TECHNISCHE UNIVERSITÄT DRESDEN

MOS - VIRTUALIZATION

Tobias Stumpf, Marcus Hähnel

WS 2017/18

Goals

Give you an overview about:

- virtualization and virtual machines in general,
- hardware virtualization on x86,
- our research regarding virtualization.

We will not discuss:

- lots and lots of details,
- language runtimes,
- how to use XEN/KVM/...

What is Virtualization?
Outline

What is Virtualization?

Very Short History

Virtualization on x86

Example: L4Linux

Example: NOVA

Example: Karma VMM

What is Virtualization?

Starting Point

You want to write a new operating system that is

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

but . . .

What is Virtualization?

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!

What is Virtualization?

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 [...]"

Roscoe:2007:HV:1361397.1361401

What is Virtualization?
Virtualization

virtual existing in essence or effect though not in actual fact

http://wordnetweb.princeton.edu

“All problems in computer science can be solved by another level of indirection.”

David Wheeler

What is Virtualization?
Emulation

Suppose you develop for a system G (guest, e.g. an ARM-based phone) on your workstation H (host, e.g., an x86 PC). An emulator for G running on H precisely emulates G's

- CPU,
- memory subsystem, and
- I/O devices.

Ideally, programs running on the emulated G exhibit the same behaviour as when running on a real G (except for timing).

What is Virtualization?

Emulation (cont'd)

The emulator

- simulates every instruction in software as it is executed,
- prevents direct access to H's resources from code running inside G,
- maps G's devices onto H's devices,
- may run multiple times on H.

What is Virtualization?

Mapping G to H

Both systems may have considerably different

- instructions sets and
- hardware devices

making emulation slow and complex (depending on emulation fidelity).

What is Virtualization?

G = H

If host and emulated hardware architecture is (about) the same,

- interpreting every executed instruction seems not necessary,
- near-native execution speed should be possible.

This is (easily) possible, if the architecture is virtualizable.

What is Virtualization?

From Emulation to Virtualization

A **virtual machine** is defined to be an

> "efficient, isolated duplicate of a real machine."

Popek:1974:FRV:361011.361073

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

What is Virtualization?

Idea: Executing the guest as a user process

Just run the guest operating system as a normal user process on the host. A virtual machine monitor process needs to handle:

What is Virtualization?

Idea: Executing the guest as a user process

Just run the guest operating system as a normal user process on the host. A virtual machine monitor process needs to handle:

- address space changes,
- device accesses,
- system calls,
- ...

Most of these are not problematic, because they trap to the host kernel (SIGSEGV).

TECHNISCHE UNIVERSITÄT DRESDEN

What is Virtualization?

A hypothetical instruction: OUT

Suppose our system has the instruction `OUT` that writes to a device register in kernel mode.

How should it behave in user mode?

Option 1:
Just do nothing.

Option 2:
Cause a trap to kernel mode.

What is Virtualization?

A hypothetical instruction: OUT

Suppose our system has the instruction `OUT` that writes to a device register in kernel mode.

How should it behave in user mode?

Option 1:
~~Just do nothing.~~

Option 2:
Cause a trap to kernel mode.

Otherwise device access cannot be (easily) virtualized.

What is Virtualization?

Virtualizable?

…is a property of the Instruction Set Architecture (ISA). Instructions are divided into two classes:

A sensitive instruction

- changes or
- depends in its behavior

on the processor's configuration or mode.

A privileged instruction causes a trap (unconditional control transfer to privileged mode) when executed in user mode.

What is Virtualization?

Trap & Emulate

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

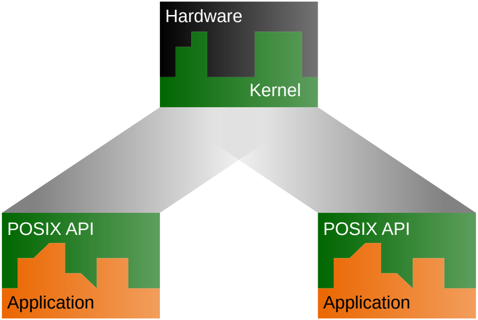
- execute guest in unprivileged mode,
- emulate all instructions that cause traps.

