A kernel in a library
Genode’s custom kernel approach

Martin Stein
<martin.stein@genode-labs.com>
Outline

1. Motivation

2. Overview

3. Scheduling

4. Capabilities

5. Communication
Outline

1. Motivation
2. Overview
3. Scheduling
4. Capabilities
5. Communication
The impetus of diversity

- NOVA, Fiasco.OC, OKL4, L4ka::Pistachio, L4/Fiasco, Linux SeL4
The impetus of diversity

- NOVA, Fiasco.OC, OKL4, L4ka::Pistachio, L4/Fiasco, Linux SeL4

- Flexibility in development and application
Genode on third-party Kernels

The impetus of diversity

- NOVA, Fiasco.OC, OKL4, L4ka::Pistachio, L4/Fiasco, Linux SeL4
- Flexibility in development and application
- Versatility in testing
Kernel perspective

- Aim for comprehensive security concept
Kernel perspective

- Aim for comprehensive security concept
- Self-contained unit that mistrusts all users
Kernel perspective

- Aim for comprehensive security concept
- Self-contained unit that mistrusts all users

Perspective of Genode’s Core

- Bring Kernel concept in line with Genode API
Genode on third-party Kernels

Kernel perspective

- Aim for comprehensive security concept
- Self-contained unit that mistrusts all users

Perspective of Genode’s Core

- Bring Kernel concept in line with Genode API
- Must be trusted anyway
Genode on third-party Kernels

**Drawbacks**

- Concepts get bend in shape (Signals)
Drawbacks

- Concepts get bend in shape (Signals)
- Work is done redundantly (memory management)
Genode on third-party Kernels

Drawbacks

- Concepts get bend in shape (Signals)
- Work is done redundantly (memory management)
- Deficiencies get worked around (Capability delegation)
Creating a custom solution

Idea

- Kernel that trusts Core and is designed for Core’s needs
Creating a custom solution

Idea

- Kernel that trusts Core and is designed for Core’s needs
- Minimalistic library that enables Core to run as root domain
Creating a custom solution

Idea

- Kernel that trusts Core and is designed for Core’s needs
- Minimalistic library that enables Core to run as root domain
- Run most critical code in the simplest manner
1. Motivation

2. Overview

3. Scheduling

4. Capabilities

5. Communication
Kernel tasks

- Exception vectors
Kernel tasks

- Exception vectors
- Scheduling
Kernel tasks

- Exception vectors
- Scheduling
- Controls interrupts
Kernel tasks

- Exception vectors
- Scheduling
- Controls interrupts
- Communication: IPC and Signals
Kernel tasks

- Exception vectors
- Scheduling
- Controls interrupts
- Communication: IPC and Signals
- Capabilities
Kernel tasks

- Exception vectors
- Scheduling
- Controls interrupts
- Communication: IPC and Signals
- Capabilities
- Cache and TLB maintenance
Kernel tasks

- Exception vectors
- Scheduling
- Controls interrupts
- Communication: IPC and Signals
- Capabilities
- Cache and TLB maintenance
- Virtualization
## Kernel interface

