



# KVM: Virtualisation The Linux Way

---

**Amit Shah**

**[amit.shah@qumranet.com](mailto:amit.shah@qumranet.com)**

**GEEP**



# Virtualisation Strategies

- ◆ “Native” Hypervisors
  - ◆ Have a runtime
  - ◆ Need a “primary” guest OS
  - ◆ Examples: Xen, VMWare ESX Server, IBM mainframes
- ◆ Containers
  - ◆ Different namespaces for different guests
  - ◆ Run on host kernel
  - ◆ Userland can be different from host
  - ◆ Examples: OpenVZ, FreeVPS, Linux-Vserver
- ◆ Paravirtualisation
- ◆ Emulation
  - ◆ Examples: QEMU, PearPC

# KVM: Architectures Supported

- ◆ S390
  - ◆ IBM mainframes: a hypervisor is a must
  - ◆ Included in 2.6.26
- ◆ IA-64
  - ◆ Included in 2.6.26
- ◆ X86
  - ◆ Included in 2.6.20
  - ◆ KVM-lite: PV Linux guest on non-VT<sub>x</sub> / non-SVM host (proposed)
- ◆ PowerPC
  - ◆ PV
  - ◆ Architecture support for hypervisor
  - ◆ Included in 2.6.26

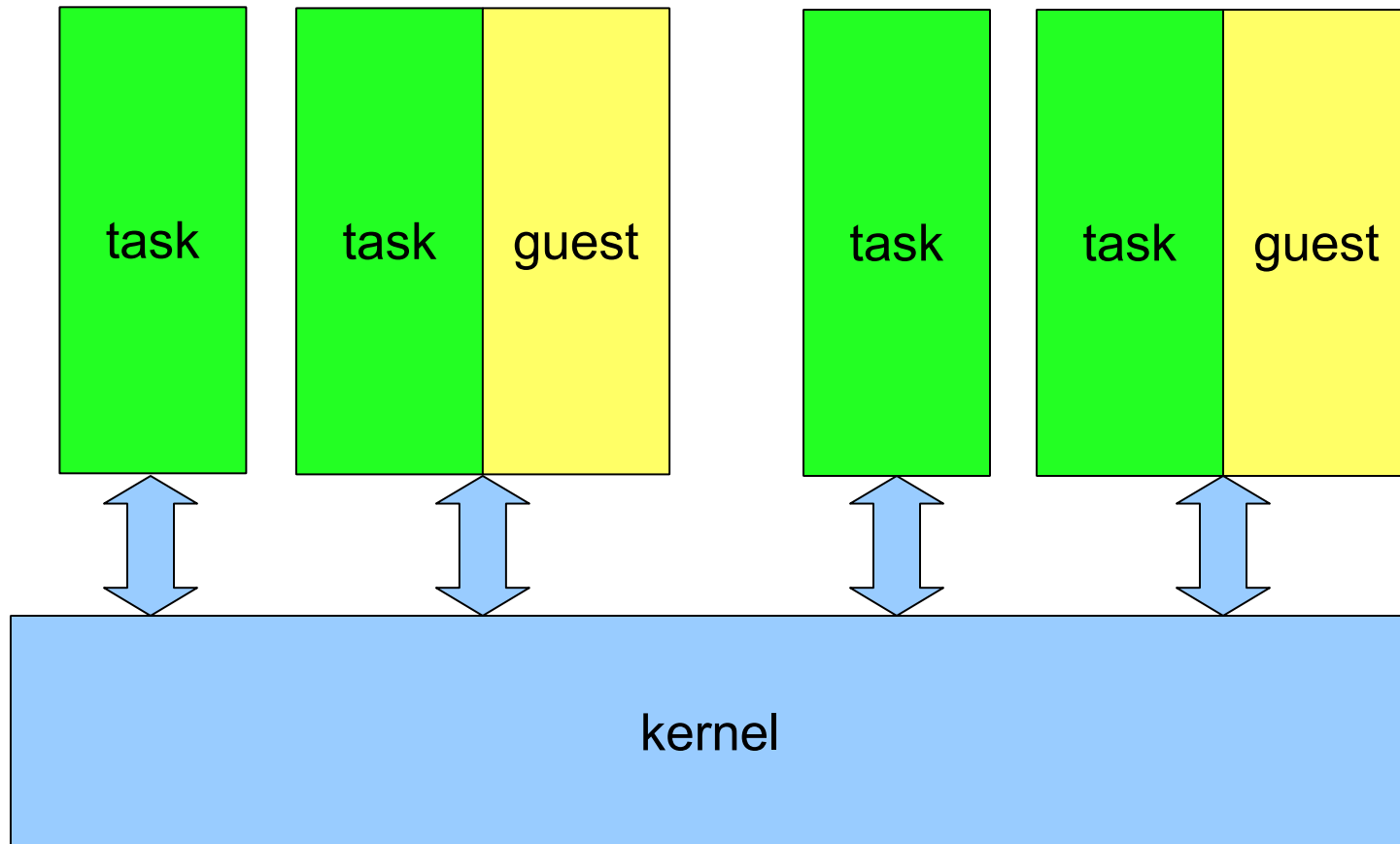
# X86 Hardware Extensions

- ◆ 'guest mode' in addition to user and kernel modes
- ◆ Raise a trap for all privileged instructions
- ◆ Virtualised registers
- ◆ Processor
  - ◆ Intel-VT<sub>x</sub> (VMX)
  - ◆ AMD-V (SVM)
- ◆ MM
  - ◆ EPT (Intel)
  - ◆ NPT (AMD)
- ◆ IO
  - ◆ VT-d (Intel)
  - ◆ IOMMU (AMD)

# What's handled in the kernel?

- ◆ CPU virtualisation (special instructions)
- ◆ MMU virtualisation
- ◆ Local APIC, PIC, IOAPIC, PIT
- ◆ (guest) paravirtualised network and block device drivers
  - ◆ virtio-net
  - ◆ virtio-block
- ◆ (guest) paravirtualised kernel support code
  - ◆ paravirt\_ops
  - ◆ MMU
- ◆ (guest) paravirtualised clock driver

