Muen - An x86/64 Separation Kernel for High Assurance

Reto Buerki Adrian-Ken Rueegsegger

Institute for Internet Technologies and Applications
University of Applied Sciences Rapperswil

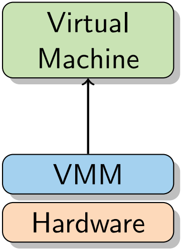
August 29, 2013

無縁

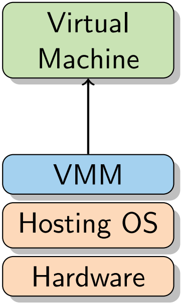
Outline

Virtualization

- Virtualization performed by *virtual machine monitor* (VMM)



(a) *Type I, native or bare metal VMM.* Runs directly on the hardware in the most privileged processor mode.



(b) *Type II or hosted VMM.* The VMM runs on top of a conventional operating system and uses OS services.

Intel Virtualization Technologies

- VT-x is Intel's technology for virtualization on the x86 platform
- Virtual machine state stored in virtual-machine control structure (VMCS)
- Virtual-machine extensions (VMX) provide CPU instructions to manage VMCS
- VMM runs in VMX root mode
- Virtual machines run in VMX non-root mode
- Hardware assisted virtualization simplifies implementation of VMM

SPARK

- Precisely defined programming language based on Ada
- Intended for writing high integrity and security software
- Program and proof annotations as Ada comments
- Allows proof of absence of runtime errors
- Allows partial proof of correctness
- Industrial usage in Avionics, Space, Medical Systems and Military

```
type Color_Type is (Red, Green, Blue);

procedure Exchange (X, Y: in out Color_Type);
--# derives X from Y &
--#         Y from X;
--# post X = Y~ and Y = X~;
```

Separation Kernel

- Concept introduced by John Rushby (1981)
- Partition system into multiple subjects which behave as if they were running on dedicated hardware
- Kernel must guarantee component separation
- Ideal as basis for a component-based system
- No channels for information flow between components other than those explicitly provided
- Partitioning and isolation of resources (CPU, memory, devices, . . .)
- Static configuration during integration
- Only includes necessary features $\rightarrow$ small TCB
- Well suited for formal verification

Motivation

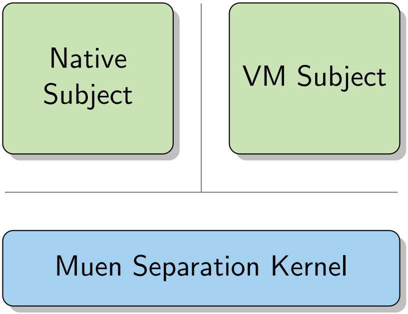
- Currently available (monolithic) systems unsuitable
- Implementation suitable for high assurance systems
- Increase confidence in systems built with COTS hardware
- Public sources and documentation enable third-party review
- Many advances in Intel hardware support for virtualization

Goals

- Open-source separation kernel (GPLv3+)
- Implementation in SPARK
- Proof of absence of runtime errors
- Small code size
- Reduction to essential functionality
- Leverage latest hardware features of Intel platform (VT-x, EPT, VT-d, . . .)
- Target platform is 64-bit Intel
- Only allow intended data flows
- Prevent or limit possible side-/covert channels

Architecture

- Kernel guarantees subject isolation
- Spatial isolation by memory management, VT-x
- Temporal isolation by scheduling



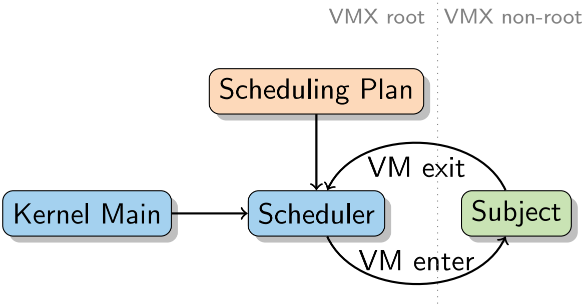
Policy

- Specifies system configuration
  - Hardware of target platform
  - Kernel configuration
  - Subject configuration
  - Scheduling plans
- skpolicy tool compiles XML to SPARK sources

```
<subject id="2" name="crypter" profile="native" cpu="2"
    pml4_address="270000" io_bitmap_address="274000"
    msr_bitmap_address="276000">
  &crypterinit;
  <memory_layout>
    &cryptermem;
    <!-- crypter request  page       -->
    <memory_region physical_address="29d000"
        virtual_address="10000" size="4k" alignment="4k"
         writable="false" executable="false" memory_type
        ="WB"/>
```

