



KVM: A Hypervisor for All Seasons

Avi Kivity
avi@qumranet.com

November 2007

Virtualization

- ◆ Simulation of computer system in software
- ◆ Components
 - ◆ Processor: register state, instructions, exceptions
 - ◆ Memory Management: paging, protection, tlb
 - ◆ I/O: Networking, storage, human interface
- ◆ Performance and fidelity are critical

Uses

- ◆ Server consolidation
- ◆ Testing, R&D
- ◆ Virtual Desktops

Virtualization basics

- ◆ Trap changes to privileged state
 - ◆ Guest cannot access hardware
- ◆ Hide privileged state
 - ◆ Guest cannot detect that the host is changing things behind its back
- ◆ Example: interrupt enable flag

A Look Back

- ◆ VMWare
 - ◆ Just-in-time compilation of binaries
- ◆ Xen
 - ◆ Paravirtualized guests

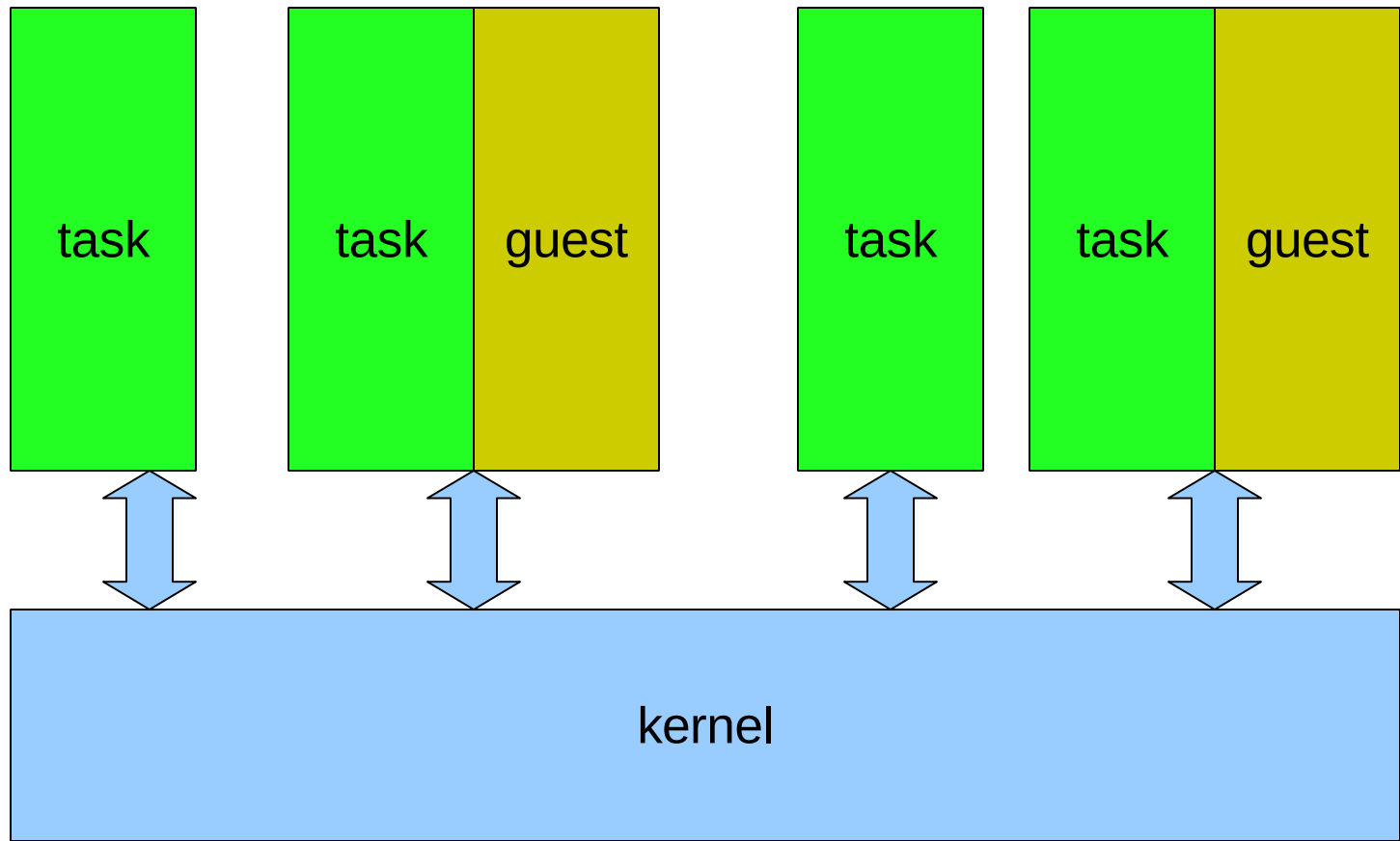
Virtualization

- ◆ Simulation of computer system in software
- ◆ Components
 - ◆ Processor Management
 - ◆ Memory Management
 - ◆ IO Management
- ◆ Difference from *emulation* is emphasis on near-native performance