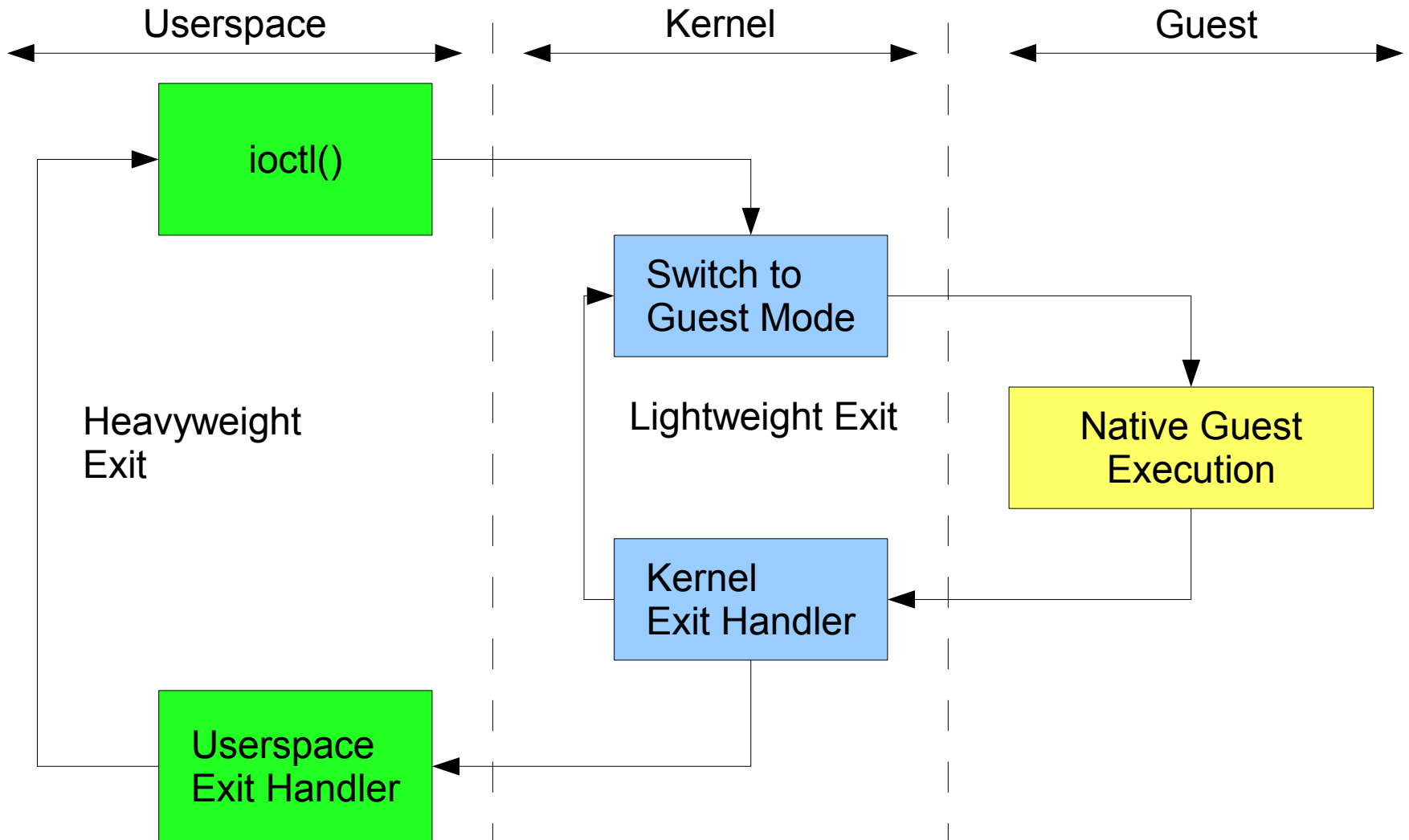
# KVM Process Model



# KVM Process Model (cont'd)

- ◆ Guests are scheduled as regular processes
- ◆ `kill(1)`, `top(1)` work as expected
- ◆ Guest physical memory is mapped into the task's virtual memory space
- ◆ Virtual processors in one VM are threads

# KVM Execution Model



# Flow Example: Memory Access

- ◆ Guest accesses an unmapped memory location
- ◆ Hardware traps into kernel mode
- ◆ kvm walks the guest page table, determines guest physical address
- ◆ kvm performs guest physical -> host physical translation
- ◆ kvm installs shadow page table entry containing guest virtual -> host physical translation
- ◆ Processor restarts execution of faulting instruction

# Paravirtualisation

- ◆ Modifying guest OS for performance
- ◆ Virtio
  - ◆ Common drivers for all hypervisors
  - ◆ Hypervisor-specific backend
  - ◆ KVM backend in qemu
  - ◆ Faster performance
  - ◆ Efficient block, net drivers
  - ◆ Balloon
  - ◆ Iguest, KVM use it already
- ◆ PV DMA
  - ◆ Pass through Ethernet devices
- ◆ paravirt\_ops