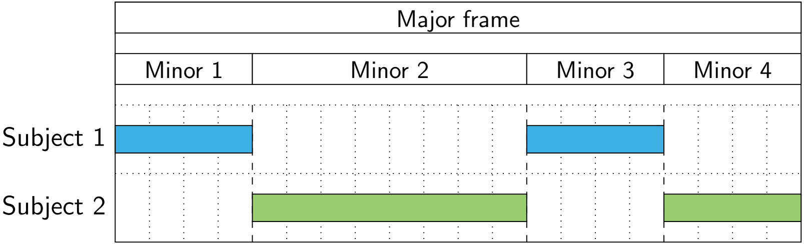
Scheduler I

- Fixed cyclic scheduler
- Use of VMX preemption timer



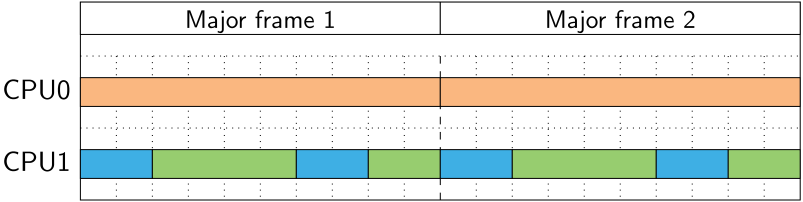
Scheduler II

- Major frame consisting of minor frames
- Minor frames specify subject and time slice in ticks
- Scheduling plan specifies minor frames per logical CPU
- $\tau 0$ subject can switch scheduling plan



Scheduler III

```
<major_frame>
    <cpu>
        <minor_frame subject_id="0" ticks="200"/>
    </cpu>
    <cpu>
        <minor_frame subject_id="1" ticks="40"/>
        <minor_frame subject_id="2" ticks="80"/>
        <minor_frame subject_id="1" ticks="40"/>
        <minor_frame subject_id="2" ticks="40"/>
    </cpu>
</major_frame>
```



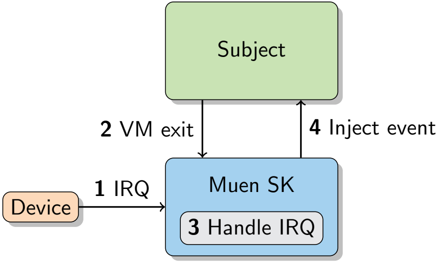
Traps

- Transition to VMX root mode is called a trap
- Policy specifies per-subject trap table
- Trap causes subject handover according to policy
- Trap may inject interrupt in destination subject
- Reserved traps are handled differently
  - VMX preemption timer
  - External interrupt
  - Interrupt window
  - Hypercall
- Virtualization using "Trap and Emulate"

```
<trap_table>
    <entry kind="*" dst_subject="sm" dst_vector="36"/>
</trap_table>
```

External Interrupts

- Policy assigns devices to subjects
- Setup of interrupt routing according to policy



1. External interrupts cause traps on designated CPU
2. Kernel adds pending event to destination subject
3. Pending events are injected on resumption of subject
4. Subject handles injected event as interrupt

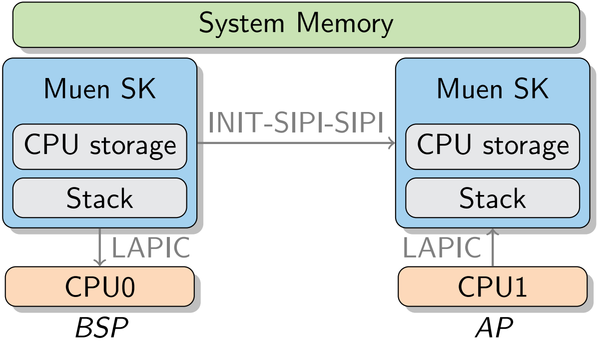
Event Handling

- Event is a hypercall triggered by subject using VMCALL instruction
- Policy specifies per-subject event table
- Handover events transfers execution to destination subject optionally injecting an interrupt
- Interrupt events inject interrupt in destination subject with optional IPI

