TECHNISCHE UNIVERSITÄT DRESDEN

Department of Computer Science Institute of Systems Architecture, Operating Systems Group

VIRTUALIZATION

Julian Stecklina (jsteckli@os.inf.tu-dresden.de)

Dresden, 2011/6/23

00 Goals

Give you an overview about:

- virtualization in general,
- hardware virtualization on x86,
- doing virtualization the microkernel-way.

We will *not* discuss:

- history [Go74],
- language runtimes,
- how to use XEN/KVM/. . .

00 Outline

01 Outline

01 Starting Point

You want to write a new operating system that is

- secure,
- trustworthy,
- small,
- fast,
- fancy.

but . . .

01 Commodity Applications

Users expect to run all the software they are used to ("legacy"):

- browsers,
- Word,
- iTunes,
- certified business applications,
- new (Windows/DirectX) and ancient (DOS) games.

Porting or rewriting all is *infeasible*!

01 Commodity Hardware

Users expect to run an (x86) OS on any PC. You need to support:

- input devices (USB, PS2, keyboards, mice, tablets, . . .)
- graphics adapters (ATI, Intel, nVidia, Matrox, SiS, . . .)
- disks (SATA, SAS, FC, IDE, USB, Firewire, . . .)
- network (Ethernet, ATM, Wifi, . . .)
- printer, scanner, webcams, . . .

Porting or rewriting all is *infeasible*!

01 One Solution: Virtualization

"By virtualizing a commodity OS [..] we gain support for legacy applications, and devices we don't want to write drivers for."

"All this allows the research community to finally escape the straitjacket of POSIX or Windows compatibility [...]"

[Ro07]

01 Definitions

A virtual machine is defined to be an

> "efficient, isolated duplicate of a real machine." [Po74]

The software that provides this illusion is the *Virtual Machine Monitor* (VMM, mostly used synonymous with *Hypervisor*).

01 Virtualizable?

. . . is a property of the *Instruction Set Architecture* (ISA). Instructions can be categorized.

A *sensitive* instruction

- changes the configuration or mode of the processor, or
- depends in its behavior on the processor's state.

A *privileged* instruction

- causes a trap

in user mode.

Instructions that are neither are called *innocuous*.

01 Trap & Emulate

If all sensitive instructions are privileged,
a VMM can be written. [Po74]

The Trap&Emulate idea is to

- execute guest in separate address space in unprivileged mode,
- emulate all instructions that cause traps.

02 Outline

02 Virtualization Holes

x86 includes non-privileged sensitive instructions, such as

- popf,
- mov/pop (load segment registers),
- mov (writing page tables, IO, . . .),
- . . .

Straight-forward trap&emulate not possible on x86!

02 Fixing Virtualization Holes: Emulation

Emulate the virtual CPU instruction by instruction:

- independent of host CPU, but
- terribly slow.

Speed up emulation by

- binary translation.

02 VMware Workstation: Binary Translation

First commercial virtualization solution for x86, introduced in ˜1999. Overcame limitations of the x86 architecture:

- translate problematic instructions into appropriate calls to the VMM
- can avoid costly traps for privileged instructions

Binary translation can provide decent performance [Ro05, Ba08] but:

- requires complex runtime translation engine (self-modifying code. . .)

Other examples: KQemu, Virtual Box, Valgrind

02 Paravirtualization

Why all the trouble? Just "port" a guest operating system to the interface of your choice.
Paravirtualization can

- provide better performance,
- simplify VMM [Li10]

but at a maintainance cost and you need the source code!

Compromise: Use paravirtualized drivers for I/O performance (KVM virtio, VMware).

Examples are MkLinux, L4Linux, Xen, ...

02 Pre-virtualization

Paravirtualization can also be semiautomated [Le05].

- Sensitive instructions are automatically identified (in compiler output).
- Sensitive memory access needs to manually identified.
- Leave markers in binary.
- On VM load-time, VMM replaces instructions with emulation code.
- In-Place VMM translates to hypervisor calls.

Benefits:

- less effort than plain paravirtualization,
- comparable speed.

02 Hardware Support for Virtualization

Late Pentium 4 (2005) introduced hardware support for virtualization: Intel VT.
(AMD-V is conceptually very similar)

- root mode vs. non-root mode
  - root mode runs hypervisor
  - non-root mode runs guest
- situations that Intel VT cannot handle trap to root mode (*VM Exit*)
- special memory region (VMCS) holds host/guest state
- reduced software complexity

Supported by all major virtualization solutions today.

