KVM: Linux-based Virtualization

Avi Kivity
avi@qumranet.com

Columbia University Advanced OS/Virtualization course
Agenda

- Quick view
- Features
- KVM Execution loop
- Memory management
- Linux Integration
- Paravirtualization
- I/O
- Power management
- Non-x86
- Real time
- Xenner
- 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
- 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

- Reuse Linux code as much as possible
- Focus on virtualization, leave other things to respective developers
- Integrate well into existing infrastructure, codebase, and mindset
- Benefit from semi-related advances in Linux
KVM

Ordinary Linux Process

User VM

User VM

User VM

Modules

Linux

Driver

Driver

Driver

Hardware
KVM model benefits

- Reuse scheduler, memory management, bringup
- Reuse Linux driver portfolio
- Reuse I/O stack
- Reuse management stack
KVM Process Model

- task
- guest
- task
- guest

kernel
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

Userspace

ioctl()

Kernel exit handler

Switch to Guest Mode

Kernel exit handler

Guest

Native Guest Execution
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

- **Kernel Address Space**
- **User Address Space**
- **Guest physical address space**
- **VMM userspace code and data**
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

- 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
Preempt notifiers

External interrupt or trap

Restore host state

Context switch

VMM process in host kernel

Guest

Restore guest state

Context switch

Scheduler

Other process
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

- Yesterday's hot topic
  - Needed for decent MMU performance without two-dimensional paging
  - Intrusive
- KVM has modular paravirtualization support
  - Turn on and off as needed by hardware
  - Still needs hardware virtualization extensions
- Supported areas
  - Hypercall-based, batched mmu operations
  - Clock
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 System Z, aka mainframe)
  - KVM support recently integrated
- ia64 (aka Itanium)
  - ditto
- PowerPC embedded
  - In development
  - Coming soon to a cell phone near you
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
Community

- Main contributors
  - AMD, IBM, Intel, Qumranet, Red Hat
- Typical open source project
  - Mailing lists, IRC
- Annual Developer's Forum
  - This year at Napa, California in June
- Will love to see you contribute
Conclusions

- Simple model - no excess baggage
- Fully featured
- Good performance
- Catching up from behind - but at a tremendous pace

Participate at
http://kvm.qumranet.com
Thank You