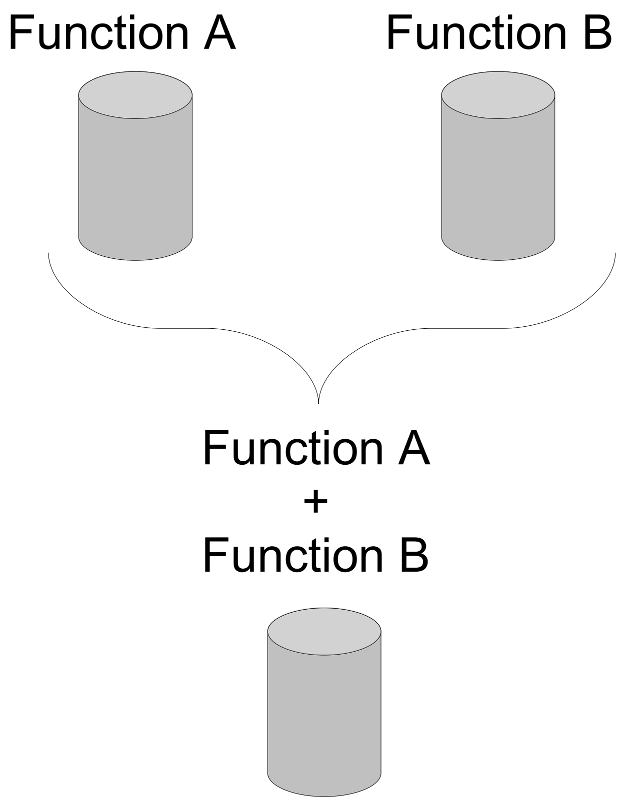
Introduction
Terms

- ***Kernel***: „*[...] is used to denote the part of the operating system that is mandatory and common to all other software.*“ [Liedke, 1995]
  - Monolithic kernel: Scheduling, interrupt handling, memory management and device drivers are part of the kernel
  - Microkernel: „[...] a concept is tolerated inside the µ-kernel only if moving it outside the kernel [...] would prevent the implementation of the system's required functionality“ [Liedke, 1995]

- **Real-Time**: „*A real-time system responds in a (timely) predictable way to all individual unpredictable external stimuli arrivals.*“ [Timmerman+, 2005]
  - Soft Real-Time: Time constraints have to be met on average
  - Hard Real-Time: Time constraints have to be met always

Requirements
IMA

Avionic: All electronic devices in aviation.



<u>Goals of Integrated Modular Avionics (IMA):</u>

- *Functionality*: More functionality in less space
- *Safety*: Easy handling, reconfiguration on hardware errors
- *Costs*: Modular architecture cut costs in development and maintenance

Requirements
IMA-Criteria

Requirements for an IMA-Kernel:

- ***Real-Time:***

  The kernel must meet the real-time requirements of the most demanding application running on top. → hard real-time

- ***Partitioning:***

  “The behaviour and performance of software in one partition must be unaffected by the software in other partitions.“ [Rushby, 1999]

  - Space: Partitions must not manipulate data within each other → neither in memory nor on devices
  - Time: Partitions must not steal time from each other

Requirements
IMA-Criteria

Requirements (continued):

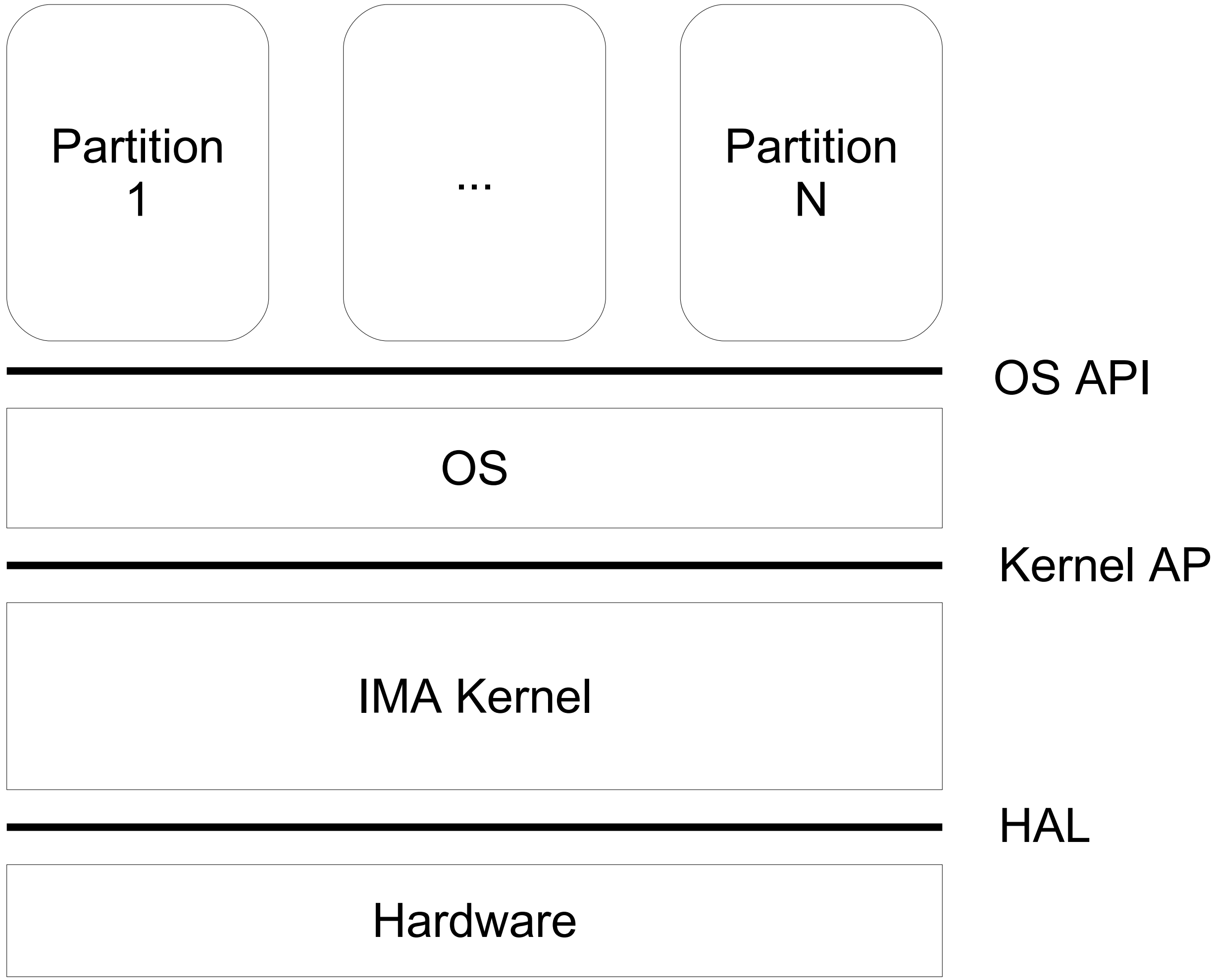
- **Trusted Computing Base (TCB):**
  - Minimal TCB → easier certification
  - Less code → less bugs [Herder+, 2006]

