KVM: Linux-based Virtualization

Chris Wright <chrisw@redhat.com>

Agenda

- Quick view
- Features
- libvirt
- oVirt
- KVM Execution loop
- Memory management
- Linux Integration
- Paravirtualization
- I/O
- Power Management
- Non-x86
- Real time
- Xenner
- Roadmap
- Community
- Conclusions

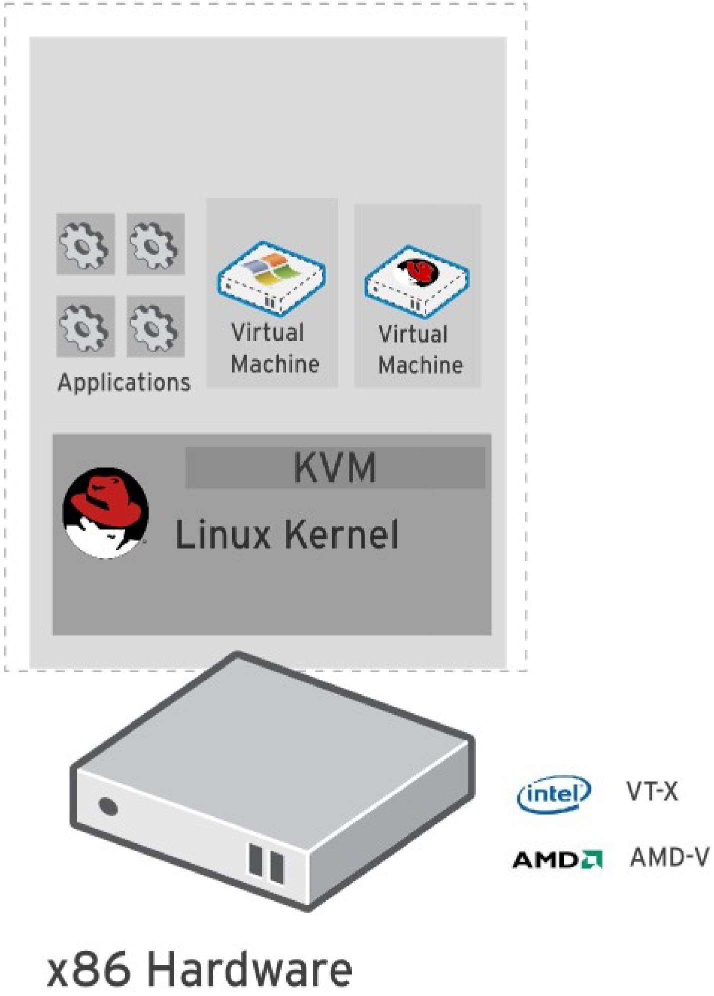
At a glance

- KVM – the Kernel-based Virtual Machine – is a Linux kernel module that turns Linux into a hypervisor
- Requires hardware virtualization extensions
    - paravirtualization where makes sense
- Supports multiple architectures: x86 (32- and 64- bit) s390 (mainframes), PowerPC, ia64 (Itanium)
- Competitive performance and feature set
- Advanced memory management
- Tightly integrated into Linux

The KVM approach

- A hypervisor needs
  - A scheduler and memory management
  - An I/O stack
  - Device drivers
  - A management stack
  - Networking
  - Platform support code
- Linux has world-class support for all this, so why reinvent the wheel?
- Reuse Linux code as much as possible
- Focus on virtualization, leave other things to respective developers
- Benefit from semi-related advances in Linux

Architecture Overview of KVM



- KVM: Kernel-based Virtual Machine – Full virtualization solution for Linux
- Incorporated into the Linux kernel in 2006
- Converts Linux into a hypervisor. Run unmodified OSes as guests.
- KVM architecture provides high “feature-velocity” – leverages the power of Linux

KVM features

- Leverage HW virtualization support
  - VT-x/AMD-V, EPT/NPT, VT-d/IOMMU
- CPU and memory overcommit
- High performance paravirtual i/o
- Hotplug (cpu, block, nic)
- SMP guests
- Live migration
- Power Management
- NUMA
- PCI Device Assignment and SR-IOV
- Page sharing
- SPICE
- KVM autotest

