Microkernel virtualization under one roof
- dare the impossible -

Alexander Böttcher
<alexander.boettcher@genode-labs.com>
1. Introduction

2. Kernel interfaces

3. VM interface harmonization

4. VMMs harmonized

5. Conclusion
1. Introduction

2. Kernel interfaces

3. VM interface harmonization

4. VMMs harmonized

5. Conclusion
Motivation

Off-the-shell virtualization solution ridden with complexity.

Application of virtualization call for trustworthy solutions.

Complexity defeats trust.

Alternative approach
→ Microkernels with hardware assisted virtualization extensions
Component based virtualization architecture

- Guest OS
- Guest OS
- Guest OS

VMM

Apps
Resource management
Drivers

9,000 SLOC

NOVA Microhypervisor

Resource management

non-root mode

root mode

Kernel
Genode OS framework
General supported kernels on Genode

- Base-HW
- NOVA (Microhypervisor)
- FIASCO
- FIASCO.OC
- OKL4
- seL4
- L4Ka
- L4s
Kernels with hardware assisted virtualization
## VMM inventory of Genode

### Hardware assisted virtualization/separation support

<table>
<thead>
<tr>
<th>Microkernel</th>
<th>Host</th>
<th>VMM</th>
<th>Guest vCPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>hw</td>
<td>ARM, 32bit</td>
<td>custom</td>
<td>1, 32bit</td>
</tr>
<tr>
<td>hw/trustzone</td>
<td>ARM, 32bit</td>
<td>custom</td>
<td>1, 32bit</td>
</tr>
<tr>
<td>hw with Muen</td>
<td>Intel, 64bit</td>
<td>VBox 4</td>
<td>1, 32bit</td>
</tr>
<tr>
<td>NOVA</td>
<td>Intel &amp; AMD</td>
<td>Seoul</td>
<td>N, 32bit</td>
</tr>
<tr>
<td></td>
<td>32bit, 64bit</td>
<td>VBox 4</td>
<td>N, 32bit, 64 bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VBox 5</td>
<td>N, 32bit, 64 bit</td>
</tr>
</tbody>
</table>
Vision: VMMs runnable on all kernels w/o re-compilation
Research challenge

Vision: VMMs runnable on all kernels w/o re-compilation

- Focus on x86 microkernels for now
- → NOVA, seL4, Fiasco.OC, and -hw-
Vision: VMMs runnable on all kernels w/o re-compilation

- Focus on x86 microkernels for now
  → NOVA, seL4, Fiasco.OC, and -hw-

Approach: Generalize VM interface as used by -hw-
1. Introduction

2. Kernel interfaces

3. VM interface harmonization

4. VMMs harmonized

5. Conclusion
Flow of a virtualization event

User-level VMM

Guest OS

Microkernel virtualization under one roof - dare the impossible -
vCPU state on NOVA

Transfer: UTCB, VMCS/VMCB agnostic, partial state support
vCPU state on Fiasco.OC

Transfer: vCPU state, **not** VMCS/VMCB **agnostic**, **full** state
vCPU state on seL4

Transfer: **hybrid** - IPCBuffer & **syscall per VMCS register**
IPCBuffer: VM exit - 17 registers, VM enter - 3 registers
Control flow on NOVA

UTCB

VMM

thread
IPC call
user space

IPC reply

UTCB

NOVA microhypervisor

VMCS/VMCB

vCPU
Control flow on Fiasco.OC

Microkernel virtualization under one roof - dare the impossible - 17
Control flow on seL4
Control flow on Genode’s -hw-
1. Introduction

2. Kernel interfaces

3. VM interface harmonization

4. VMMs harmonized

5. Conclusion
Design goals

VMM → just a component

Genode components designed event driven
- Non-blocking thread (entrypoint) register for event sources
- Events cause transition in state machine
- State transition by Genode signal or RPC
Design goals

VMM → just a component

Genode components designed event driven
- Non-blocking thread (entrypoint) register for event sources
- Events cause transition in state machine
- State transition by Genode signal or RPC