- **Open standards:**

  Independence from manufacturer ensure availability of components

- **Modularity:**
  - Re-use of components → less development costs
  - Exchangeable components → less storage costs (product cycle > 10 years)

Requirements
IMA-Model



Generic IMA model according to [Bennett, 2003]

Analyses
Outline

Limitations:

- Open Source:
  - Easy to obtain
  - No problems with NDAs
  - Assured source code availability for project duration
  - Problems with commercial developers: Acquisition, bankrupt
- Goal-oriented selection:
  - Only kernel with Linux available
- Pre-selection in [Bennett, 2003]:
  - Best choice to start kernel development for IMA is L4
  - Many projects have evolved further → second glance worthwhile

Analyses
Outline

Excluded kernel (selection):

- Mach: 1st generation µ-kernel, [Bennett, 2003]
- C5: Predecessor of Chord OS, [Bennett, 2003]
- RTEMS: Single address space OS (no partitioning)
- MicroC/OS-II: Not Open Source in a narrower sense, no Linux
- (xBSD) Unix: [Bennett, 2003]
- VxWorks: Not Open Source
- QNX: Not Open Source
- PikeOS: Not Open Source

Analyses
Outline

Short-listed kernel:

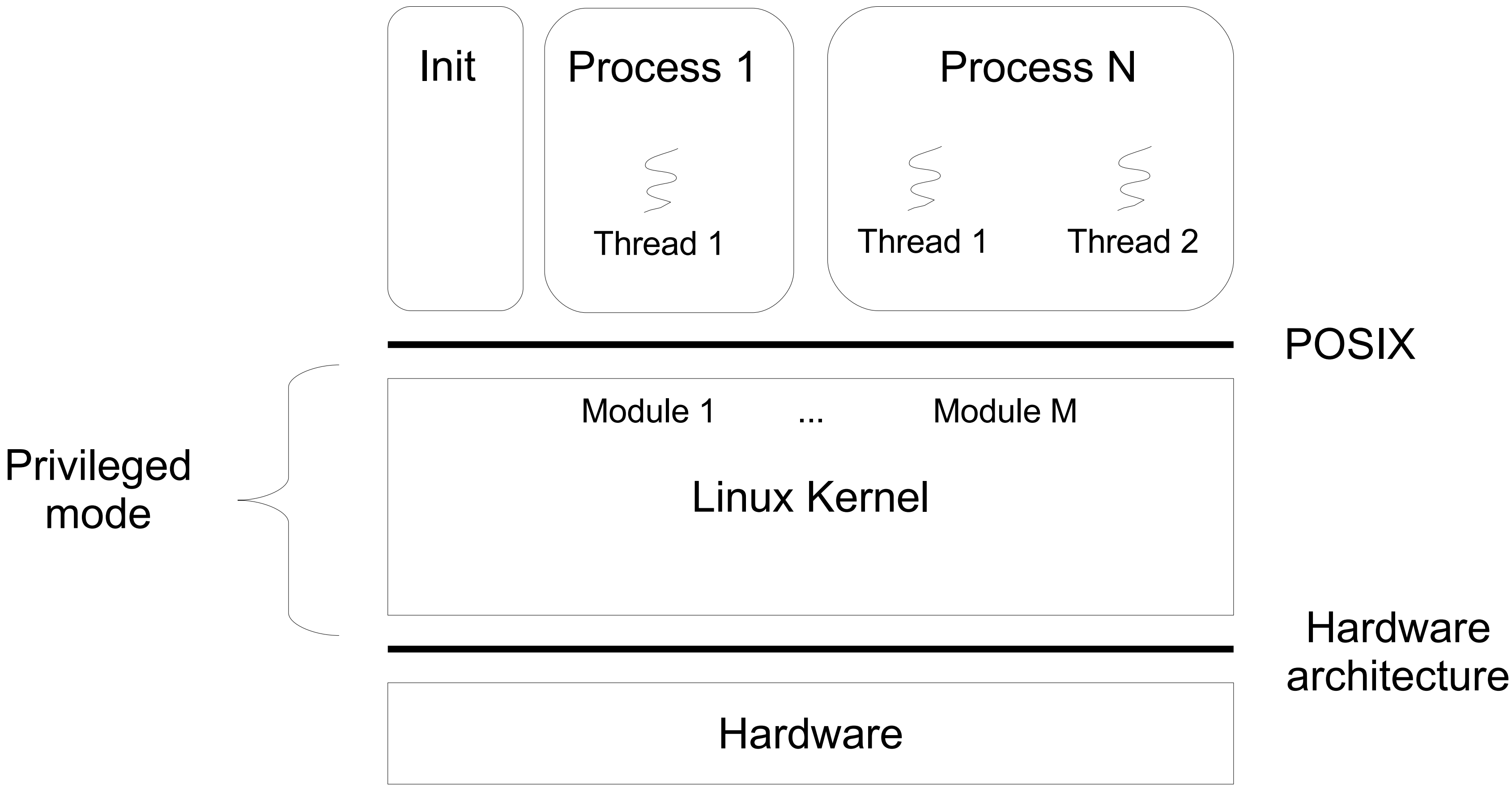
- Linux: Monolithic kernel with and without real-time patches
- Xen: Virtual Machine Monitor (VMM) from University of Cambridge, UK
- Minix 3: µ-kernel OS from Vrije Universiteit Amsterdam
- L4: Generic µ-kernel API from Jochen Liedtke