Libvirt Features

- Hypervisor agnositc
  - Xen, KVM, QEMU, LXC, UML, OpenVZ
- Provisioning, lifecycle management
- Storage
  - IDE/SCSI/LVM/FC/Multipath/NPIV/NFS
- Networking
  - Bridging, bonding, vlans, etc
- Secure remote management
  - TLS, Kerberos
- Many common language bindings
  - Python, perl, ruby, ocaml, c#, java
- CIM provider
- AMQP agent

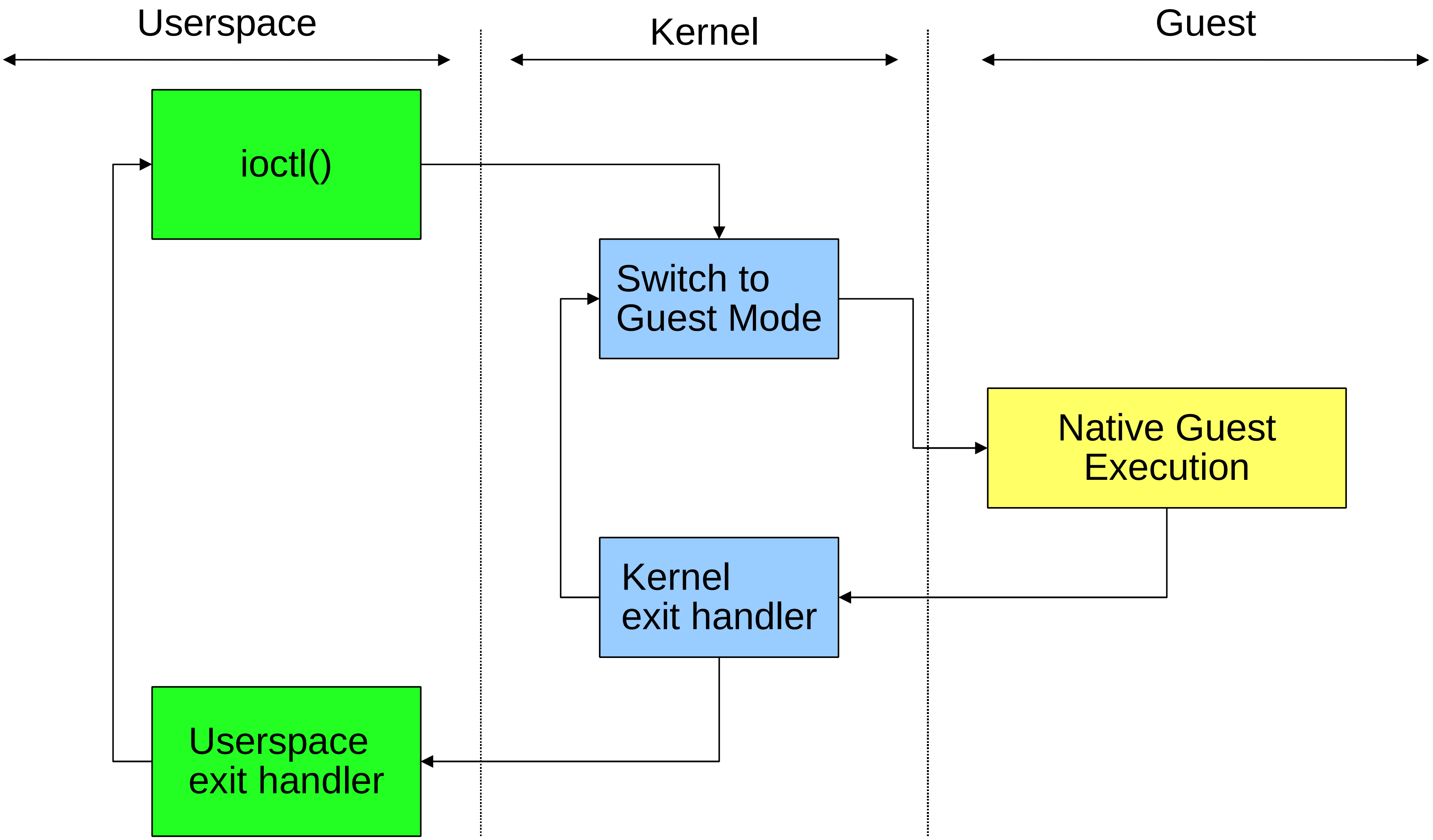
oVirt features

- Scalable data center virtualization management
  - Server and desktop
- Small footprint virtualization hosting platform
- Web UI for centralized remote mgmt
- Directory integration
- Hierarchical resource pools
- Statistics gathering
- Provisioning, SLA, load balancing

