Vancouver on NOVA

Virtualization with a small TCB

Bernhard Kauer
bk@vmmon.org

OS Group
Technische Universität Dresden

Salt Lake City
October 9, 2009

The big picture

Our vision: general-purpose OS

- trustworthy and secure
- small and fast

Our problem: Legacy

- support many old applications
- need drivers for any device

Solution

By virtualizing a commodity OS over a low-level kernel, we gain support for legacy applications, and devices we don't want to write drivers for.

Roscoe et al., *Hype and Virtue*, HOTOS XI 2007

Virtualization

Technique

- run an OS and its Apps on virtual hardware

faithfull virtualization: OS unmodified

- virtual hardware behaves like real one
- simple with hardware support (VMX, SVM)
- Examples: KVM, HyperV, Nova, ...

paravirtualization: OS modified

- profit from higher level interface (paravirtops, virtio)
- Examples: L4Linux, Xen dom0, ...

Whats new?

Our Applications

Secure	**Insecure**
BLAC	Firefox
Nitpicker	Xserver + Apps
GSM stack	Java + Apps

$\Rightarrow$ virtual applicance runs side-by-side insecure OS

A small Trusted Computing Base!

- TCB: all code App security depends on
- layer of indirection: virtual environment

TCB Size

TCB of a secure application in KSLOC

- BLAC: 118
- Nitpicker: 10

Xen - 440

Xen	100
dom0	200
Qemu	140

KVM - 360

KVM	20
Linux	200
Qemu	140

Nova - 25

HV	8
Env	5
VMM	12

Outline

Hypervisor and VMM are *not* synonyms!

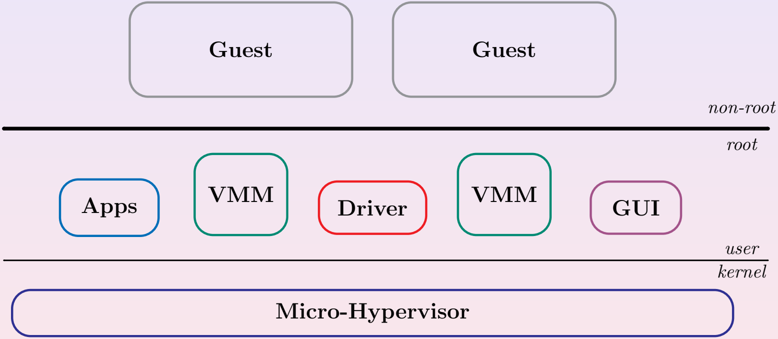
HV: Hypervisor

- low-level **kernel** with HW virtualization support
- runs in most privileged CPU mode

VMM: Virtual Machine Monitor

- external program, sometimes linked to the HV
- provides hardware interface to a **guest** OS
- **virtualizes** device, complex CPU features

NOVA Architecture



NOVA Hypervisor

Implementation

- hypervisor from scratch by Udo Steinberg
- micro-kernel approach: only security and performance features
- supports Apps and VMs on x86, VMX, SVM

Kernel Interface

- ECs as threads and VCPUs
- PDs as adress-spaces and VMs
- capabilities to name kernel objects
- L4-like map+unmap
- portals as server entry points
- fast synchronous IPC
- semaphores and exceptions

Exception Handling

Rational

- How to handle exceptions and VCPU faults?
- local handler or notify pager or VMM
- signal handling: force a thread into an exception

Exception IPC

- kernel generated message
- select EC state to transfer in destination portal
- reply writes state back

Outline

One or many Guests?

Many Guests

- looks like a kernel extension
- What if it crashes?

One VMM per VM

security	VMM compromise/crash affects only its Guest
flexibility	start many VMM instances
performance	avoid layer of indirection

Whats needed?

device models

- timer: PIT, RTC, HPET, PMTimer, ...
- interrupt controller: PIC, LAPIC, IOAPIC
- PCI: hostbridge, disk, network, ...

instruction emulator

real-mode on Intel CPUs

sensitive `cpuid`, `hlt`, `wrmsr`, task-switch

MMIO access to device registers

virtual BIOS

- console input, output, disk access
- resource discovery: memory layout, ACPI tables, ...

Reusing existing code?