Methodology:

- Qualitativ or quantitativ analyses? → Qualitative analyses!
- LoC metric: `find . -regex '.*\.\(c\|cc\)' -print0 | xargs -0 cat | wc -l`

Analyses
Linux

Free multi-user UNIX for PC



Analyses

Linux

Real-Time:

- POSIX RT extension: RT scheduling classes, locking pages to physical memory
- Any device driver may block the system

Partitioning:

- Space partitioning enforced by MMU above kernel
- User-Mode scheduling possible within a process

Open standards:

- POSIX is “quasi free”

Modularity:

- Defined interface for device drivers
- Source code split in architecture and generic code

TCB:

- Monolithic kernel
- Kernel 2.6.9. for IA32 without drivers < 150.000 LoC

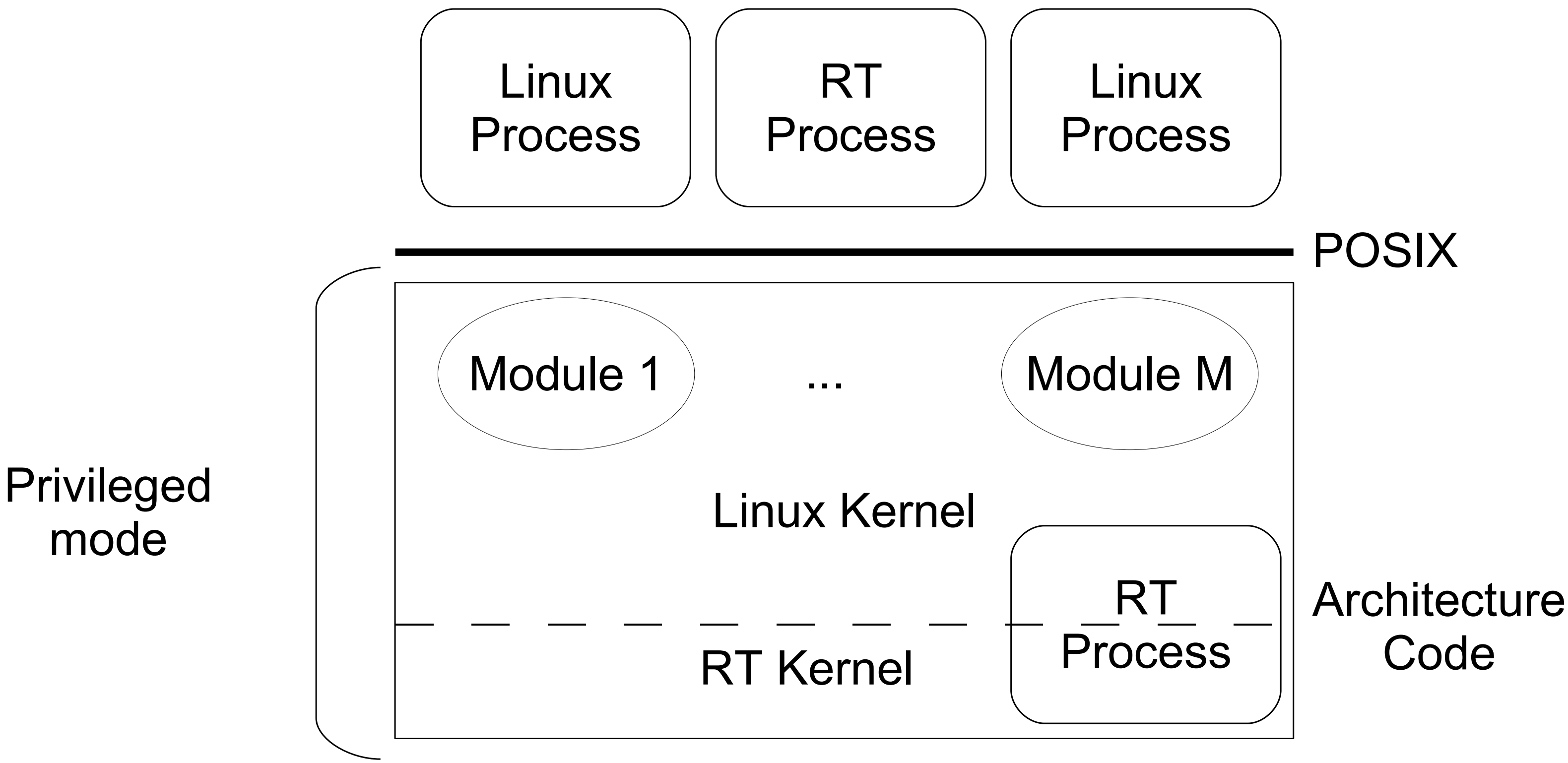
Analyses
Linux+RT

Real-Time extensions for Linux:

- **Patching:**
  - Reduced interrupt latency
  - Less none-preemptible kernel code
  - Some RT patches already included with kernel 2.6.18
  - RT-distributions (selection): *TimeSys*, *MontaVista*
- **Dual-Kernel:**
  - *RTLinux*
    - Linux is an idle thread of the µ-kernel
    - µ-kernel and Linux share kernel mode → no partitioning
    - Software patent → not Open Source in a narrower sense
    - Windriver announced on 20/02/07 that it bought all rights for RTLinux (including the patent)