KVM Execution Model

- Three modes for thread execution instead of the traditional two:
  - User mode
  - Kernel mode
  - Guest mode
- A virtual CPU is implemented using a Linux thread
- The Linux scheduler is responsible for scheduling a virtual cpu, as it is a normal thread

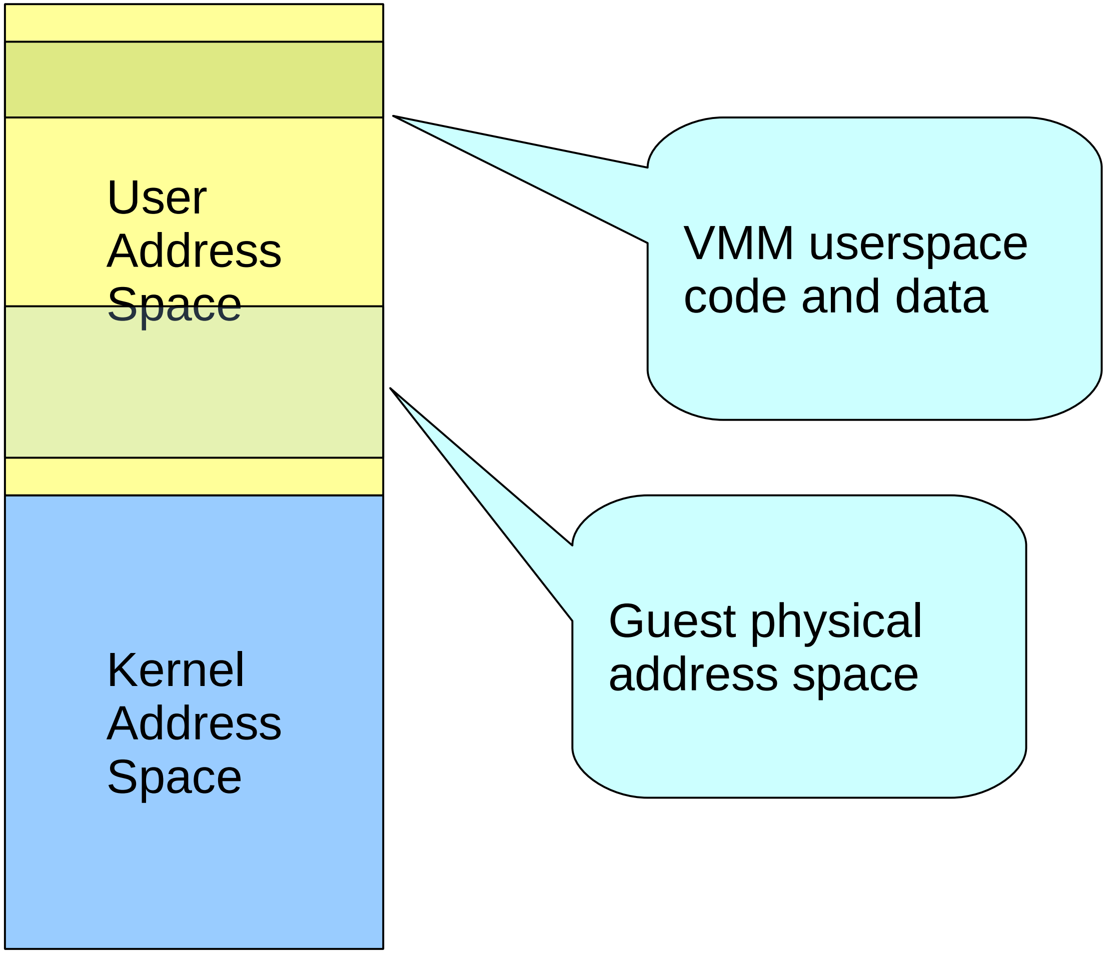
KVM Execution Model



KVM Execution Model

- Guest code executes natively
  - Apart from trap'n'emulate instructions
- Performance critical or security critical operations handled in kernel
  - Mode transitions
  - Shadow MMU
- I/O emulation and management handled in userspace
  - Qemu-derived code base
  - Other users welcome

