An Exploration of the seL4 Kernel from Genode’s Perspective

Norman Feske
<norman.feske@genode-labs.com>
Outline

1. Background (Genode)
2. The seL4 project
3. Capabilities and kernel objects
4. Virtual memory
5. What’s next?
Outline

1. Background (Genode)
2. The seL4 project
3. Capabilities and kernel objects
4. Virtual memory
5. What’s next?
Motivations behind Genode

- Principle of least privilege
Motivations behind Genode

- Principle of least privilege
- Mixed criticality
Motivations behind Genode

- Principle of least privilege
- Mixed criticality
- Dependability
Motivations behind Genode

- Principle of least privilege
- Mixed criticality
- Dependability
- Scalability
Motivations behind Genode

- Principle of least privilege
- Mixed criticality
- Dependability
- Scalability
- Flexibility
Key technologies

- Microkernels
- Componentization, kernelization
- Capability-based security
- Virtualization
Key technologies

- Microkernels
- Componentization, kernelization
- Capability-based security
- Virtualization

...but how to compose those?
Combined with virtualization
Genode operating-system framework

An Exploration of the seL4 Kernel from Genode’s Perspective
The Book “Genode Foundations”

The seL4 kernel

- Developed by NICTA (DATA61) / UNSW in Sydney
- The world’s first formally verified OS kernel
The seL4 kernel

- Developed by NICTA (DATA61) / UNSW in Sydney
- The world’s first formally verified OS kernel
- Capability-based security
- Resilient against kernel-resource exhaustion
- Supports ARM and x86
The seL4 kernel

- Developed by NICTA (DATA61) / UNSW in Sydney
- The world’s first formally verified OS kernel
- Capability-based security
- Resilient against kernel-resource exhaustion
- Supports ARM and x86
- GPLv2 since August 2014
- Active and dedicated community
Outline

1. Background (Genode)
2. The seL4 project
3. Capabilities and kernel objects
4. Virtual memory
5. What’s next?
# seL4 kernel-object inventory

<table>
<thead>
<tr>
<th>seL4 kernel object</th>
<th>Analogy</th>
</tr>
</thead>
<tbody>
<tr>
<td>UntypedObject</td>
<td>Range of physical memory</td>
</tr>
<tr>
<td>TCBObject</td>
<td>Thread</td>
</tr>
<tr>
<td>EndpointObject</td>
<td>Destination of IPC calls</td>
</tr>
<tr>
<td>AsyncEndpointObject</td>
<td>Recipient of signals</td>
</tr>
<tr>
<td>CapTableObject (&quot;CNode&quot;)</td>
<td>Array of capabilities</td>
</tr>
<tr>
<td>IA32_4K</td>
<td>4 KiB page frame</td>
</tr>
<tr>
<td>IA32_4M</td>
<td>4 MiB page frame</td>
</tr>
<tr>
<td>IA32_PageTableObject</td>
<td>Page table</td>
</tr>
<tr>
<td>IA32_PageDirectoryObject</td>
<td>Protection domain</td>
</tr>
</tbody>
</table>
seL4 capabilities ("selectors")

```
<table>
<thead>
<tr>
<th>CSpace</th>
<th>CNode</th>
<th>CNode</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0x5</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0x6</td>
<td>0xffff</td>
<td>0xffff</td>
</tr>
<tr>
<td>0xfff</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0x55</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0xff</td>
<td>0x200</td>
<td>0xfff</td>
</tr>
<tr>
<td>0xfff</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
```

An Exploration of the seL4 Kernel from Genode’s Perspective
Once upon a time, there was untyped memory...

--- boot info ---
initThreadCNodeSizeBits: 12
untyped:

[38] [00100000,00107fff]
[39] [00108000,00109fff]
[3a] [001a0000,001bffff]
[3b] [001c0000,001fffff]
[3c] [00200000,003fffffff]
[3d] [00400000,007fffffff]
[3e] [00800000,00fffffff]
[3f] [01000000,01fffffff]
[40] [02000000,02fffffff]
[41] [03000000,037fffffff]
[42] [03800000,03bfffffff]
[43] [03c00000,03dfffffff]
[44] [03e00000,03efffff]
[45] [03f00000,037fffffff]
[46] [03f80000,03fbffff]
[47] [03fc0000,03dfffff]
[48] [03fe0000,03efffff]
[49] [03ff0000,03f7ffff]
[4a] [03ff8000,03fbffff]
[4b] [03ffe000,03dfffffff]
[4c] [00189000,001897ff]
Kernel-object creation

CSpace
...
0x44
...
...
Untyped Memory
Kernel-object creation (2)

CSpace
...
0x17
...
0x44
...

Untyped Memory

offset

Endpoint

An Exploration of the seL4 Kernel from Genode’s Perspective
Managing untyped memory

- Book keeping
  - Tracking of free physical memory
  - seL4: physical address range ↔ untyped memory selector

Untyped memory regions are naturally aligned
There are adjacent untyped memory regions
Kernel objects cannot span multiple untyped memory regions
→ Trick: natural alignment of all allocations
Managing untyped memory

- Book keeping
  - Tracking of free physical memory
  - seL4: physical address range $\leftrightarrow$ untyped memory selector

- Untyped memory regions are naturally aligned
Managing untyped memory

- Book keeping
  - Tracking of free physical memory
  - *seL4*: physical address range $\leftrightarrow$ untyped memory selector

- Untyped memory regions are naturally aligned

- There are adjacent untyped memory regions
Managing untyped memory

- Book keeping
  - Tracking of free physical memory
  - seL4: physical address range ↔ untyped memory selector