Intel VT-x (*Vanderpool*)

- introduces *Root* and *non-Root* modes
- *VMCS* holds VM state, accessed via `vmread`, `vmwrite` instructions
- `vmlaunch`, `vmresume`, ...

AMD-V (SVM/*Pacifica*)

- introduces *Host* and *Guest* modes
- *VMCB* holds VM state, accessed as normal memory page
- `vmrun`, ...

02 Virtualization with Hardware Support

The general idea is

- guest OS executes until it traps to the hypervisor (VM Exit),
- hypervisor inspects state, handles event, resumes VM.

02 Handling VM Exits

Some cases are simple, such as `rdmsr`:

- store value of virtual MSR in `EDX:EAX`,
- advance `EIP`.

Some very difficult/expensive, such as MMIO:

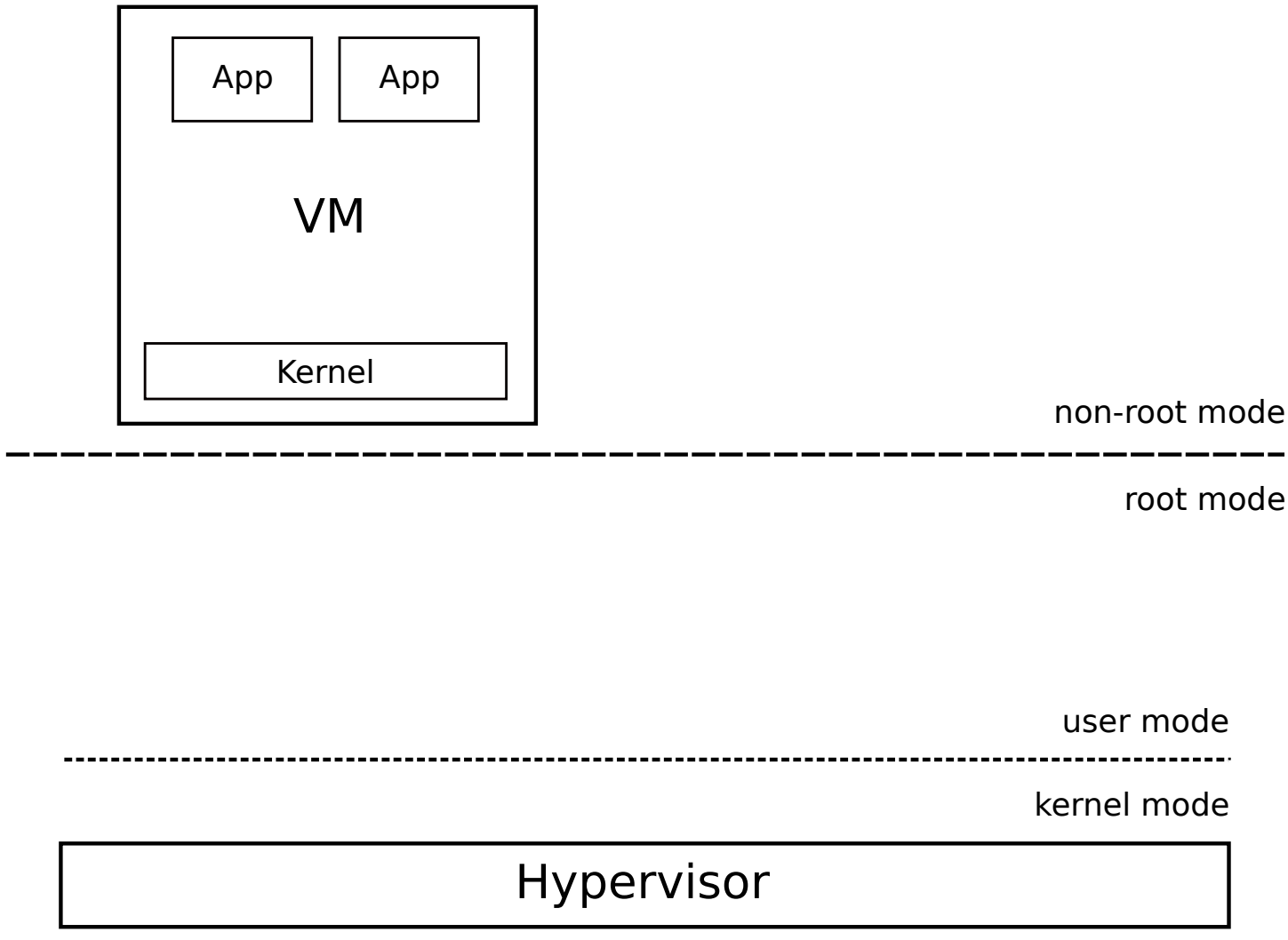
- not enough information provided by hardware
- unknown whether it was an `mov`, `and`, ...