VM event → just another event source
I/O event → just another event source

Kernel agnostic ABI
Unified vCPU state per platform
Envisioned vCPU handling

- VMM
- timer
- network

entrypoint

VM event

signal

user space

kernel space

vCPU0

vCPUn

kernel

Microkernel virtualization under one roof - dare the impossible -
Envisioned vCPU handling - multi core

VMM

entrypoint A

entrypoint B

user space

kernel space

kernel

vCPU A0 ... vCPU An

vCPU B0 ... vCPU Bn

Microkernel virtualization under one roof - dare the impossible -
**VM interface - kernel agnostic**

Genode -base- library with unified ABI in ld.lib.so
VM connection/session → VM address space established

- `create_vcpu()` - setup new vCPUs
- `cpu_state()` - access to guest state
- `attach/detach()` - memory management of VM
- `VM_handler` class - registration for VM event handling
- `run/pause()` - control execution of vCPUs - non-blocking
VM interface - kernel agnostic

- VMM
- ld.lib.so
- VM interface (client)
- init
- core
- VM interface (server)
- user space
- kernel space
- kernel

Microkernel virtualization under one roof - dare the impossible -
VM interface - kernel agnostic
Server: 200-400 LOC
Client: NOVA, seL4: ~500 - Fiasco.OC: ~1000 - hw: ~30 LOC
Control flow on Genode’s -hw- and NOVA

Microkernel virtualization under one roof - dare the impossible -
Control flow on Genode’s -hw- and NOVA

Event source (timer)

VMM

Entrypoint

hw/NOVA

signal/IPC call

run/IPC reply
non-blocking

VM exit

kernel

vCPU

vCPU0

vCPU1

VM exit

VM resume

event (timeout)

pause/recall
non-blocking

signal/IPC call

run/IPC reply
inject vIRQ

Microkernel virtualization under one roof - dare the impossible -
Control flow on seL4 and Fiasco.OC

Microkernel virtualization under one roof - dare the impossible -
Control flow on seL4 and Fiasco.OC

Blocking syscall unfortunate → complicates life

- Kernels provide mechanism to cancel

Avoid special case handling in Genode for first take
→ Workaround: spawn per vCPU extra thread
Control flow on seL4 and Fiasco.OC

Event source (timer)

VMM

- Entrypoint
- Handler0
- Handler1

- run
- non-blocking
- vmenter/vmresume

kernel

VM

- vCPU0
- vCPU1

- VM resume
- VM resume
Control flow on seL4 and Fiasco.OC

Event source (timer)

VMM
- Entrypoint
- Handler0
- seL4/Fiasco.OC

Kernel
- syscall returns
- signal
- inject vIRQ

VM
- vCPU0
- vCPU1

- run non-blocking
- vmenter/vmresume blocking syscall
- pause
- cancel vmenter/vmresume
- vmenter/vmresume syscall returns
- vmenter/vmresume inject vIRQ

VM resume
1. Introduction

2. Kernel interfaces

3. VM interface harmonization

4. VMMs harmonized

5. Conclusion
Control flow and exit handling on few instructions

- Multiple vCPUs, multiple EPs, multiple physical CPUs
Control flow and exit handling on few instructions

- Multiple vCPUs, multiple EPs, multiple physical CPUs

sel4 v9.0:

- Kernel fault on VMEnter by non vCPU thread → patch

No unrestricted guest support → patch

Scheduling bug if vCPU spins → starvation → patch

Kernel denies to boot on non VT-x platforms → patch

→ Working toy VMM on all 3 kernels

→ no AMD support by seL4
VMM unit test

Control flow and exit handling on few instructions

- Multiple vCPUs, multiple EPs, multiple physical CPUs

sel4 v9.0:

- Kernel fault on VMEnter by non vCPU thread → patch
- No unrestricted guest support → patch
Control flow and exit handling on few instructions