The KVM approach

- ◆ Reuse Linux code as much as possible
- ◆ Focus on virtualization, leave other things to respective developers
- ◆ Integrate well into existing infrastructure, codebase, and mindset

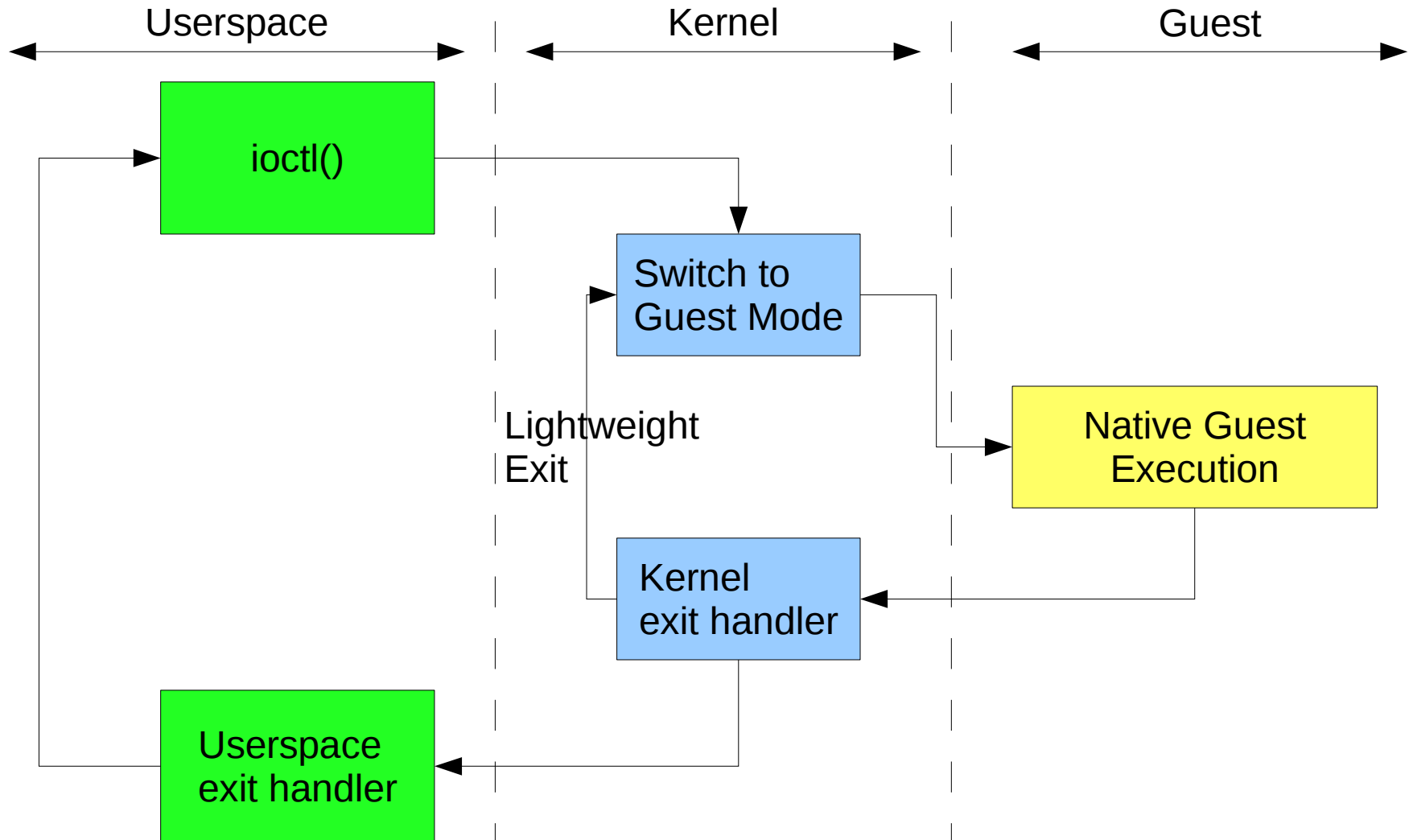
kvm process model



KVM process model (cont'd)

- ◆ Guests are scheduled as regular processes
- ◆ Guests are processes that can execute in three modes: user mode, kernel mode, and *guest* mode
- ◆ `kill(1)`, `top(1)` work as expected
- ◆ Guest physical memory is mapped into the task's virtual memory space
- ◆ Virtual processors in a VM are threads in the host process

KVM Execution Model



What's handled in the kernel?