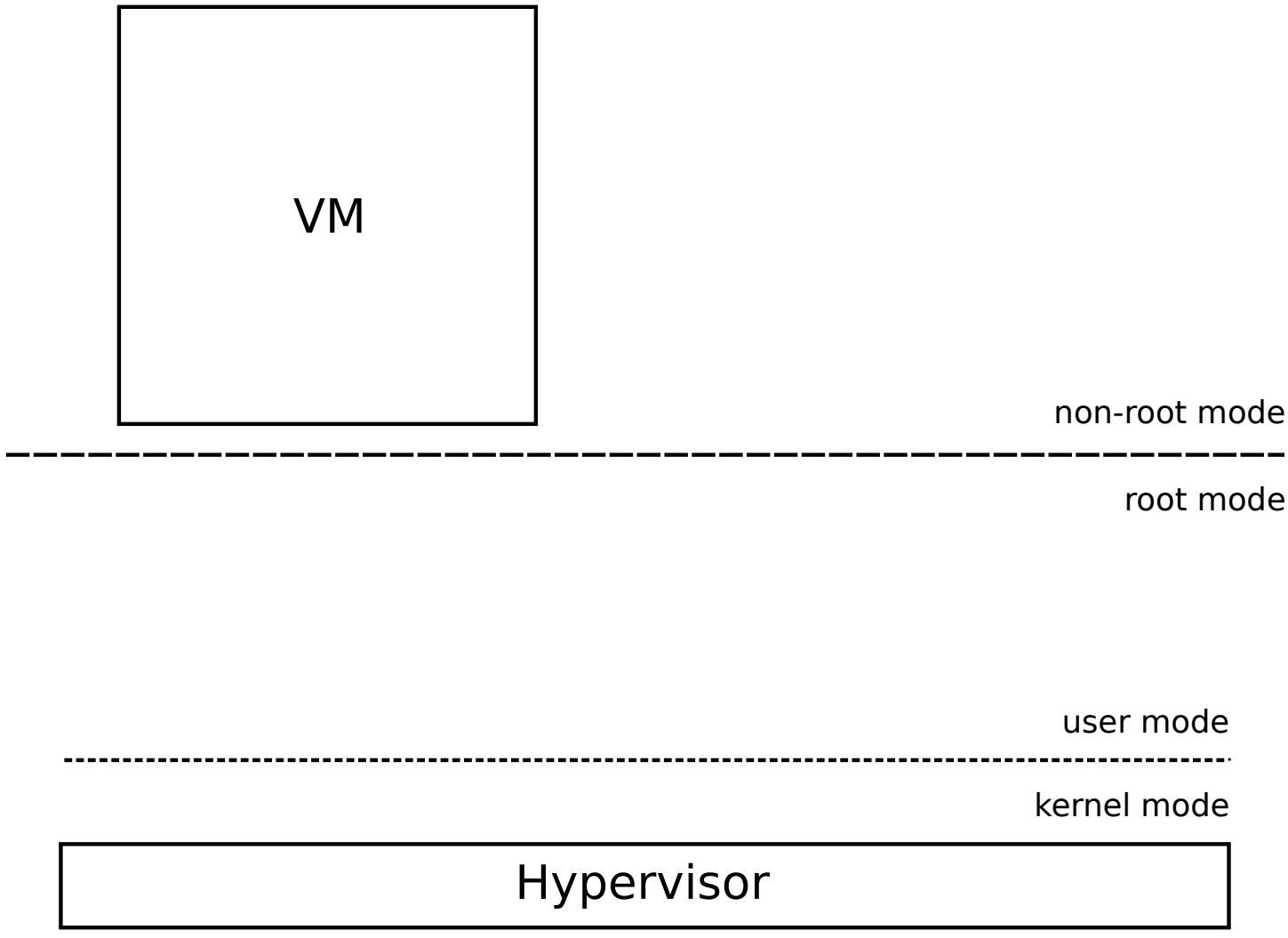
02 Instruction Emulator

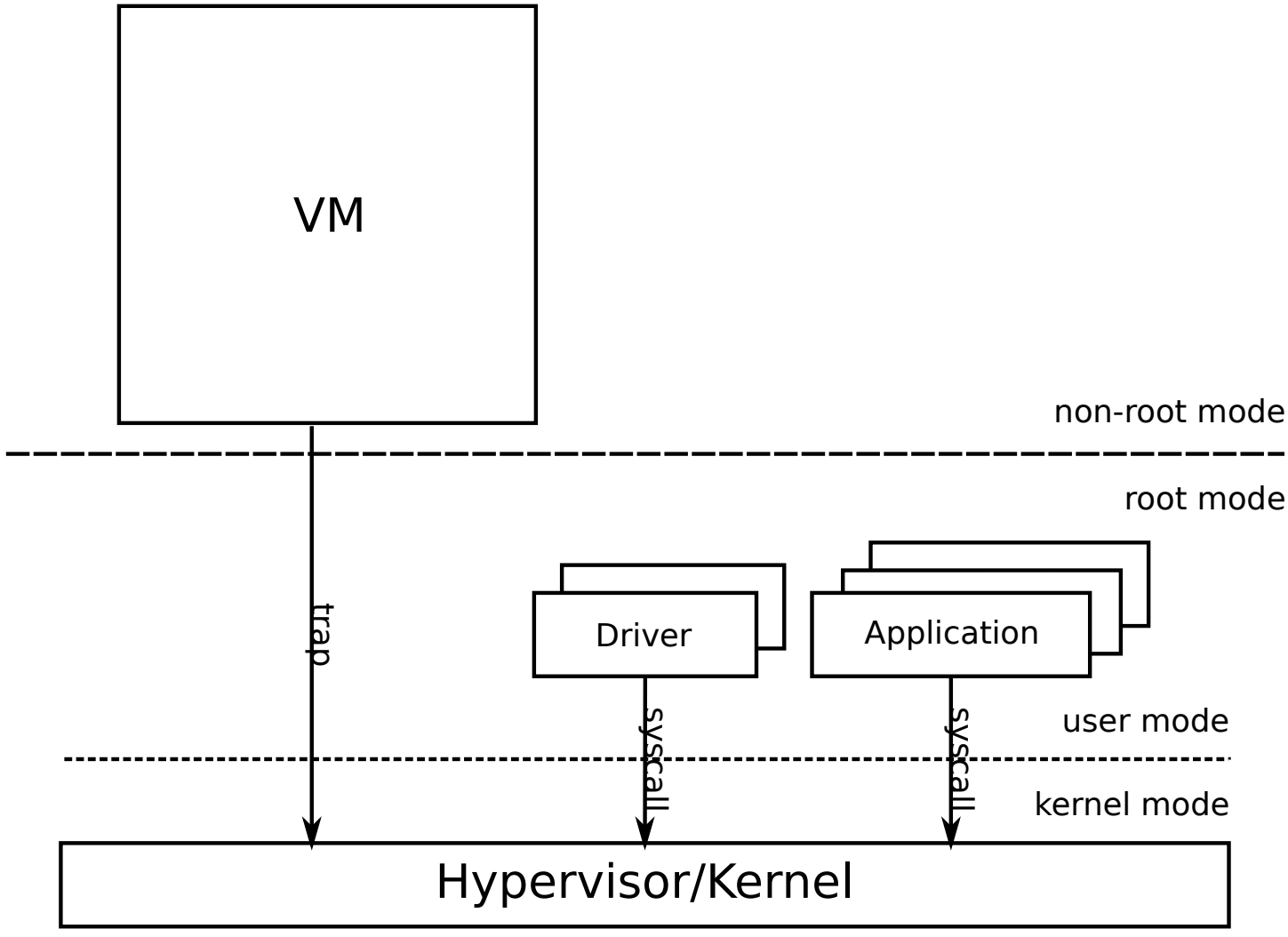
Intel VT and AMD-V still require an instruction emulator, e.g. for

- running 16-bit code (not in AMD-V, latest Intel VT),
  - BIOS
  - boot loaders
- handling MMIO (need to emulate instruction that caused a page fault)
  - realized as non-present page in guest physical mode
  - emulate offending instruction
- ...