Inherit the Bugs

```
CPU  qemu          eventinj fails with #PF
CPU  qemu          invlpga does not generate a #GP on user access
CPU  qemu          not all SVM intercepts does have the right priority, e.g. the rdtsc intercept
CPU  qemu          LTR allows to load 0 selector
CPU  qemu kvm      non CR0_WP bit works only on USER-readonly pages!
CPU  xen           locked operation in emulator do not terminate
!CPU kvm xen       decode only 8 prefixes instead of 14
!CPU qemu          ioio intercept fails on unaligned access
!CPU qemu          ioio intercept just checks single bit
!CPU qemu          rdpmc intercept missing
!CPU qemu          cpuid 0x8000000A unimplemented
!CPU qemu          vmptrld does not generate a #GP
!CPU qemu kvm xen  more than 15 byte opcode length - no #GP
PIC  bochs         does not check for io_len
PIC  fiasco        single EOI after PIC reset
PIC  qemu          poll mode does not set the ISR
PIC  qemu          special mask mode missing
PIT  qemu          periodic modes starts with new counter immediately
PPI  qemu xen      dummy refresh clock instead of PIT_C1 output
RTC  all           irq flags not set when irq disabled
RTC  fiasco        out80 in ack
RTC  qemu          12hour format not in range 1..12
RTC  qemu          alarm test broken
RTC  qemu bochs    regs visible during update-cycle
RTC  qemu bochs    seconds bit7 not readonly
KEYB bochs         panic when OUTB and cmd-response needed
KEYB bochs         panic when controller buffer overflows or written in the wrong order
KEYB bochs         panic on resend
KEYB bochs         immediately reset
KEYB fiasco        legacy polling the keyboard in the timer-irq
KEYB qemu          SCS1+SCS3 does not work
KEYB qemu          output port IRQ1+IRQ12 are set both!
KEYB qemu          overflow could result in corrupted data
KEYB qemu          read-mode response is put into aux buffer
KEYB qemu          cmd response buffer missing
KEYB qemu          does not indicate a scancode overflow
KEYB qemu          immediately reset
KEYB qemu          keyboard buffer too large
MISC bootstrp      out80 in reboot and cons_init
MOUS qemu          does not apply scaling
UART bochs         iir id bits wrong
UART fiasco        clear fifos with 3 outs
UART fiasco        why disabling IRQs on transmit?
UART qemu          does not implement irq disable on OUT2
```

Reusing existing code?

Inherit the Bugs

```
CPU   qemu          eventinj fails with #PF
CPU   qemu          invlpga does not generate a #GP on user access
CPU   qemu          not all SVM intercepts does have the right priority, e.g. the rdtsc intercept
CPU   qemu          LTR allows to load 0 selector
CPU   qemu kvm      non CR0_WP bit works only on USER-readonly pages!
CPU   xen           locked operation in emulator do not terminate
!CPU  kvm xen       decode only 8 prefixes instead of 14
!CPU  qemu          ioio intercept fails on unaligned access
!CPU  qemu          ioio intercept just checks single bit
!CPU  qemu          rdpmc intercept missing
!CPU  qemu          cpuid 0x8000000A unimplemented
!CPU  qemu          vmptrld does not generate a #GP
!CPU  qemu kvm xen  more than 15 byte opcode length - no #GP
PIC   bochs         does not check for io len
PIC   fiasco
PIC   qemu
PIC   qemu
PIT   qemu
PPI   qemu xen
RTC   all
RTC   fiasco
RTC   qemu
RTC   qemu
RTC   qemu bochs
RTC   qemu bochs
KEYB  bochs
KEYB  bochs         pa
KEYB  bochs         panic on resend
KEYB  bochs         immediately reset
KEYB  fiasco        legacy polling the keyboard in the timer-irq
KEYB  qemu          SCS1+SCS3 does not work
KEYB  qemu          output port IRQ1+IRQ12 are set both!
KEYB  qemu          overflow could result in corrupted data
KEYB  qemu          read-mode response is put into aux buffer
KEYB  qemu          cmd response buffer missing
KEYB  qemu          does not indicate a scancode overflow
KEYB  qemu          immediately reset
KEYB  qemu          keyboard buffer too large
MISC  bootstrp      out80 in reboot and cons_init
MOUS  qemu          does not apply scaling
UART  bochs         iir id bits wrong
UART  fiasco        clear fifos with 3 outs
UART  fiasco        why disabling IRQs on transmit?
UART  qemu          does not implement irq disable on OUT2
```

VMM from scratch

- could reuse Bochs or Qemu
- many bugs and missing features
- understand the devices to fix them

Reusing existing code?