- ◆ CPU virtualization (special instructions)
- ◆ MMU virtualization
- ◆ Local APIC, PIC, and IOAPIC
- ◆ (planned) paravirtualized network and block device
- ◆ (planned) paravirtualized guest kernel support code

KVM Initial Design Parameters

- ◆ x86 only
- ◆ Focus on full virtualization
- ◆ Zero modifications to host kernels

We were wrong!

- ◆ Support for new architectures is underway
 - ◆ s390: the big daddy of virtualization
 - ◆ PowerPC: for embedded/consumer electronics
 - ◆ ia64: large machines
- ◆ Paravirtualization support
 - ◆ Improves performance, timing accuracy
- ◆ Modifications to host to improve scheduling, add swapping

Oh yes, swapping

- ◆ KVM can swap guest memory
- ◆ Needed for untrusted guests
- ◆ Enables mmu games like page migration – needed for good NUMA support

KVM is a full-spectrum hypervisor

- ◆ Servers
- ◆ Desktops/laptops
- ◆ Small/embedded boards
- ◆ Really large machines

KVM on servers

- ◆ Same management tools and infrastructure as Linux
- ◆ Supports virtual machines and regular processes side-by-side
 - ◆ Useful for real-time server processing workloads
- ◆ Uses the Linux scheduler and I/O stack
- ◆ Live migration
- ◆ Integrates with cluster filesystems

KVM on desktops/laptops

- ◆ Will boot on anything Linux boots
- ◆ Normal desktop doesn't change
- ◆ Excellent power management
- ◆ Suspend/resume while virtual machines are running
- ◆ Swap idle VMs when not in use
- ◆ Passthrough devices
 - ◆ USB
 - ◆ PCI (planned)

KVM for embedded

- ◆ Board support is in Linux already
- ◆ Can make footprint as small as needed
- ◆ Real-time scheduling support

KVM and big iron

- ◆ Leverage many years of Linux scalability work
 - ◆ NUMA
 - ◆ Page migration
- ◆ Linux already supports 4096-processor machines
 - ◆ Competitors can only dream of this

Paravirtualized device drivers

- ◆ virtio
 - ◆ Common drivers for all hypervisors
 - ◆ Hypervisor-specific backend
 - ◆ KVM backend is in progress

Release philosophy

- ◆ Development snapshots every 1-2 weeks
 - ◆ Release early and often
 - ◆ Features introduced quickly
 - ◆ Bugs fixed quickly
 - ◆ Bugs added quickly...
 - ◆ Allows developers and users to track and test the latest and greatest
- ◆ Stable releases part of Linux 2.6.x
 - ◆ With bugfixes going into Linux 2.6.x.y

Differences between Xen and KVM

- ◆ No dom0
 - ◆ Removes scaling bottleneck
 - ◆ Removes priority inversion problem
- ◆ Interrupts handled directly by host
 - ◆ Xen requires the scheduler to dispatch interrupts
- ◆ Uses the Linux bootstrap
 - ◆ Bringing up KVM on a new machine is trivial
- ◆ Uses well-known Linux environment
 - ◆ Rich kernel APIs
 - ◆ Large developer base

Status & plans

◆ Status

- ◆ Many supported guests
- ◆ Good cpu/mmu performance
- ◆ Good host scaling

◆ Plans

- ◆ Add more paravirtualization (both cpu and I/O)
- ◆ Improve guest scaling
- ◆ Add more hardware support (architectures, hardware features)
- ◆ Improve swapping

KVM pros

- ◆ Leverages Linux scheduler, memory management, I/O
- ◆ No scheduler involvement for I/O
- ◆ Uses existing Linux security model (can run VM as ordinary user)
- ◆ Uses existing management tools
- ◆ Power management
- ◆ Guest memory swapping
- ◆ Real-time scheduling
- ◆ Leverages Linux development momentum



Thank You