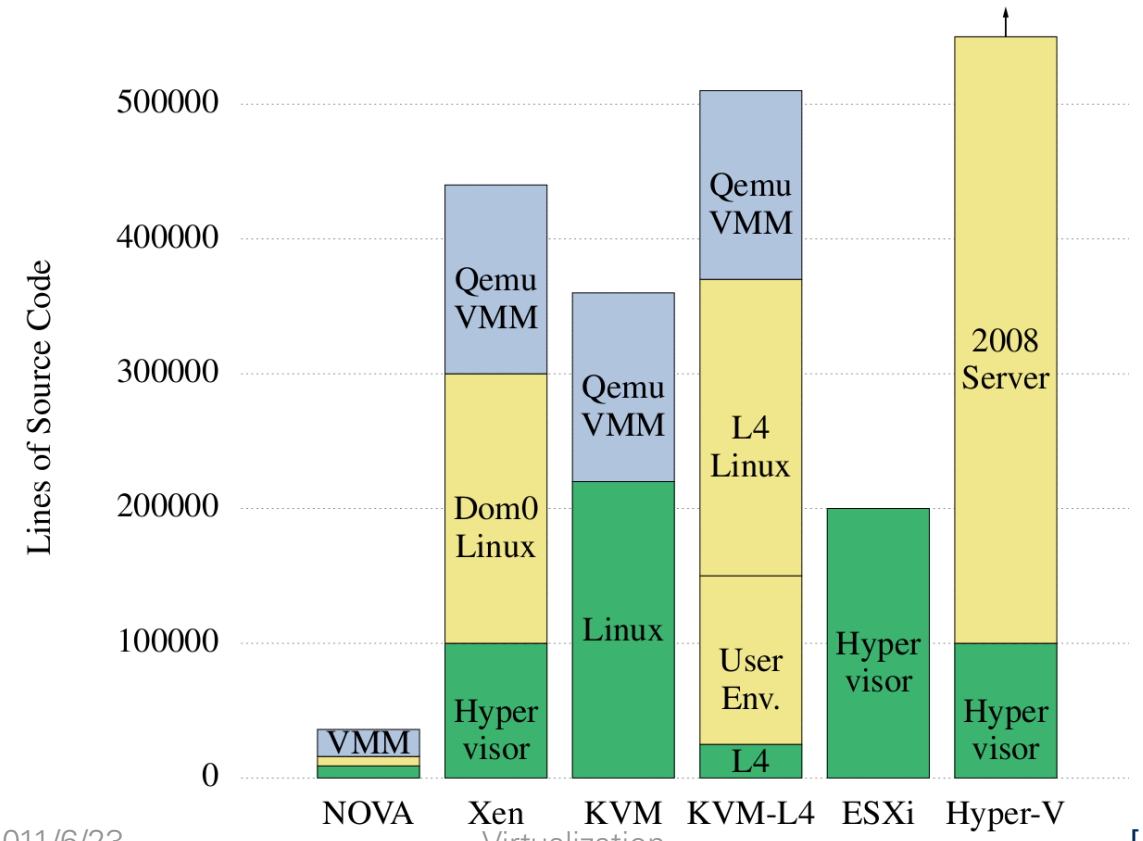
03 Outline


App
App
VM
Kernel
non-root mode
root mode
user mode
kernel mode
Hypervisor


VM
non-root mode
root mode
user mode
kernel mode
Hypervisor


VM
non-root mode
root mode
trap
Driver
Application
syscall
syscall
user mode
kernel mode
Hypervisor/Kernel