What is Virtualization?

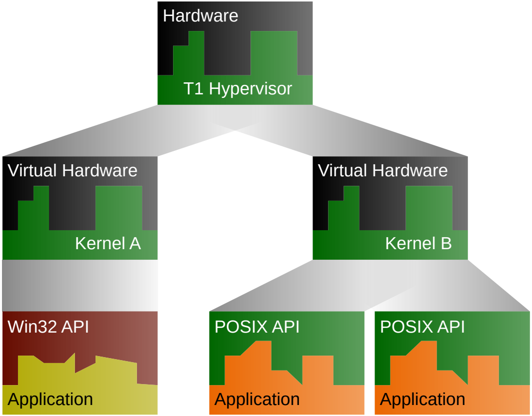
Trap & Emulate (cont'd)

Formal Requirements for
Virtualizable Third-Generation Architectures
`http://portal.acm.org/citation.cfm?id=361073`

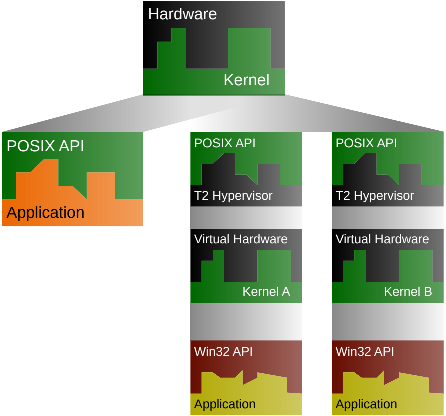
What is Virtualization?
Where to put the VMM?



What is Virtualization?



What is Virtualization?



What is Virtualization?

Type 1 vs. Type 2

Type 1 are implemented on the bare metal (bare-metal hypervisors):

- no OS overhead
- complete control over host resources
- high maintainance effort

Popular examples are

- Xen,
- VMware ESXi.

What is Virtualization?

Type 1 vs. Type 2 (cont'd)

Type 2 run as normal process on top of an OS (hosted hypervisors):

- doesn't reinvent the wheel
- performance may suffer
- usually need kernel support for access to CPU's virtualization features

Popular examples are

- KVM,
- VMware Server/Workstation,
- VirtualBox,
- ...

What is Virtualization?

Paravirtualization

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

- provide better performance,
- simplify VMM

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

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

Examples are Usermode Linux, L4Linux, Xen/XenoLinux, DragonFlyBSD VKERNEL, ...

What is Virtualization?

Reimplementation of the OS Interface

Why deal with the OS kernel at all? Reimplement its interface! E.g. wine reimplements (virtualizes) the Windows ABI.

- Run unmodified Windows binaries.
- Windows API calls are mapped to Linux/FreeBSD/Solaris/MacOS X equivalents.
- Huge moving target!

Can also be used to recompile Windows applications as native applications linking to winelib $\Rightarrow$ API “virtualization”

What is Virtualization?

Recap

- Virtualization is an overloaded term. Classification criteria:
  - Target
    real hardware, OS API, OS ABI, ...
  - Emulation vs. Virtualization
    Interpret some or all instructions?
  - Guest Modifications?
    Paravirtualization

TECHNISCHE UNIVERSITÄT DRESDEN

What is Virtualization?

Recap (cont'd)

- A (Popek/Goldberg) Virtual Machine is an
  - efficient,
  - isolated
  - duplicate of a real machine.
- The software that implements the VM is the Virtual Machine Monitor (hypervisor).
- Type 1 (“bare-metal”) hypervisors run as kernel.
- Type 2 (“hosted”) hypervisors run as applications on a conventional OS.

Very Short History

Outline

Very Short History

"Virtual machines have finally arrived. Dismissed for a number of years as merely academic curiosities, they are now seen as cost-effective techniques for organizing computer systems resources to provide extraordinary system flexibility and support for certain unique applications."

Very Short History

> "Virtual machines have finally arrived. Dismissed for a number of years as merely academic curiosities, they are now seen as cost-effective techniques for organizing computer systems resources to provide extraordinary system flexibility and support for certain unique applications."