Smaller when rewritten

Experiment: network model

- wrapped ne2k and e1000 model from Qemu
- as much code as all our current models together!
- fixed backend interface: requires double copy

VMM Design
Motivation

VMMs are complex: Qemu

- $>$ 400 KSLOC
- architecture dependent device models
- strange dependencies (RTC → HPET, PIC → IOAPIC → LAPIC)

⇒ apply Multi-Server approach to VMM

User-level is complex: Multi-Server

- many independent servers — many device models
- small reusable interfaces — PIO, MMIO, PCI interfaces

VMM Design
Idea

Interface through a bus

- similar to FSB, PCI, ISA on the motherboard
- here: send and listen for messages on virtual busses

Example: device model

receives
- in+out, memory read+write, PCI config-space

sends
- raise interrupt line and backend request

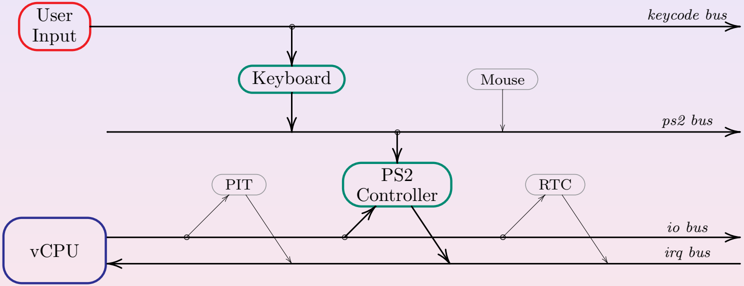
backend and driver communication

mouse movements and clicks

disk read+write block to disk

VMM Design

BUS Architecture



VMM Design
Properties

Results

- independent reusable components

vBIOS	keyboard driver
VESA	instruction emulator

- NOVA specific VMM code: currently 700 SLOC
- user-level environment is also componentized

User-level Environment

In which environment runs the VMM?

- Porting

  Posix Linux, Minix, ...
  multi-server L4re, Iguana, Genode, Hurd, ...

- hypervisor: different abstractions
- existing user-level software bigger than needed

Implemented

drivers AHCI, keyboard, serial, RTC, ...
services startup, console, timer, disk, network hub, bootp

$\Rightarrow$ minimal environment - currently 5000 SLOC

A small TCB - How to achieve it?

1. carefull select devices to virtualize
   - good understanding of a device simplifies model
2. avoid unnecessary features
   - architecture independent instruction-emulator
   - binary translation
3. reuse components
4. apply new development techniques
   - generate instruction-decode switch for emulator
5. minimal-complexity oriented design
   - virtual BIOS code implemented in the VMM

Testing

Code Quality matters!

Example: networking bug

- symptoms: network stopped under load
- long search: two bugs
  1. race condition in Linux driver
  2. missing recovery in ne2k model

Approach

- implement only essential device models
- as accurate as possible
- finish one, then start next

Testing

Different Strategies

running some OS

- bad idea for correctness - just popular code paths
- reproducing and debugging is hard
- helpful for performance bugs

manual audit against specification

- does not scale well
- examples: PIC, serial, ...

exercise all cases

- examples: PIT, RTC, ...

Preliminary Performance Figures

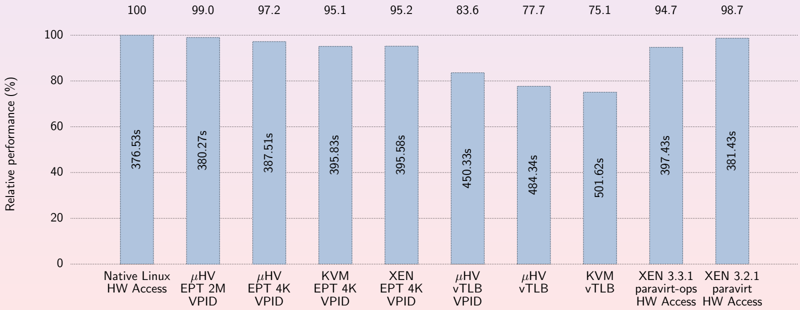
Memory-bound benchmark

Preliminary Performance Figures

Memory-bound benchmark

parallel kernel compile

- more than 97% native performance with EPT
- on par with KVM, Xen
- even better: gain 1-2% from large pages

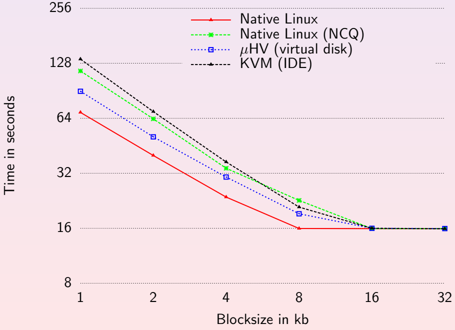


Preliminary Performance Figures

IO-bound benchmark

Disk

- disk stressing with different block sizes
- faster: we emulate SATA instead of IDE
- todo: comparison with paravirt, ESX, multiple VMs...



Summary

OS virtualization

- can be done efficiently with a small TCB
- on NOVA is ready for research

Future Work

- SMP and scalability
- real-time and VMs
- para-virtualization obsolete?
- NOVA as our development platform