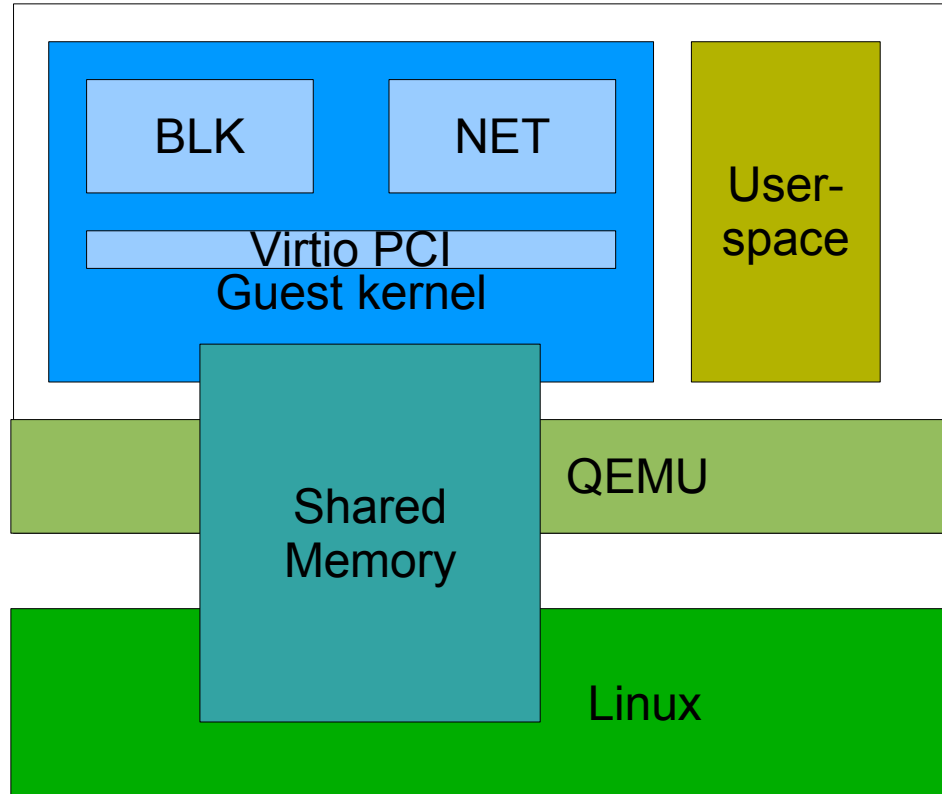
# Network Devices

- ◆ Fully virtualised device performance not great
  - ◆ 55 Mbps for RTL
  - ◆ Lots of IO-exits per packet
- ◆ Decided to implement a modern e1000
  - ◆ Advantages:
    - ◆ All code in userspace (qemu)
    - ◆ All existing drivers recognise device
  - ◆ IRQ coalescing
  - ◆ Only 2-3 IO-exits per packet
  - ◆ Goes in excess of 800 Mbps

# Virtio Net

- ◆ Shared memory between host and guest
- ◆ Two queues: recv and send
- ◆ Ring buffer within each queue
- ◆ 'available' pointer controlled by guest
- ◆ 'used' pointer controlled by host

# Virtio-net on KVM



# Ideas

- ◆ Shared memory between host and guest via virtio-pci
- ◆ Shared directory between host and guest using virtio + fuse
- ◆ VMGL (OpenGL for Virtual Machines) support
- ◆ <http://kvm.qumranet.com/kvmwiki/TODO>

# KVM Pros

- ◆ Leverages Linux scheduler, memory management, I/O
- ◆ No scheduler involvement for I/O
- ◆ Full virtualisation: No changes to the guest necessary
  - ◆ Paravirt drivers available for better performance
- ◆ Uses existing Linux security model
  - ◆ can run VM as ordinary user
- ◆ Uses existing management tools
- ◆ Power management
- ◆ Guest memory swapping
- ◆ Real-time scheduling, NUMA
- ◆ Leverages Linux development momentum: all new drivers, {cpu, disk} schedulers, file systems, etc

# Distro / Industry interest

- ◆ libvirt
  - ◆ Managing various guests under a hypervisor
  - ◆ Support for Xen, KVM