Goldberg:1974:SVM:2412365.2412592

Very Short History

Early History: IBM

Erik Pitti, CC-BY, `www.flickr.com/people/24205142@N00`

TECHNISCHE UNIVERSITÄT DRESDEN

Very Short History

Early History: IBM

Virtualization was pioneered with IBM's CP/CMS in $\sim$ 1967 running on System/360 and System/370:

CP Control Program
provided System/360 virtual machines.

CMS Cambridge Monitor System (later Conversational Monitor System)
single-user OS.

At the time more flexible and efficient than time-sharing multi-user systems.

Very Short History

Early History: IBM (cont'd)

CP encodes guest state in a hardware-defined format.

SIE
: Start Interpretive Execution (instruction) runs the VM until a trap or interrupt occurs. CP resume control and handles trap.

CP provides:

- memory protection between VMs,
- preemptive scheduling.

Gave rise to IBM's VM line of operating systems.
First release: 1972
Latest release: z/VM 6.4 (November 11, 2016)

Very Short History

Virtualization is Great

- Consolidation
  - improve server utilization
- Isolation
  - isolate services for security reasons or
  - because of incompatibility
- Reuse
  - run legacy software
- Development

…but was confined to the mainframe world for a very long time.

. . . fast forward to the late nineties . . .

TECHNISCHE UNIVERSITÄT DRESDEN

Virtualization on x86

Outline

Virtualization on x86

Is x86 Virtualizable?

x86 has several virtualization holes that violate Popek&Goldberg requirement.

- Possibly too expensive to trap on every privileged instruction.
- `popf` (pop flags) silently ignores writes to the Interrupt Enable flag in user mode. Should trap!

Virtualization on x86

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 on the fly
- can avoid costly traps for privileged instructions

Provided decent performance but:

- requires complex runtime translation engine

Other examples: KQemu, Virtual Box, Valgrind

Virtualization on x86

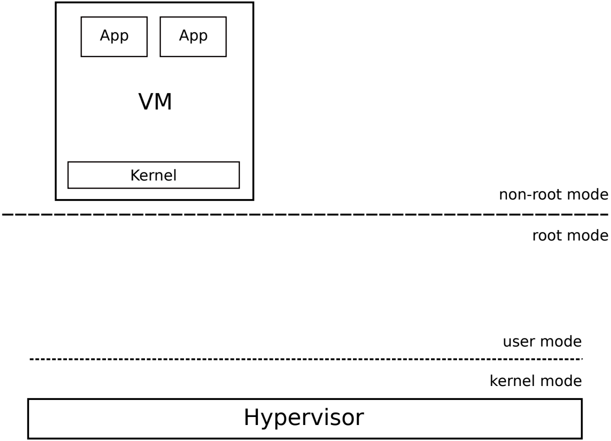
Hardware Support for Virtualization

Late Pentium 4 (2004) introduced hardware support for virtualization: Intel VT. (AMD-V is conceptually very similar)

- root mode vs. non-root mode
  - duplicates x86 protection rings
  - 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 guest state
- reduced software complexity

Supported by all major virtualization solutions today.

Virtualization on x86



Virtualization on x86

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 memory-mapped IO (need to emulate instruction that caused a page fault)
  - realized as non-present page
  - emulate offending instruction
- ...

Virtualization on x86

MMU Virtualization

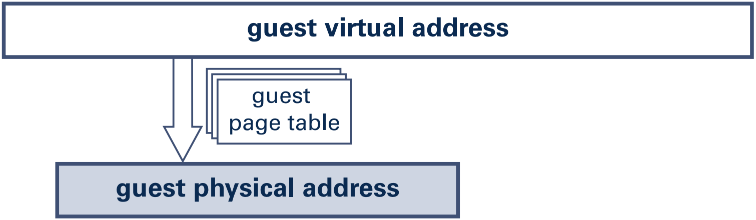
Early versions of Intel VT do not completely virtualize the MMU. The VMM has to handle guest virtual memory.