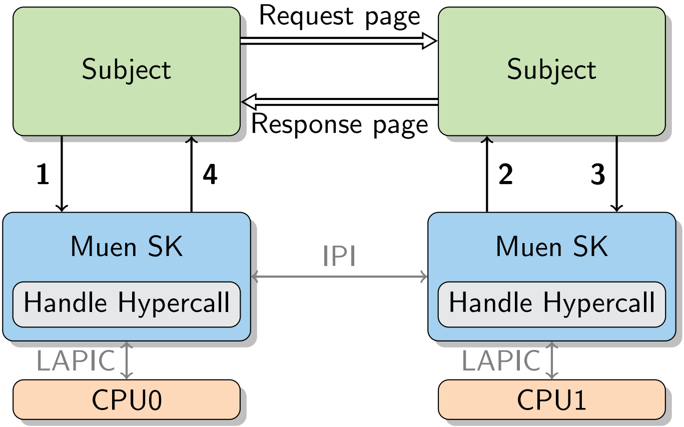
```
<event_table>
    <interrupt event="1" dst_subject="s2" dst_vector="33"
        send_ipi="true"/>
    <handover  event="2" dst_subject="s3"/>
</event_table>
```

Multicore

- Kernel starts on bootstrap processor (BSP)
- BSP starts application processors (APs)
- All CPUs synchronize on major frame changes

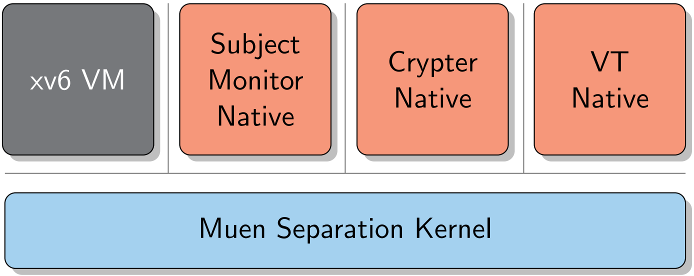


Inter-Core Events



Demo

- Untrusted VM subject running MIT's xv6 OS
- Native VT subject provides virtual terminals and keyboard
- Native subject monitor (SM) observes xv6 subject
  - Emulates port I/O
  - Halts xv6 on invalid operation
- Native crypter provides hashing service
  - Inter-subject communication using shared memory pages
  - Signalisation using event mechanism



VMX Controls - Exiting

Event	Native	VM
External interrupt	✓	✓
VMX preemption timer	✓	✓
Execute `INVLPG`	✓	✓
Execute `MONITOR`	✓	✓
Execute `MWAIT`	✓	✓
Execute `RDPMC`	✓	✓
Execute `RDTSC`	✓	✓
Execute `WBINVD`	✓	✓
MOV to CR3	✓	
MOV from CR3	✓	
MOV to CR8	✓	✓
MOV from CR8	✓	✓
MOV to/from debug registers	✓	✓
I/O port access	✓	✓
MSR access	✓	✓
Exceptions	✓	

System Resources

- Assigned to subjects according to policy
- Assignment is static at integration time

Memory

- Specified by memory regions in kernel/subjects spec
- Policy compiler creates page tables

Devices

- Assignment to subject grants resources (memory, ports, interrupts)
- Policy compiler
  - Maps memory regions into subject's address space
  - Enables I/O port access via VMCS I/O bitmap
  - Creates interrupt routing table entry

Execution Environment

Component	VMCS	State	Denied
General purpose registers		✓	
Segment registers	✓		
Instruction pointer	✓		
Flag register	✓		
CR0	✓		
CR2		✓	
CR3	✓		
CR4	✓		
CR8			✓
Descriptor table registers	✓		
DR0-3			✓
DR6			✓
DR7			✓
x87 FPU registers			✓
MMX registers			✓
XMM registers			✓
MSRs			(✓)

Temporal Isolation

- Fixed cyclic scheduler
- Static scheduling plan generated from policy
- Subject preemption using VMX preemption timer
- Sum of minor frame lengths per CPU/major frame are equal
- Global barrier sync at beginning of major frame

Results

- Minimal Zero-Footprint Run-Time (RTS)
- Kernel
- Tools
  - Policy compilation tool (`skpolicy`)
  - Config generation tool (`skconfig`)
  - Packaging tool (`skpacker`)
- Subjects
  - Initial $\tau 0$ implementation
  - Virtual terminals & keyboard
  - xv6 OS with minimal adjustments
  - Subject monitor for xv6
  - Crypter
  - Dumper
- Documentation

Results - Kernel

- Source code statistics:
  - ~260 lines of Assembly
  - ~2470 lines of SPARK
- Proof of absence of runtime errors (All VCs discharged)
- Static assignment of resources according to policy
- Multicore support
- EPT and memory typing (PAT)
- Event mechanism
- Support for native 64-bit subjects
- Support for VM subjects

Future Work

Mid-term

- Linux virtualization
- Hardware passthrough/PCIe virtualization
- Extend $\tau 0$
- Covert/Side-Channel analysis

Long-term

- MP subjects
- Fully virtualized subjects (Windows)
- Power Management
- Performance optimization
- Formal verification

Questions?

Thank you for your attention!