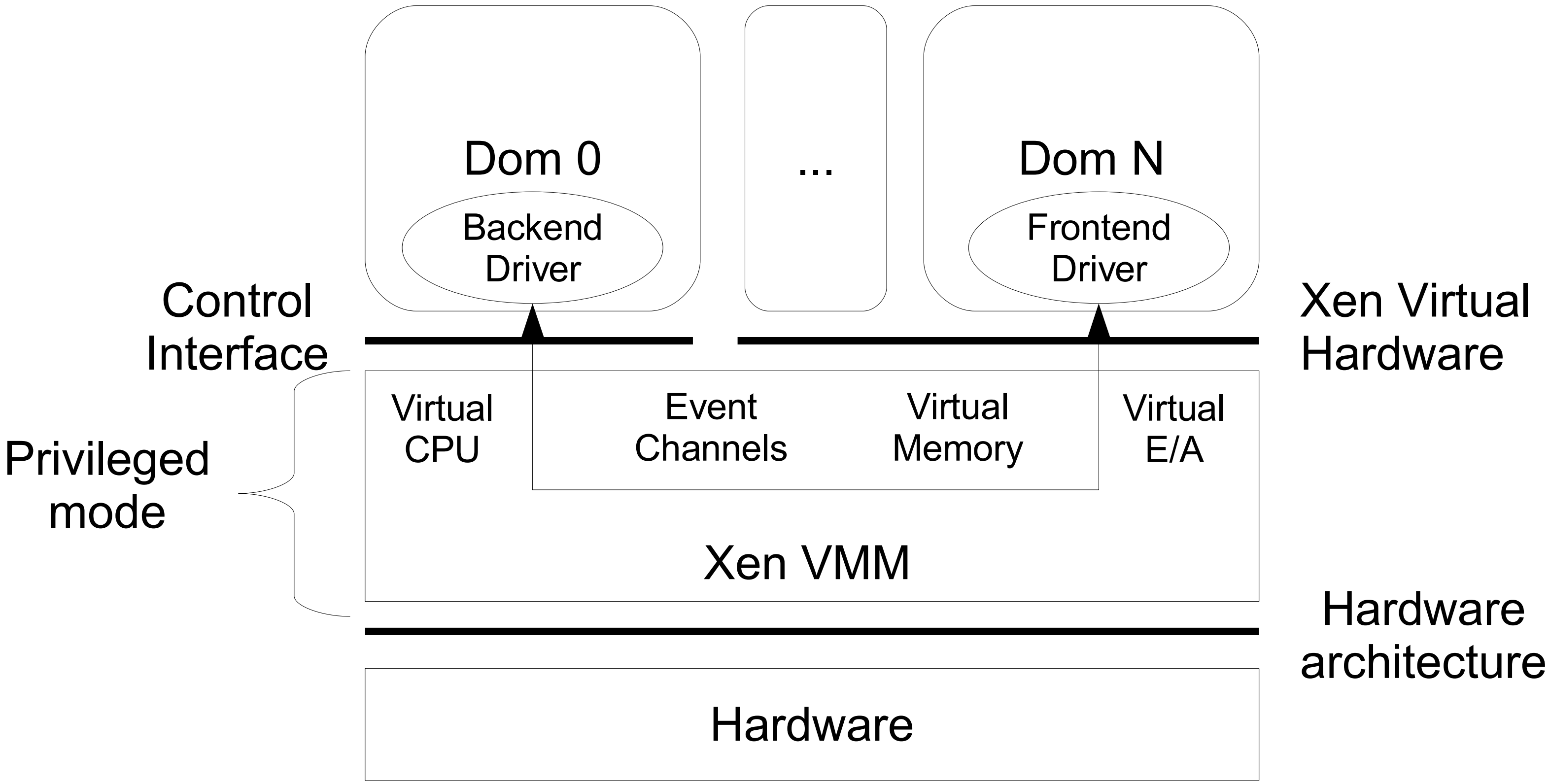
Analyses
Linux+RT

- **Dual-Kernel:**
  - *RTAI/Adeos*
    - Adeos I-Pipe is loaded as kernel module → no partitioning
    - RT-tasks running in user- or kernel mode, co-scheduler for RT-asks



Analyses
Xen

Virtualised hardware for up to 100 guest OS [Barham+, 2003]



Analyses
Xen

Real-Time:

- EDF-Scheduler
- Split-Driver: Backend in Dom0, Frontend in guest OS

Partitioning:

- Performance isolation through virtualisation of memory, CPU, I/O, interrupts
- 2-stage scheduling: VMM scheduler on domain level and scheduler of guest OS

Open standards:

- Virtual hardware is subset of real hardware
- Xen management API

Modularity:

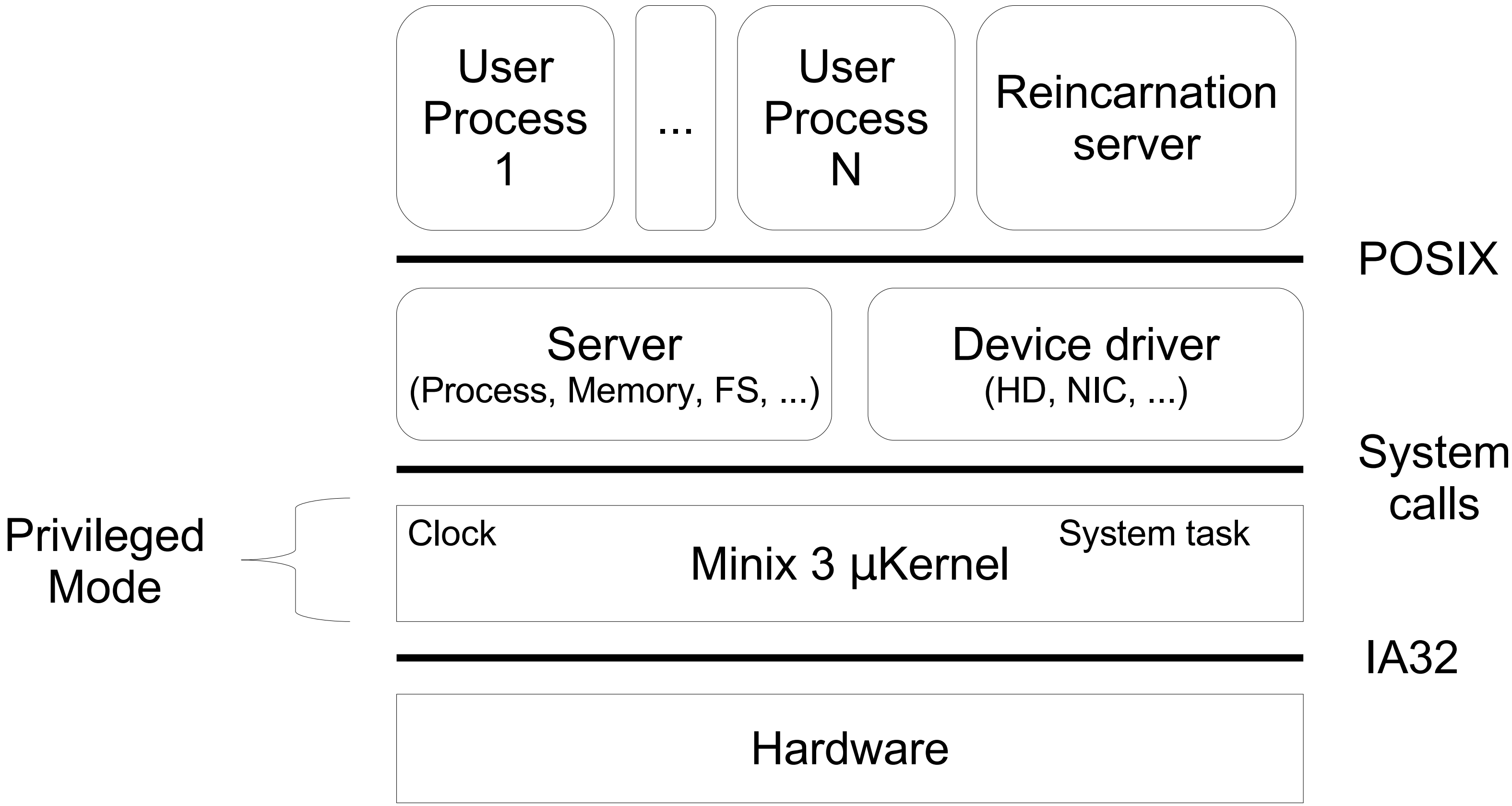
- Logical separated VMM, Dom0 and guest OS

TCB:

- Hypervisor + Dom0
- `xen-3.0.4_1-src/xen/arch/x86` < 60.000 LoC

Analyses
Minix 3

Computer system following the „TV model“ [Herder+, 2006]



Analyses
Minix 3

Real-Time:

- No RT-scheduler
- Memory management uses swapping

Partitioning:

- Address spaces protected by MMU
- Process-level scheduling

Open standards:

- POSIX

Modularity:

- Tight relations between µ-kernel and OS-services
- Servers and device drivers are isolated from µ-kernel

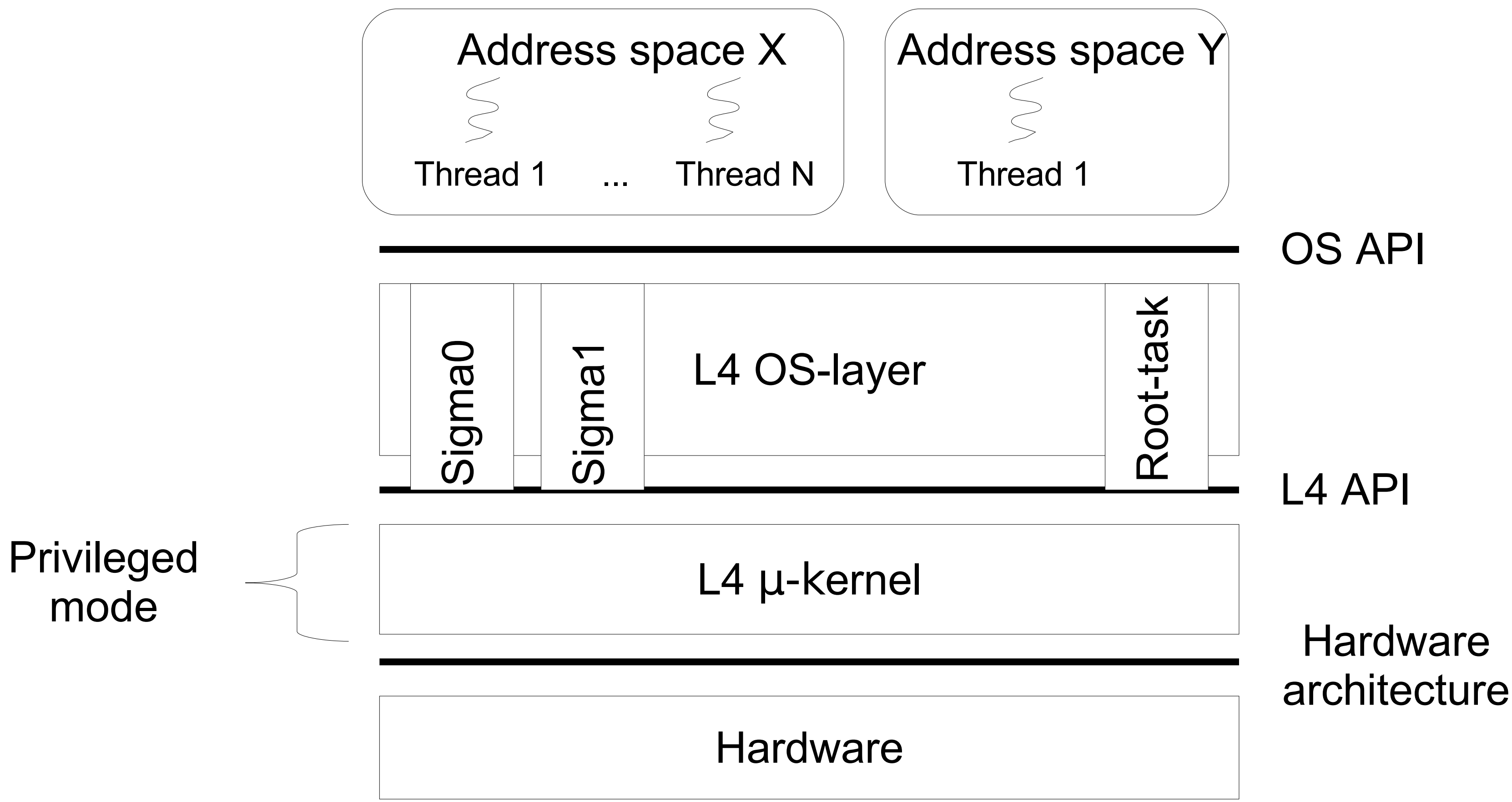
TCB:

µ-kernel + clock driver + system task + process- and memory manager

- `src/kernel/*.c` < 5.000 LoC (only IA32)

Analyses
L4

Hardware abstraction by address spaces, threads, IPC



Analyses
L4

Real-Time:

- RR-scheduler with 256 priorities
- Interrupts by IPC, handling in user mode

Partitioning:

- Address spaces protected by MMU
- Timeslice donation enables scheduling within address spaces

Open standards:

- Kernel API with different versions: V2, X.0, X.2, N1
- OS API depends on OS-layer

Modularity:

- Isolation of µ-kernel and user mode servers/device drivers
- L4Ka::Pistachio and successors separate API / architecture / platform

TCB:

- µKernel, Sigma0, Root-task
- L4Ka::Pistachio for IA32 < 15.000 LoC

Analyses
Decision

Rating:

- Linux: Big TCB (-), no hard real-time (-)
- Linux+RT: Increased complexity (-)
- Xen: Clean partitioning (+), small TCB (+), no hardware abstraction (-)
- Minix 3: No real-time (-), small TCB (+), no kernel API (-)
- L4: Hard real-time [Ruocco, 2006] (+), hardware abstraction (+), defined kernel API (+)

None-IMA factors:

- Availability for PowerPC 750: Linux, L4
- Roadmap of L4 projects:
  - → NICTA: seL4, L4.verified (mathematical verified correctness)
  - → TU Dresden: Verified Fiasco

Implementation
Alternatives

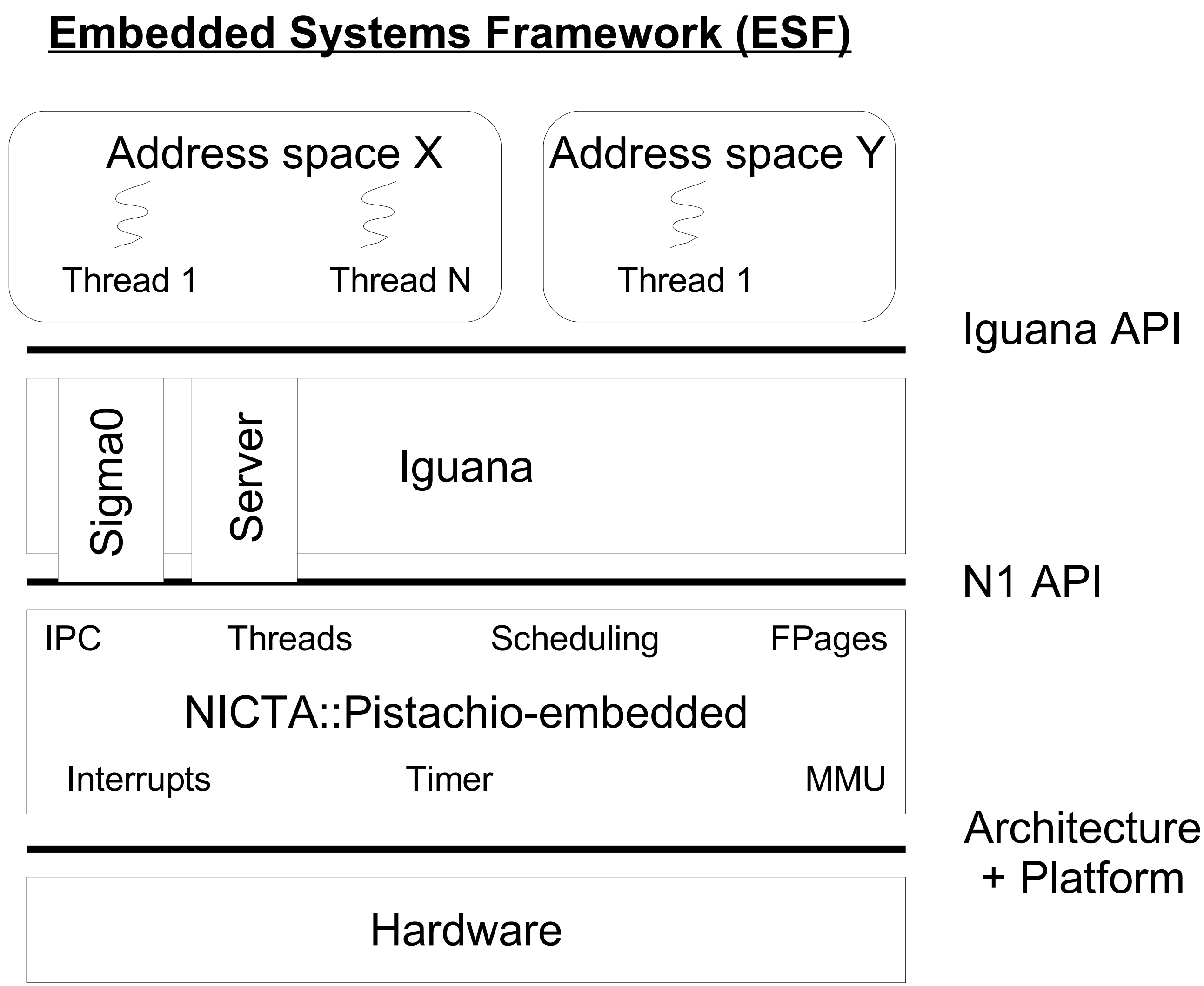
L4 µKernel:

Name	***API***	***Project***	***Architectures***
Pistachio	X.2	L4Ka	IA32/64, ARM, PowerPC32/64, ...
Pistachio-embedded	N1	NICTA	IA32, ARM, MIPS
OKL4	OKL4	OKL	IA32, ARM
Fiasco	V2/X.0	DROPS	IA32

Linux on L4:

Name	***Precondition***
Afterburner	Pistachio + GCC-Afterburner
Wombat	Pistachio-embedded + ESF ◄——
L4Linux	Fiasco + L4Env
User Mode Linux	Linux

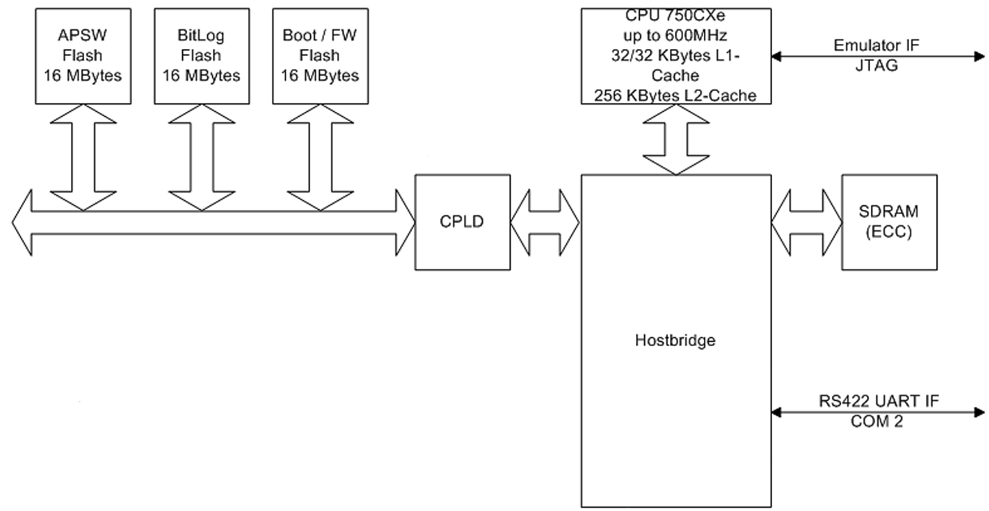
Implementation
ESF-Modell



Implementation
Hardware

Board details (relevant parts):

- PowerPC 750 Cxe @ 600 MHz
- 128 MB SD-RAM
- 16 MB boot flash
- 100 MHz bus frequency
- Hostbridge Marvell GT-6426x



CPU details:

- Decremeter works with (bus frequency)/4
- Separated Memory Mangement Unit (MMU) for Data and Instructions
- 4 Block Address Translations per MMU
- Segmented paging using 4 KB pages
- TLB with 128 entries per MMU