Four different types of memory addresses:

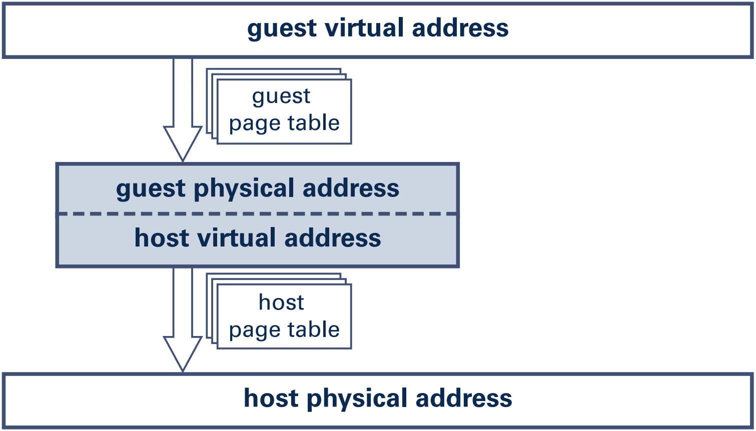
- hPA Host Physical Address
- hVA Host Virtual Address
- gPA Guest Physical Address
- gVA Guest Virtual Address

Usually gPA = hVA or other simple mapping (offset).

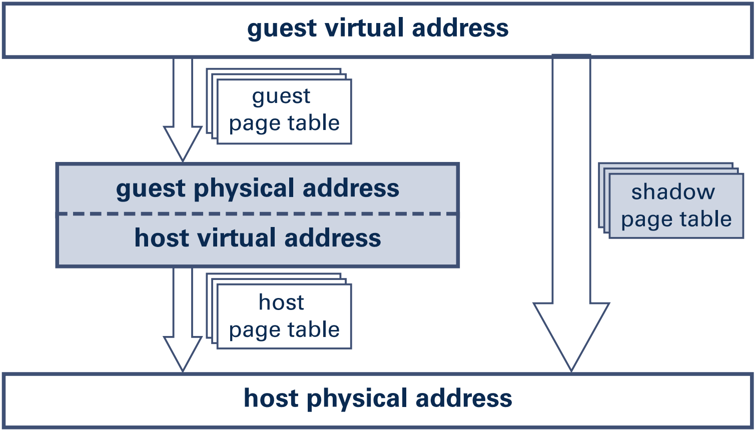
Virtualization on x86



Virtualization on x86



Virtualization on x86



Virtualization on x86

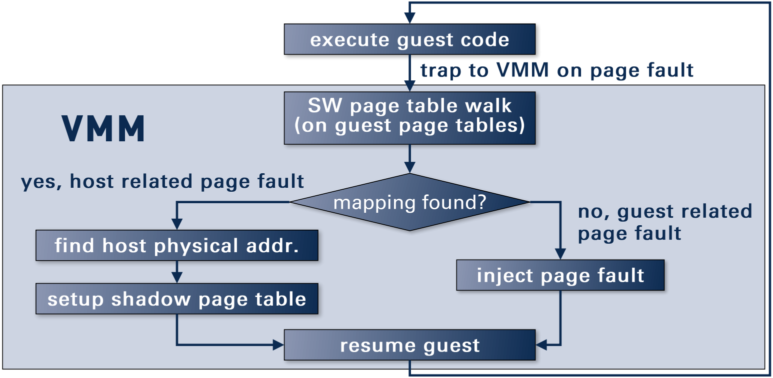
Shadow Page Tables

If the hardware can handle only one page table, the hypervisor must maintain a shadow page table that

- merges guest and host page table (maps from GVA to HPA),
- must be adapted on changes to virtual memory layout.

Virtualization on x86

Shadow Paging in a Nutshell



Virtualization on x86

Drawbacks of Shadow Paging

Maintaining Shadow Page Tables causes significant overhead, because they need to be updated or recreated on

- guest page table modification,
- guest address space switch.

Certain workloads are penalized.