03 Complexity is Bad

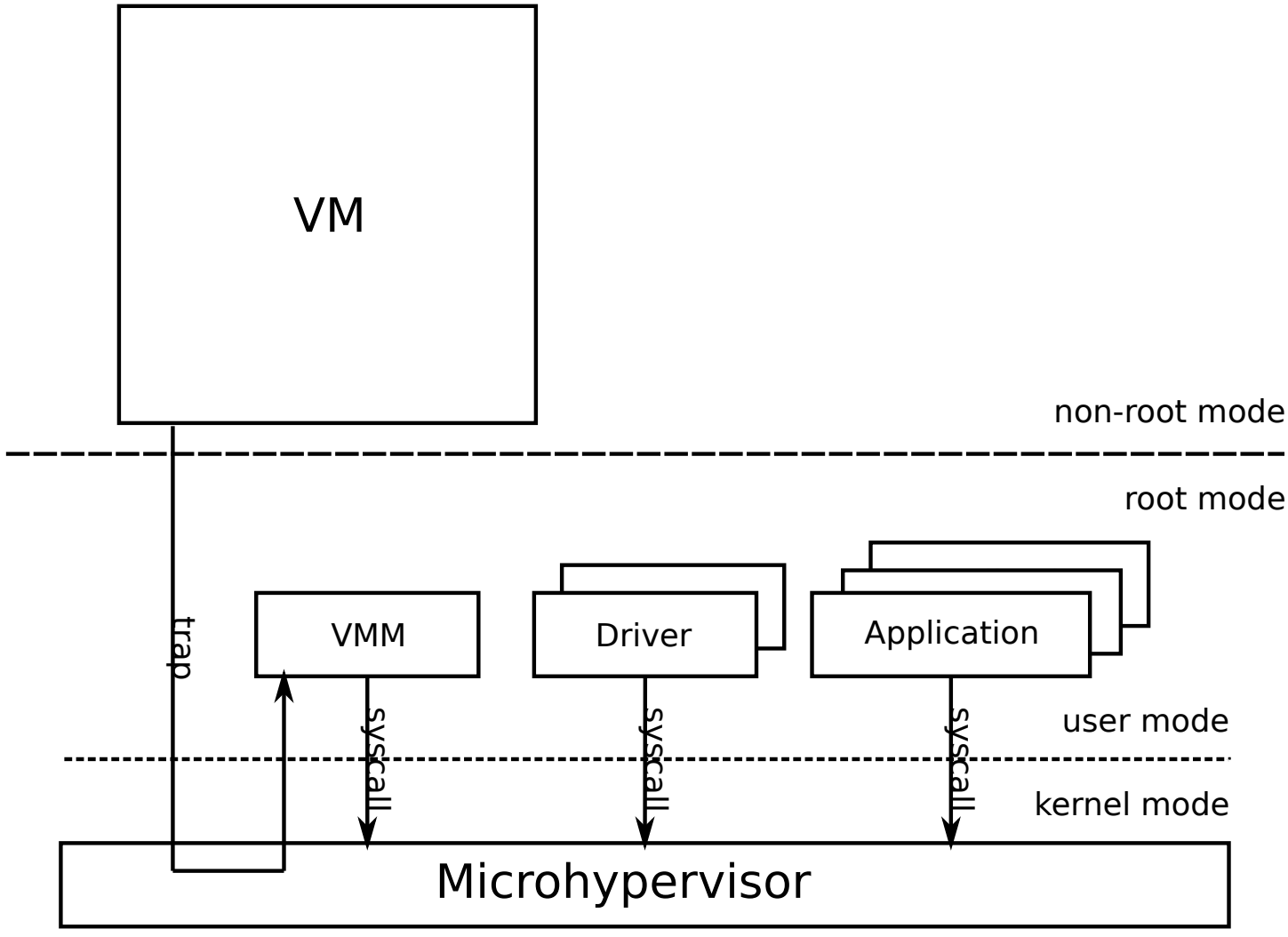


[St10]

03 Secunia Advisory SA25073 (Qemu)

`http://secunia.com/advisories/25073/`

- "The size of ethernet frames is not correctly checked against the MTU before being copied into the registers of the NE2000 network driver. This can be exploited to cause a heap-based buffer overflow."
- " An error within the handling of the aam instruction can result in a division by zero."
- . . .


VM
non-root mode
root mode
trap
VMM
Driver
Application
syscall
syscall
syscall
user mode
kernel mode
Microhypervisor

03 What needs to be in the Microhypervisor?

Ideally nothing, but

- VT-x instructions are privileged:
  - starting/stopping a VM
  - access to VMCS
- hypervisor has to validate guest state to enforce isolation.

03 Microhypervisor vs. VMM

We make a distinction between both terms [St10, Ag10].

Microhypervisor

- "the kernel part"
- provides isolation
- mechanisms, no policies
- enables safe access to virtualization features to userspace

VMM

- "the userland part"
- CPU emulation
- device emulation

03 State Transfer (Shared Memory)

Because of splitted Hypervisor and VMM, guest state must be transferred between the two. We could use *shared memory*:

- share VMCB (AMD-V)
- provide copy of VMCS to VMM (VT-x)
- VMM modifies shared memory during emulation

But hypervisor has to *verify the complete state* every time the VMM had access. Reading out VMCS is *expensive*.

03 Selective State Transfer

Depending on VM Exit reason the VMM specifies which state it needs and what it changes:

cpuid (**EIP**, instruction length, **EAX**) → (**EIP**, EAX, ECX, EDX, EBX)

03 Device Emulation

Given a decoded MMIO access, what now?