  - ◆ APIs between UI, middle layer and virtualisation backend
- ◆ Distributions
  - ◆ Debian
  - ◆ Ubuntu
  - ◆ RedHat EL
  - ◆ SLES
- ◆ Qumranet
  - ◆ Dekstop Virtualisation

# 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

# Journey

- ◆ Linux 2.6.20 (4 Feb 2007): Initial release
- ◆ Linux 2.6.21 (25 Apr 2007): Stability, suspend/resume
- ◆ Linux 2.6.22 (8 Jul 2007): Stable ABI
  - ◆ Old userspace, new kernel
  - ◆ New userspace, old kernel
- ◆ Linux 2.6.23 (9 Oct 2007): SMP, performance
- ◆ Linux 2.6.24 (24 Jan 2008): In-kernel APIC, preemptibility, virtio
- ◆ Linux 2.6.25 (16 Apr 2008): Guest swapping, paravirt\_ops, balloon drv
- ◆ Linux 2.6.26 (soon): PowerPC, s390, IA64, NPT, EPT, more paravirt (mmu), ...

# KVM is Developer-friendly

- ◆ No need to reboot (usually)
- ◆ Netconsole, oprofile, all the tools work
- ◆ Small codebase
- ◆ Friendly community

# Future

- ◆ Consolidate various virtualisation solutions existing in the kernel
  - ◆ Started with move to virt/ from drivers/kvm/
- ◆ More hardware features support
- ◆ More paravirtualisation support
- ◆ Improve guest scaling
- ◆ Better support for management layers like libvirt
- ◆ Intel Real Mode Emulation

# Do Read

- ◆ virt/\*, arch/[x86|ia64|s390|powerpc]/kvm/\*
- ◆ [KvmForum2007](http://kvm.qumranet.com) wiki page on <http://kvm.qumranet.com>
- ◆ [kvm@vger.kernel.org](mailto:kvm@vger.kernel.org)
- ◆ [virtualization@lists.osdl.org](mailto:virtualization@lists.osdl.org)



**Thank You**

---