Virtualization on x86

Nested Paging

Introduced in the Intel Nehalem (EPT) and AMD Barcelona (Nested Paging) microarchitectures, the CPU can handle

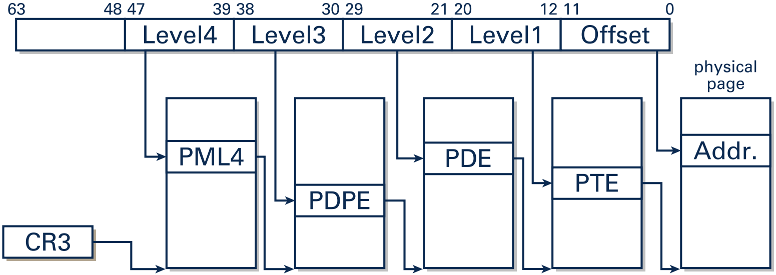
- guest and
- host page table

at the same time. Can reduce VM Exits by two orders of magnitude, but introduces

- measurable constant overhead ($< 1\%$)

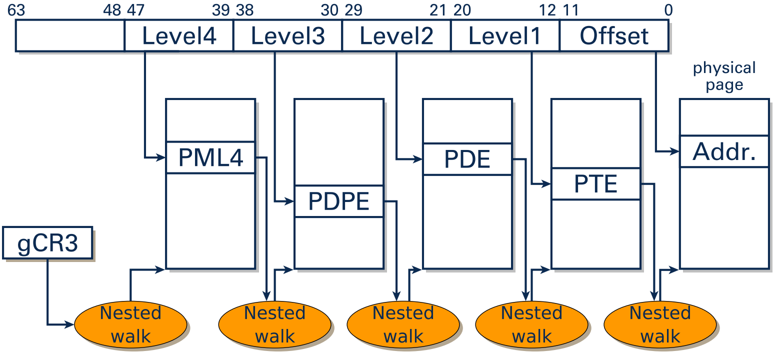
Virtualization on x86

Native Address Translation



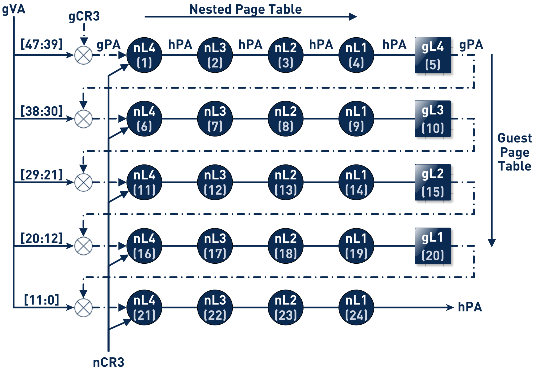
Virtualization on x86

Guest Address Translation



Virtualization on x86

2D Page Table Walk



Virtualization on x86

Nested Paging - Linux Kernel Compile Time

Event	Shadow Paging	Nested Paging
vTLB Fill	181,966,391	
Guest Page Fault	13,987,802	
CR Read/Write	3,000,321	
vTLB Flush	2,328,044	
INVLPG	537,270	
Hardware Interrupts	239,142	174,558
Port I/O	723,274	610,589
Memory-Mapped I/O	75,151	76,285
HLT	4,027	3,738
Interrupt Window	3,371	2,171
Sum	202,864,793	867,341
Runtime (seconds)	645	470
Exit/s	314,519	1,845

Steinberg:2010:NMS:1755913.1755935

Example: L4Linux

Outline

Example: L4Linux

L4Linux

. . . is a paravirtualized Linux first presented at SOSP'97 running on the original L4 kernel.

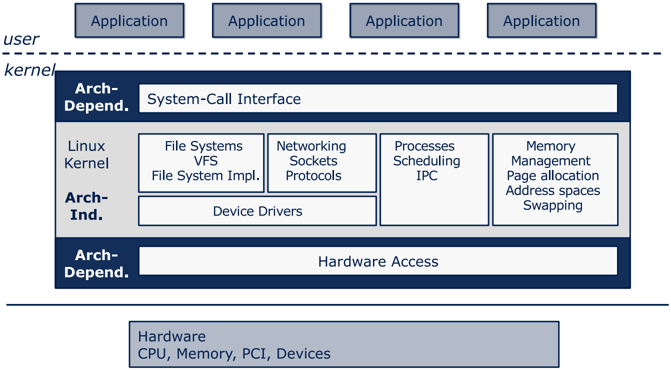
- L4Linux predates the x86 virtualization hype
- L4Linux 2.2 supported MIPS and x86
- L4Linux 2.4 first version to run on L4Env
- L4Linux 2.6 uses Fiasco.OC's paravirtualization features