- Guest OS expects access to hardware
- VMM must provide virtual hardware:
  - implement hardware state machine in software (*device model*)

03 Access to Devices

I/O Ports

- separate 16-bit address space for devices
- accessed via `in`/`out` instructions
- ye legacy way

MMIO

- device registers are mapped into (uncachable) physical memory
- accessed via `mov`, ...
- needs instruction emulation

03 Example: PIC 8259

```
case 0x20: // outb to port 0x20
 if (value & 0x10) {
   _icw[ICW1] = value;
   _icw_mode = ICW2;
 }
...
```

```
case 0x21: // outb to port 0x21
 switch (_icw_mode) {
   case ICW2:
    _icw[ICW2] = value & 0xf8;
    _icw_mode = ICW3;
    break;
   case ICW3: ...
   case ICW4: ...
   case OCW1:
    _imr = value;
    propagate_irq(true);
    break;
  }
```

03 VMM: Needed Device Models

For a reasonably useful VMM, you need

- Timer: PIT, RTC, HPET, PMTimer
- Interrupt Controller: PIC, LAPIC, IOAPIC
- PCI hostbridge
- keyboard, mouse, VGA
- network
- SATA or IDE disk controller

But then you still cannot run a VM . . .

03 VMM: Virtual BIOS

VMM needs to emulate (parts of) BIOS:

- memory layout
- screen output
- keyboard
- disk access
- ACPI tables

Mostly used for bootloaders and early platform discovery (memory layout).