- Multiple vCPUs, multiple EPs, multiple physical CPUs

sel4 v9.0:
- Kernel fault on VMEnter by non vCPU thread → patch
- No unrestricted guest support → patch
- Scheduling bug if vCPU spins → starvation → patch
Control flow and exit handling on few instructions

- Multiple vCPUs, multiple EPs, multiple physical CPUs

**sel4 v9.0:**

- Kernel fault on VMEnter by non vCPU thread → patch
- No unrestricted guest support → patch
- Scheduling bug if vCPU spins → starvation → patch
- Kernel denies to boot on non VT-x platforms → patch

Working toy VMM on all 3 kernels → no AMD support by seL4
VMM unit test

Control flow and exit handling on few instructions
  - Multiple vCPUs, multiple EPs, multiple physical CPUs

sel4 v9.0:
  - Kernel fault on VMEnter by non vCPU thread → patch
  - No unrestricted guest support → patch
  - Scheduling bug if vCPU spins → starvation → patch
  - Kernel denies to boot on non VT-x platforms → patch
→ Working toy VMM on all 3 kernels
→ no AMD support by seL4
Replaced all NOVA specific parts
  - Simple Genode based guests for testing
Running again after few days on Genode/NOVA
Replaced all NOVA specific parts
  - Simple Genode based guests for testing
Running again after few days on Genode/NOVA

Various debugging sessions on Fiasco.OC and seL4
  → war stories (backup slides)
  → 1 kernel patch for seL4 and 1 for Fiasco.OC
Replaced all NOVA specific parts
  - Simple Genode based guests for testing
Running again after few days on Genode/NOVA

Various debugging sessions on Fiasco.OC and seL4
  → war stories (backup slides)
  → 1 kernel patch for seL4 and 1 for Fiasco.OC

State: kernel agnostic Seoul VMM on all 3 kernels
  - Guests: Genode VMs, Linux VM+network+SMP
  - seL4: kernel fault on Linux SMP VM → not investigated
VBox 5 VMM - current state

Work in progress - current state:
- Kernel agnostic VBox5 binary ready and runnable
- NOVA: simple Genode VMs running again
- seL4/Fiasco.OC: VM gets up, fails/hangs early

Known remaining challenges:
- Guest FPU state access required
  - Missing in VM interface
  - Support by seL4 and Fiasco.OC unclear
- seL4: no support for 64bit guests
Outline

1. Introduction
2. Kernel interfaces
3. VM interface harmonization
4. VMMs harmonized
5. Conclusion
Dare the impossible → possible*

- Restrictions depending on the kernel

Roadmap:
- Finish VBox5 adaptation
- Extend -hw- kernel with VT-x extensions
- Optional: support other platforms, e.g. ARM

Benefits:
- Portable VMMs across kernels
- Genode users have the ultimate kernel choice
Thank you

Genode OS Framework
https://genode.org

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

Stories around Genode
https://www.genodians.org

Genode Labs GmbH
https://www.genode-labs.com
Backup
Fiasco.OC:
- In-guest faults during protected → page mode transition
- reason: EFER status of host taken instead of guest
- Fiasco.OC: can be runtime configured → good

seL4:
- seL4: EFER register not saved on VMexit → kernel patch
CR* shadow/mask handling required on seL4 & Fiasco.OC
- Took some time, caused friction
- Open issue:
  - Kernels overwrites some bits in CR* to adhere to hardware requirements
  - Overridden bits not known/announced to VMM
  - Read back CR* modifications contains changes of hypervisor and VM mixed
    - Leads to various invalid guest states
  - Heuristics required - unexpected but manageable:
    - Job of Fiasco.OC/seL4 vs VMM?
Another test VM:

- seL4 and NOVA: worked fine
- Fiasco.OC: invalid guest state

Long long sessions of VM state diffs between kernels

- Happens on switch from protected → real mode

Source reason:

- \( v\text{IRQ} \) injection can not be reset by VMM on Fiasco.OC
- Patching kernel helps