The current status:

- based on Linux 4.13
- x86, x86-64 and ARM support
- SMP

Example: L4Linux

Native Linux



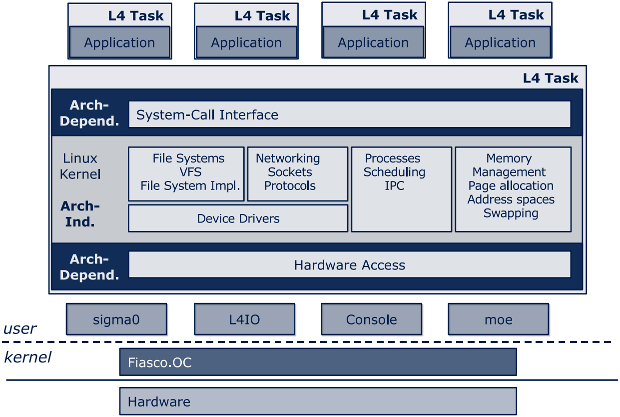
Example: L4Linux

Porting Linux to L4

Regard L4 as new hardware platform. Port small architecture dependent part:

- system call interface
  - kernel entry
  - signal delivery
  - copy from/to user space
- hardware access
  - CPU state and features
  - MMU
  - interrupts
  - memory-mapped and port I/O

Example: L4Linux

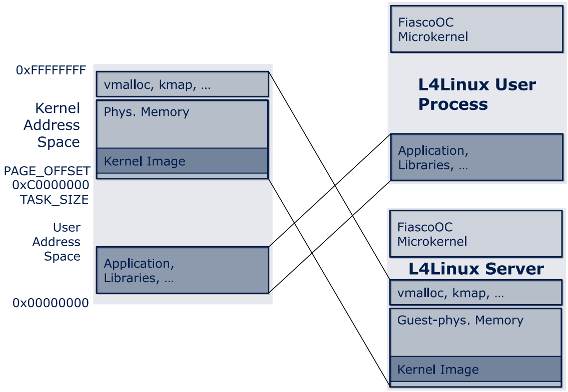


Example: L4Linux

L4Linux Architecture

- L4 specific code is divided into:
  - x86 and ARM specific code
  - hardware generic code
- Linux kernel and Linux user processes run each within a single L4 task.
  - L4Linux kernel task does not see a L4Linux process' virtual memory

Example: L4Linux



Example: L4Linux

L4Linux Challenges

The L4Linux kernel “server” has to:

- access user process data,
- manage page tables of its processes,
- handle exceptions from processes, and
- schedule them.

L4Linux user processes have to:

- “enter” the L4Linux kernel (living in a different address space).

Example: L4Linux

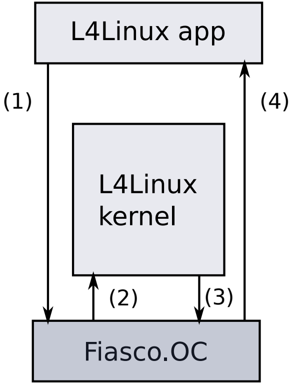
Kernel Entry

Normal Linux syscall interface (int 80h) causes trap.

- L4Linux server receives exception IPC.

Heavyweight compared to native Linux system calls:

- two address space switches,
- two Fiasco kernel entries/exits



Example: L4Linux

Threads & Interrupts

The old L4Linux has a thread for each user thread and virtual interrupt.

- Interrupts are received as messages.
- Interrupt threads have higher priority than normal Linux threads (Linux semantics).
- Interrupt threads force running user process (or idle thread) into L4Linux server.
- Linux uses CLI/STI to disable interrupts, L4Linux uses a lock.

A synchronization nightmare.