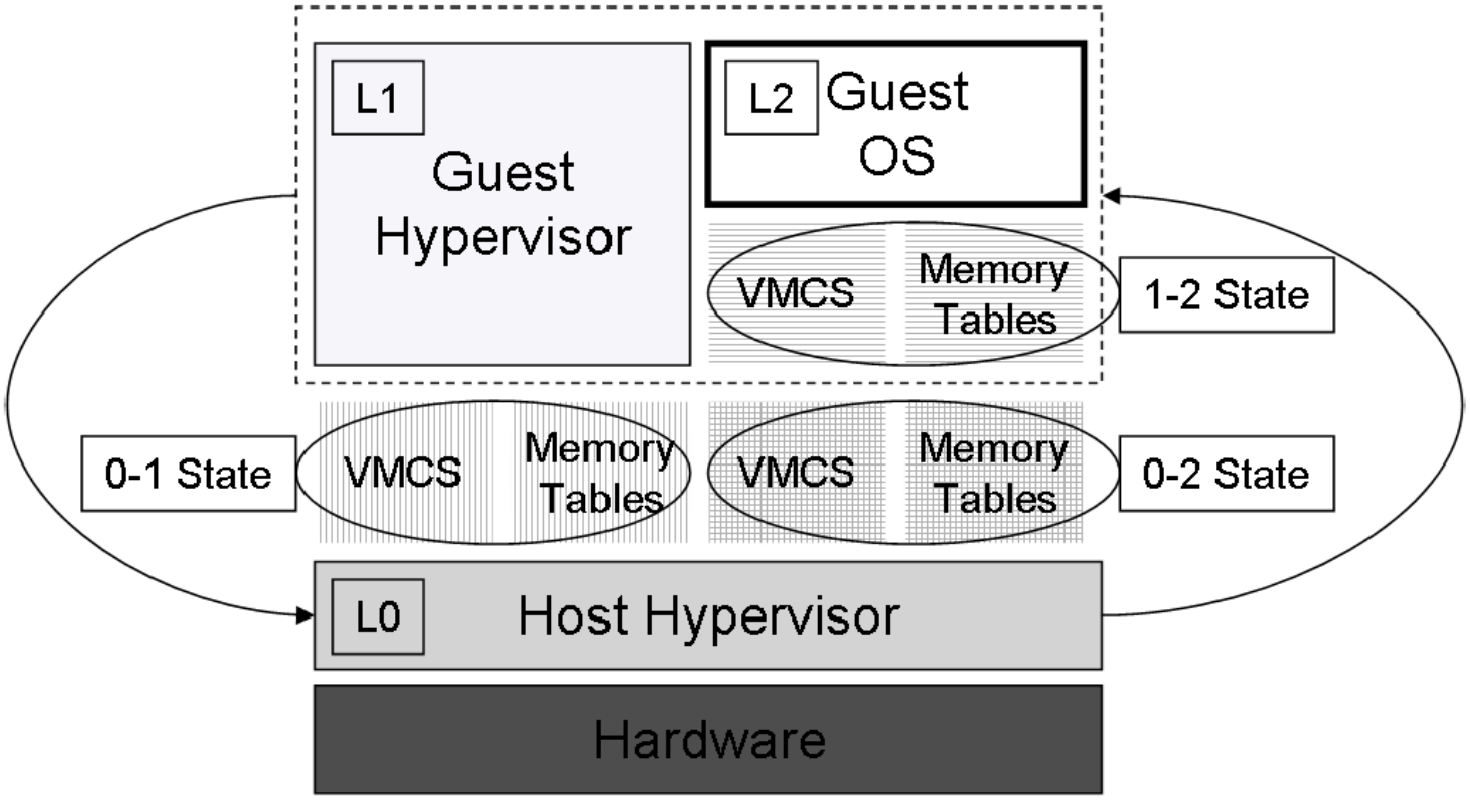
03 Further Topics

- *nested virtualization*
- *driver reuse*
- PCI passthrough
- Shadow Paging vs. Nested Paging
- IRQ handling
- Automatically generating the instruction decoder
- . . .

03 Nested Virtualization

Commodity OSes use virtualization for day-to-day operation (e.g. Windows 7 runs a XP VM). A hypervisor that wants to virtualize Windows 7 needs to support *nested virtualization*:

- the guest OS needs to have the illusion of running in root mode.
- VT-x/AMD-V only support single-level virtualization.


L1
Guest Hypervisor
L2
Guest OS
VMCS
Memory Tables
1-2 State
0-1 State
VMCS
Memory Tables
VMCS
Memory Tables
0-2 State
L0
Host Hypervisor
Hardware

[Be10]

03 Nested Virtualization

There several difficulties:

- Multidimensional-paging, although the hardware understands only 1 or 2 dimensions,
- need to merge VMCS state on `vmread`/`vmwrite`,
- VM exits become increasingly expensive.

But results [Be10] are encouraging:

- 14.5$ overhead for kernbench in a nested VM

04 Outline

04 Open Problems

- efficient device virtualization
- scalable virtualization (SMP VMs)
- driver VMs
- time virtualization
- automatically testing instruction emulators and device models
- ...

04 Recap: NOVA Architecture

Reduce TCB of each VM and application:

- hypervisor provides low-level protection domains
  - address spaces
  - virtual machines
- VM exits are relayed to VMM as IPC with selective guest state,
- one VMM per guest in (root mode) userspace,
  - possibly specialized VMMs to reduce attack surface
  - only one generic VMM implement so far

04 Next Weeks

There will be hands-on seminars starting next week in

INF E0??

05 Outline


Timothy Roscoe et al.
Hype and Virtue
https://portal.acm.org/citation.cfm?id=1361401


Robert P. Goldberg, 1974
Survey of Virtual Machine Research
http://cseweb.ucsd.edu/classes/wi08/cse221/papers/goldberg74.pdf


Gerald J. Popek, Robert P. Goldberg, 1974
Formal requirements for virtualizable third generation architectures
https://portal.acm.org/citation.cfm?id=361073


Udo Steinberg, Bernhard Kauer, 2010
NOVA: A Microhypervisor-Based Secure Virtualization Architecture
http://os.inf.tu-dresden.de/papers_ps/steinberg_eurosys2010.pdf


Joshua LeVasseur et al, 2005
Pre-Virtualization: Slashing the Cost of Virtualization
http://www.l4ka.org/downloads/publ_2005_levasseur-ua_cost-of-virtualization.pdf


Sorav Bansal and Alex Aiken, 2008
Binary Translation Using Peephole Superoptimizers

http://theory.stanford.edu/~sbansal/pubs/osdi08_html/index.html

M. Rosenblum and T. Garfinkel, 2005
Virtual Machine Monitors: Current Technology and Future Trends
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1430630

Steffen Liebergeld, 2010
Lightweight Virtualization on Microkernel-based Systems
http://os.inf.tu-dresden.de/papers_ps/liebergeld-diplom.pdf

Muli Ben-Yehuda et al, 2010
The turtles project: Design and implementation of nested virtualization
http://www.usenix.org/events/osdi10/tech/full_papers/Ben-Yehuda.pdf

Ole Agesen et al, 2010
The evolution of an x86 virtual machine monitor
http://portal.acm.org/citation.cfm?id=1899930