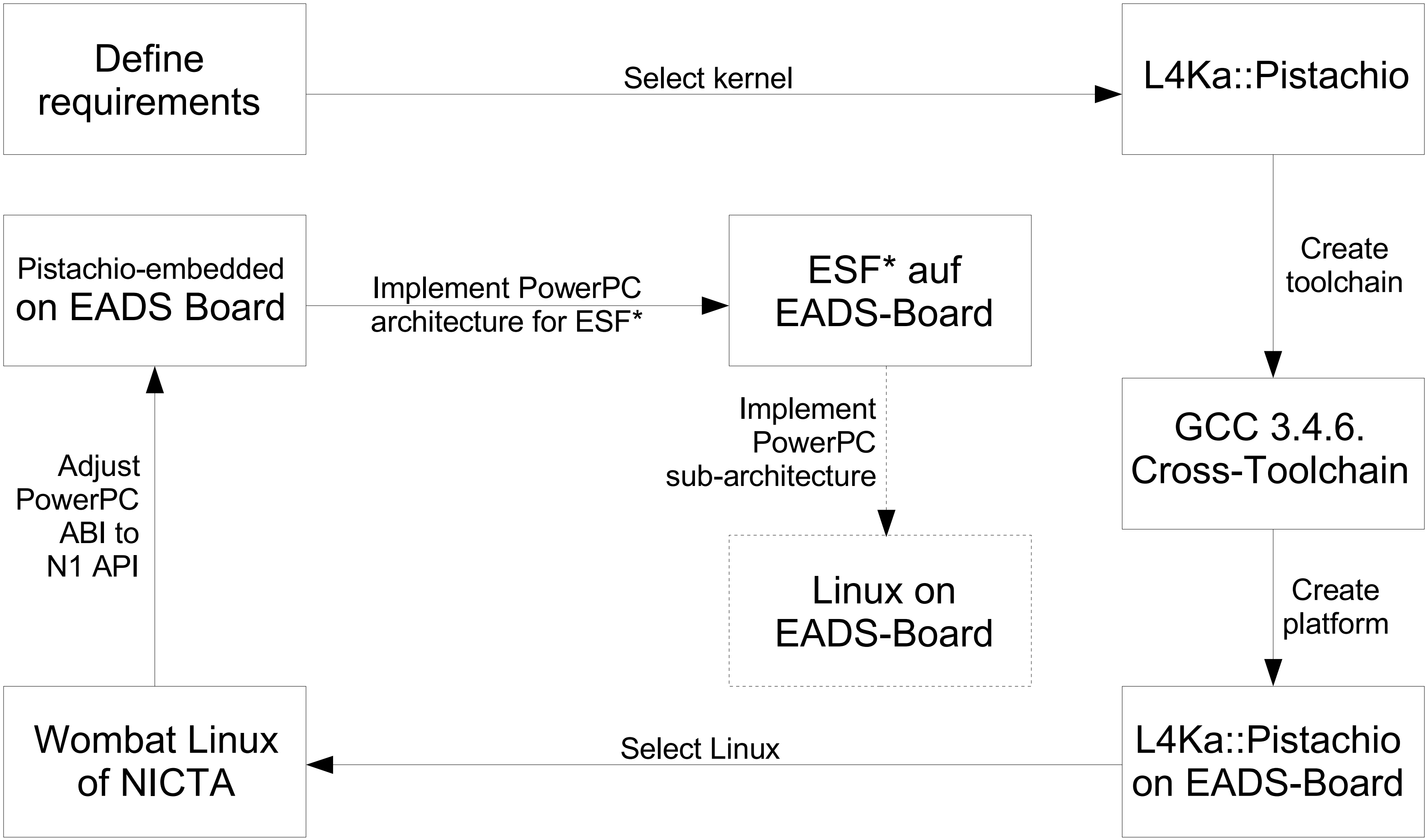
Implementation
Hardware + IMA

Hardware support for IMA-criterias:

- Real-Time:
  - ➔ Decrementer interrupt with 0.04 µs (4/Bus-frequency) resolution
- Partitioning:
  - ➔ Time:
    - Decremeter enables scheduling with variable time slices
    - Devices of inactive partitions could be masked on the interrupt controller of the hostbridge
  - ➔ Space:
    - 2 CPU Modes: User Model and Supervisor Model
    - MMU for data and instructions / with BAT or Paging
    - MMU also protects devices because of memory mapped I/O
- Modularity: 3 level architecture of processor
- TCB: n/a

Implementation

Steps



*) ESF = Embedded Systems Framework of National ICT Australia

Implementation
Steps

1. Cross-Toolchain:
   - Host Plattform: IA32/Linux → Target Platform: PowerPC32/Elf
2. Build-System:
   - Integrate PowerPC architecture and board in `SCons`
   - Add PowerPC support to `Dite` (Tool for merging ELF-Images)
3. Architecture:
   - Reorganise kernel startup code: OpenFirmware, KMem, sequence
   - N1: no more Local Thread IDs / new User Thread Control Block (UTCB)
   - Adjust ABI of 4 system calls to N1 API
   - Implement new functions of Kernel Debugger
4. Platform:
   - Boot-code of ELF-loader and kernel
   - Interrupt handling
   - Linker script
   - Implement `getc()` and `putc()` for Marvell debug I/O

Evaluation

Results

5. ESF:

- `libc`: Data types, jump functions, CRT0, platform I/O
- `libcycles`: Read cycle counter of CPU
- `libiguana`: Implement CRT0 for threads
- `libl4`: Adjust to N1 ABI and UTCB, new KDebug functions
- `libmutex`: Implement `try_lock`
- `drv_powerpc_timer`: Decrementer based user mode timer

Results:

- Pistachio-embedded supports PowerPC
- ESF supports PowerPC
- ESF is running on Hellas-board

Evaluation
IMA

Real-Time:

- L4Ka::Pistachio(-embedded) using static time slices of length: 1953 µs
- Interrupts are disabled inside the kernel

Partitioning:

- System call `MemoryControl` was not implemented → Caching of pages can't be prohibited → BAT-entry for Marvell-Bridge registers needed → World-writeable
- Incrementing Pagefaults don't work (bug in TLB-handling?) → Pager always assigns RW access

Evaluation
IMA

Modularity: Nothing changed to Analyses

Open standards: Nothing changed to Analyses

TCB:

- Known bugs in μ-kernel:
  - `MemoryControl` system call goes wild
  - Smashthread Unit Test fails: kernel is running out of memory for “Thread Control Block” after creating around 20 threads → memory leak?
  - Missing in TLB-update → bug in paging system for PowerPC
- Iguana needed many bug fixes an work-arounds before running → more bugs suspected

Conclusion:

- Development stadium “Alpha”. Can't be used in IMA, yet!
- PowerPC code needs revision
- Good approach, bad implementation

Actual development:

- Open Kernel Labs published OKL4 with BSD licence
- Minix 3 was ported to PowerPC by [Alting, 2006]
- Xen was ported to PowerPC 970 by IBM

Thanks for your attention!

Bibliography

- **[Alting, 2006]**: I. A. Alting: „MinixPPC: A port of the MINIX OS to the PowerPC platform”. Amsterdam. 2006
- **[Barham+, 2003]**: P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield: „Xen and the Art of Virtualization“. New York. 2003
- **[Benett, 2003]**: M. D. Benett: „A Kernel For IMA Systems“. York. 2003
- **[heise, 2003]**: heise online: „Embedded-Spezialist Wind River tritt Open-Source-Labs bei“. 2003
- **[heise, 2006]**: heise online: „Der Pinguin geht auf U-Boot-Jagd“. 2006
- **[Herder+, 2006]**: J. N. Herder, H. Bos, B. Gras, P. Homburg, A. S. Tanenbaum: „MINIX 3: A Highly Reliable, Self-Repairing Operating System. Amsterdam. 2006
- **[Liedtke, 1995]**: J. Liedtke: „On µ-Kernel Construction“. New York. 1995

Bibliography

- **[Ruocco, 2006]**: S. Ruocco: „Real-Time Programming and L4 Microkernels". Sydney. 2006
- **[Rushby, 1999]**: J. Rushby: „Partitioning in Avionics Architectures: Requirements, Mechansims, and Assurance". Menlo Park. 1999
- **[Timmerman+, 2005]:** M. Timmerman, L. Pernell: „RTOS State Of The Art - Understanding RTOS Technology And Markets?". 2005