KVM Memory Model



VMM userspace code and data

Guest physical address space

KVM Memory Model

- Guest physical memory is just a chunk of host virtual memory, so it can be
  - Swapped
  - Shared
  - Backed by large pages
  - Backed by a disk file
  - COW'ed
  - NUMA aware
- The rest of the host virtual memory is free for use by the VMM
  - Low bandwidth device emulation
  - Management code

Linux Integration

- Preemption (and voluntary sleep) hooks: preempt notifiers
- Swapping and other virtual memory management: mmu notifiers

Preempt Notifiers

- Linux may choose to suspend a vcpu's execution
- KVM runs with some guest state loaded while in kernel mode (FPU, etc.)
- Need to restore state when switching back to user mode
- Solution: Linux notifies KVM whenever it preempts a process that has guest state loaded
  - ... and when the process is scheduled back in
- Allows the best of both worlds
  - Low vmexit latency
  - Preemptibility, sleeping when paging in

MMU Notifiers

- Linux doesn't know about the KVM MMU
- So it can't
  - Flush shadow page table entries when it swaps out a page (or migrates it, or ...)
  - Query the pte accessed bit when determines the recency of a page
- Solution: add a notifier
  - for tlb flushes
  - for accessed/dirty bit checks
- With MMU notifiers, the KVM shadow MMU follows changes to the Linux view of the process memory map

Paravirtualization

- Not nearly as critical for CPU/MMU now with hardware assistance
  - Highly intrusive
- KVM has modular paravirtualization support
  - Turn on and off as needed by hardware
- Supported areas
  - Hypercall-based, batched mmu operations
  - Clock
  - I/O path (virtio)

Virtio

- Most devices emulated in userspace
  - With fairly low performance
- Paravirtualized I/O is the traditional way to accelerate I/O
- Virtio is a framework and set of drivers:
  - A hypervisor-independent, domain-independent, bus-independent protocol for transferring buffers
  - A binding layer for attaching virtio to a bus (e.g. pci)
  - Domain specific guest drivers (networking, storage, etc.)
  - Hypervisor specific host support

Power management

- A good example of how Linux integration helps
  - An especially icky area in operating systems
- KVM has
  - Automatic frequency scaling
    - with several governors
  - Suspend/resume support
    - with running virtual machines
- All with a small amount of glue code

Other cpu architectures

- s390 (aka zSeries, aka mainframe)
  - KVM support recently integrated
- ia64 (aka Itanium)
  - ditto
- PowerPC embedded

Real time

- Linux has (unmerged) hard real time support
- KVM does not interfere with the real time properties of real time Linux
- Can run virtual machines alongside hard real time processes
  - Run a GUI in a container alongside an industrial controller
  - Or a cell phone
  - Or, soak up unused cycles on real-time financials servers

Xenner

- An independent application that uses KVM
- Emulates the Xen hypervisor ABI
  - Much, much smaller than Xen
- Used to run unmodified Xen guests on KVM

Roadmap

- QEMU improvements and integration
  - Libmonitor, machine descritption
- qxl/SPICE integration
- Scalability work
  - Qemu and kvm
- Performance work
  - Block
    - i/o using linux aio
  - Network
    - GRO
    - Multiqueue virtio
    - Latency reduction
    - Zero copy
- Enlightenment

Community

- Main contributors
  - AMD, IBM, Intel, Red Hat
- Typical open source project
  - Mailing lists, IRC
- Will love to see you contribute

http://linux-kvm.org

http://libvirt.org

http://ovirt.org

Conclusions

- Simple model - no excess baggage
- Fully featured
- Great performance
- Rapidly moving forward

redhat®