<table>
<thead>
<tr>
<th>Threads, VMs</th>
<th>PDs, Capabilities</th>
<th>Communication, IRQs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core-only</td>
<td></td>
<td></td>
</tr>
<tr>
<td>thread new/del</td>
<td>obj new/del</td>
<td>signal receiver new/del</td>
</tr>
<tr>
<td>thread start</td>
<td>pd new/del</td>
<td>signal context new/del</td>
</tr>
<tr>
<td>thread pause</td>
<td>pd update</td>
<td>irq new/del</td>
</tr>
<tr>
<td>thread resume</td>
<td></td>
<td>irq ack</td>
</tr>
<tr>
<td>thread route event</td>
<td>update data region</td>
<td>signal context kill</td>
</tr>
<tr>
<td>thread quota</td>
<td>update instr region</td>
<td>signal submit</td>
</tr>
<tr>
<td>vm new/del</td>
<td>cap ack</td>
<td>signal await</td>
</tr>
<tr>
<td>vm run</td>
<td>cap delete</td>
<td>signal ack</td>
</tr>
<tr>
<td>vm pause</td>
<td></td>
<td>msg send request</td>
</tr>
<tr>
<td></td>
<td></td>
<td>msg send reply</td>
</tr>
<tr>
<td></td>
<td></td>
<td>msg await request</td>
</tr>
<tr>
<td>Common</td>
<td></td>
<td></td>
</tr>
<tr>
<td>thread pause current</td>
<td></td>
<td></td>
</tr>
<tr>
<td>thread resume local</td>
<td></td>
<td></td>
</tr>
<tr>
<td>thread yield</td>
<td>update instr region</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cap ack</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cap delete</td>
<td></td>
</tr>
</tbody>
</table>
Qualities

- All dynamic memory gets accounted
  → No exhaustion
Qualities

- All dynamic memory gets accounted
  → No exhaustion

- Modeled as state machine
  → Low complexity
  → Fast kernel passes
Trusted Computing Base

A kernel in a library Genode’s custom kernel approach

= 30 - 60 KLOC
Trusted Computing Base

A kernel in a library  Genode’s custom kernel approach

Init

Core

Kernel lib

= 22 KLOC
Hardware support

- ARMv7
  - Panda Board, i.MX53 QSB, USB Armory, Wand Board, Arndale, Odroid XU, Zynq, PBXA9
  - SMP, Virtualization, Trustzone, ...

A kernel in a library  Genode’s custom kernel approach
Hardware support

- ARMv7
  - Panda Board, i.MX53 QSB, USB Armory, Wand Board, Arndale, Odroid XU, Zynq, PBXA9
  - SMP, Virtualization, Trustzone, ...

- x86 64 Bit, Raspberry Pi (ARMv6), RISC-V, Muen Separation Kernel
1. Motivation

2. Overview

3. Scheduling

4. Capabilities

5. Communication
Scheduling

Absolute priorities

4  —  Timer
3  —  USB
2  —  WM
2  —  Audio
1  —  AVplay
0  —  GCC
0  —  CP

*USB goes mad*
Scheduling

Quota-bound priorities

4 — Timer 10%
3 — USB 13%
2 — WM 18%
2 — Audio 10%
1 — AVplay 15%
GCC
CP
Scheduling

Quota-bound priorities

- Timer
- USB
- WM
- Audio
- AVplay
- GCC
- CP

Reset quota

USB goes mad
Scheduling

Donation of CPU resources from parents to their children

- Virtualbox
  - 2
- Mupdf
  - 3
- CLI
  - 1-4
- Nitpicker
  - 5
- Init
  - 1-8
Outline

1. Motivation
2. Overview
3. Scheduling
4. Capabilities
5. Communication
Capabilities

- Automatic creation or translation on IPC delegation
Capabilities

- Automatic creation or translation on IPC delegation
- No name diversity in a PD
Capabilities

- Automatic creation or translation on IPC delegation
- No name diversity in a PD
- Costs get accounted via PD session quota
Capabilities

Collaborative lifetime management for Capabilities

- IPC with remote name
- create
- Local Name Object
- delete

Kernel

User

IPC with local name
ack
delete
IPC with local name
ack
delete
Outline

1. Motivation
2. Overview
3. Scheduling
4. Capabilities
5. Communication
IPC implicitly delegates CPU resources
Collaborative lifetime management for Signals

- Kernel
  - submit
  - assign
  - local pointer
  - deliver
  - local pointer
  - ack delivery
  - kill unblocks

- User
  - main
  - handler
  - create
  - update
  - destroy
  - kill
  - (blocks)
Thank you!

Genode OS Framework
http://genode.org

Genode Labs GmbH
http://genode-labs.com

Source code at GitHub
http://github.com/genodelabs/genode