Example: L4Linux

L4Linux on vCPUs

Simplify interrupt/exception handling by introducing vCPUs (Fiasco.OC):

- have dedicated interrupt entry points,
  - need to differentiate between interrupt and systemcall
- can be rebound to different tasks,
  - simulates address space switches
- can mask interrupts
  - emulates Interrupt Enable flag
  - don't need that lock anymore

Example: L4Linux

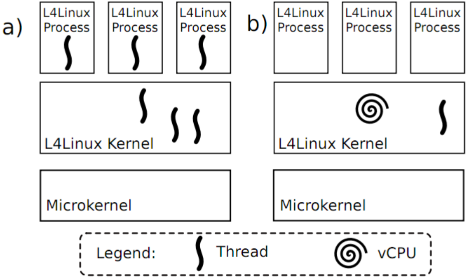


FIGURE 3: *(a) L4Linux implemented with threads and (b) L4Linux implemented with vCPUs.*

Lackorzynski˙virtualprocessors

Example: L4Linux

L4Linux as Toolbox

Reuse large parts of code from Linux:

- filesystems,
- network stack,
- device drivers,
- ...

Use hybrid applications to provide this service to native L4 applications.

Will be topic of upcoming lecture.

Example: L4Linux

Parts of L4Linux Not Covered in Detail

- Linux kernel access to user process' memory
- device drivers
- hybrid applications
- ...

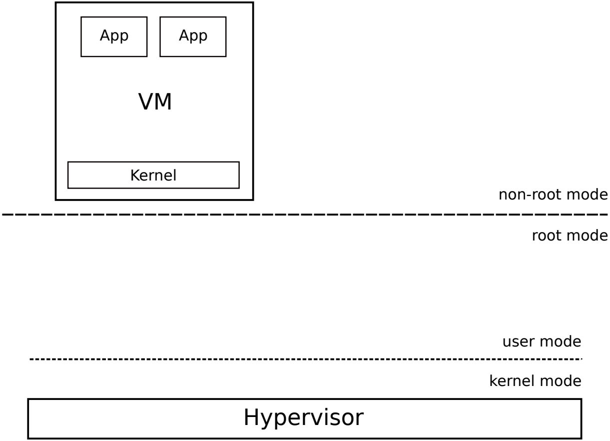
Example: NOVA
Outline

Example: NOVA

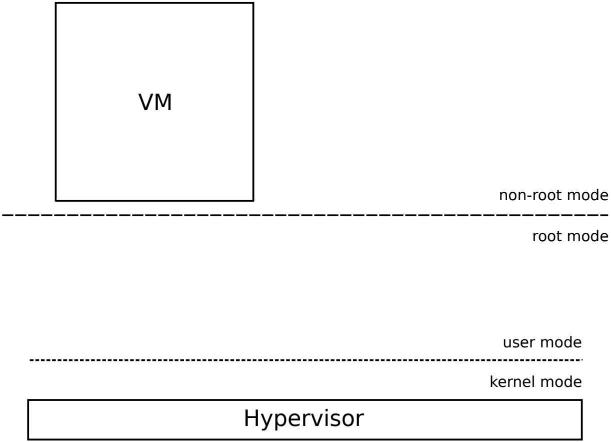
Starting Point

The NOVA OS Virtualization Architecture is a operating system developed from scratch to support virtualization.

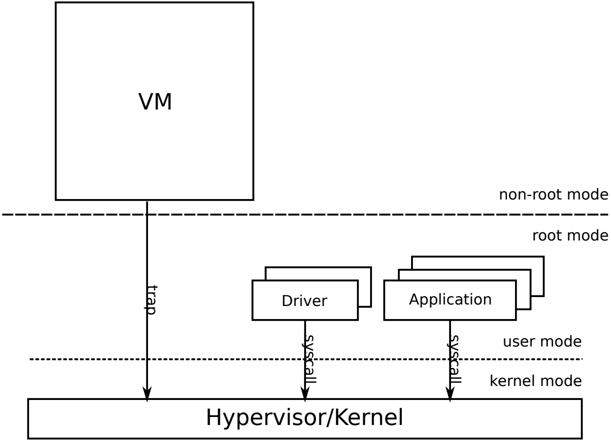
Example: NOVA



Example: NOVA



Example: NOVA



Example: NOVA

Secunia Advisory SA25073

Source: 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."
- ...