- Untyped memory regions are naturally aligned

- There are adjacent untyped memory regions

- Kernel objects cannot span multiple untyped memory regions
Managing untyped memory

- Book keeping
  - Tracking of free physical memory
  - seL4: physical address range $\leftrightarrow$ untyped memory selector

- Untyped memory regions are naturally aligned

- There are adjacent untyped memory regions

→ Trick: natural alignment of all allocations
Core’s CSpace organization

An Exploration of the seL4 Kernel from Genode’s Perspective
Capability delegation and invocation

An Exploration of the seL4 Kernel from Genode’s Perspective
An Exploration of the seL4 Kernel from Genode’s Perspective
Capability re-identification problem
Capability re-identification problem

PD A

Local Object 45

13 minted endpoint

send 13

PD B

new selector 17

meta data

send 17

PD C

34 new selector

18 new selector

???

An Exploration of the seL4 Kernel from Genode's Perspective
Problem in Genode

An Exploration of the seL4 Kernel from Genode’s Perspective
Current workaround

An Exploration of the seL4 Kernel from Genode’s Perspective 23
Current workaround

An Exploration of the seL4 Kernel from Genode’s Perspective 23
1. Background (Genode)

2. The seL4 project

3. Capabilities and kernel objects

4. Virtual memory

5. What’s next?
Virtual memory management

Problem

- Applications live in virtual memory
  - *The kernel maintains meta data and page tables*
Problem

- Applications live in virtual memory → *The kernel maintains meta data and page tables*
- Where to take the memory from?
Virtual memory management

Problem

- Applications live in virtual memory → *The kernel maintains meta data and page tables*

- Where to take the memory from? → *Traditional approach: kernel-local memory pool*
Virtual memory management

Problem

- Applications live in virtual memory → *The kernel maintains meta data and page tables*

- Where to take the memory from? → *Traditional approach: kernel-local memory pool*

- What happens when the memory get exhausted?
Virtual memory management

Problem

- Applications live in virtual memory
  \[ \text{The kernel maintains meta data and page tables} \]

- Where to take the memory from?
  \[ \text{Traditional approach: kernel-local memory pool} \]

- What happens when the memory get exhausted?
  \[ \text{Panic!} \]

- Who provokes kernel memory consumption?
  \[ \text{Untrusted application code!} \]
Problem

- Applications live in virtual memory
  → *The kernel maintains meta data and page tables*

- Where to take the memory from?
  → *Traditional approach: kernel-local memory pool*

- What happens when the memory get exhausted?
  → *Panic!*

- Who provokes kernel memory consumption?
  → *Untrusted application code!*
The seL4 way of virtual memory management
The seL4 way of virtual memory management

An Exploration of the seL4 Kernel from Genode’s Perspective 26
Attempt to map a page twice
Attempt to map a page twice

- CSpace: 0x5, 0x6, 0x7
- Page dir: 0x1, 0x3ff
- Page table: 0x1, 0x10, 0x3ff
- 4 KiB page frame

<< Frame already mapped >>

Virtual memory: 0x401000
Mapping a page twice, the seL4 way

- CSpace
  - ... 0x5
  - 0x6
  - 0x7
  - ...
  - 0x13
  - ...

- Page dir
  - ... 0x1
  - 0x3ff

- Page table
  - ... 0x1
  - 0x10
  - ...
  - 0x3ff

- 4 KiB page frame

- Virtual memory
  - 0x0
  - 0x401000
  - 0x410000

An Exploration of the seL4 Kernel from Genode's Perspective 28
Implications for Genode

On Genode, each page must be considered as shared memory
Implications for Genode

On Genode, each page must be considered as shared memory

→ One kernel object (CNode entry) for each page-table entry
Implications for Genode

On Genode, each page must be considered as shared memory

→ One kernel object (CNode entry) for each page-table entry

→ How to name the selectors?

Preallocation is infeasible.

Solution: Virtual software-loaded TLB

Fixed pool of page tables per PD, used in LRU fashion

Leveraging Genode’s resource trading mechanism:

Page-table pool size is a PD-specific
Implications for Genode

On Genode, each page must be considered as shared memory

→ One kernel object (CNode entry) for each page-table entry

→ How to name the selectors? Preallocation is infeasible.
Implications for Genode

On Genode, each page must be considered as shared memory

→ One kernel object (CNode entry) for each page-table entry

→ How to name the selectors? Preallocation is infeasible.

Solution: Virtual software-loaded TLB

- Fixed pool of page tables per PD, used in LRU fashion
Implications for Genode

On Genode, each page must be considered as shared memory

→ One kernel object (CNode entry) for each page-table entry
→ How to name the selectors? Preallocation is infeasible.

Solution: Virtual software-loaded TLB

- Fixed pool of page tables per PD, used in LRU fashion
- Leveraging Genode’s resource trading mechanism:
  → Page-table pool size is a PD-specific
1. Background (Genode)

2. The seL4 project

3. Capabilities and kernel objects

4. Virtual memory

5. What’s next?
What’s next?

- seL4 2.0
- Signal API backend, interrupts
- Memory-mapped I/O
- Real lock implementation
- Shared library support

→ Interactive scenarios by mid 2016
Open issues

- Asynchronous notifications
Open issues

- Asynchronous notifications
- Capability integrity protection
Open issues

- Asynchronous notifications
- Capability integrity protection
- Superpages
Thank you

Articles about Genode on seL4
http://genode.org/documentation/articles

Genode OS Framework
http://genode.org

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

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