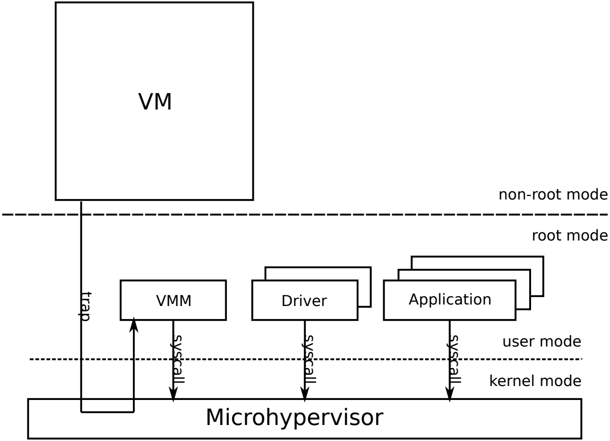
Example: NOVA

TCB of Virtual Machines

The Trusted Computing Base of a Virtual Machine is the amount of hardware and software you have to trust to guarantee this VM's security. (More in lecture on Security)
For e.g. KVM this (conservatively) includes:

- the Linux kernel,
- Qemu.

Example: NOVA



Example: NOVA

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.

Example: NOVA

Microhypervisor vs. VMM

We make a distinction between both terms Steinberg10; 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

Example: NOVA
NOVA Architecture

Reduce complexity of hypervisor:

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

Example: NOVA

VMM: Needed Device Models

For a reasonably useful VMM, you need

- Instruction Emulator
- 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 ...

Example: NOVA
VMM: Virtual BIOS

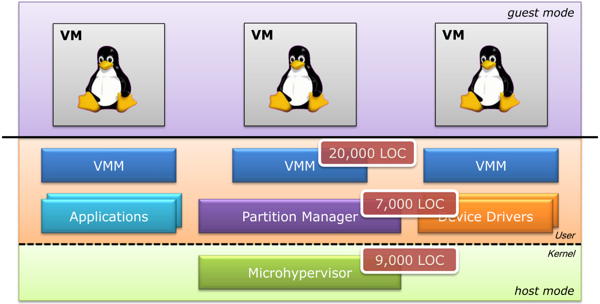
VMM needs to emulate (parts of) BIOS:

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

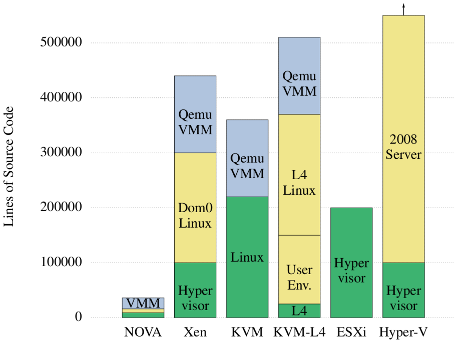
Mostly used for bootloaders and early platform discovery (memory layout).

Example: NOVA

NOVA OS Virtualization Architecture



Example: NOVA
TCB compared



Steinberg:2010:NMS:1755913.1755935

Example: Karma VMM

Outline

Example: Karma VMM

Example: Karma VMM

Idea: Reduce TCB of VMM by using paravirtualization and hardware-assisted virtualization.

- Implemented on Fiasco using AMD-V
- Small VMM: 3800 LOC
- 300 LOC changed in Linux
- No instruction emulator required
  - no MMIO
  - no 16-bit code
- Only simple paravirtualized device models required: 2600 LOC
  - salvaged from L4Linux

Example: Karma VMM

Recap: Examples

- L4Linux is the paravirtualized workhorse on L4/Fiasco.OC:
  - reuse Linux applications
  - reuse Linux components
- NOVA provides faithful virtualization with small TCB for VMs:
  - one VMM per VM
  - run unmodified commodity operating systems
- Karma uses hardware virtualization extensions to simplify paravirtualization

References