

Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

Systems Software &
Architecture Lab.
Seoul National University

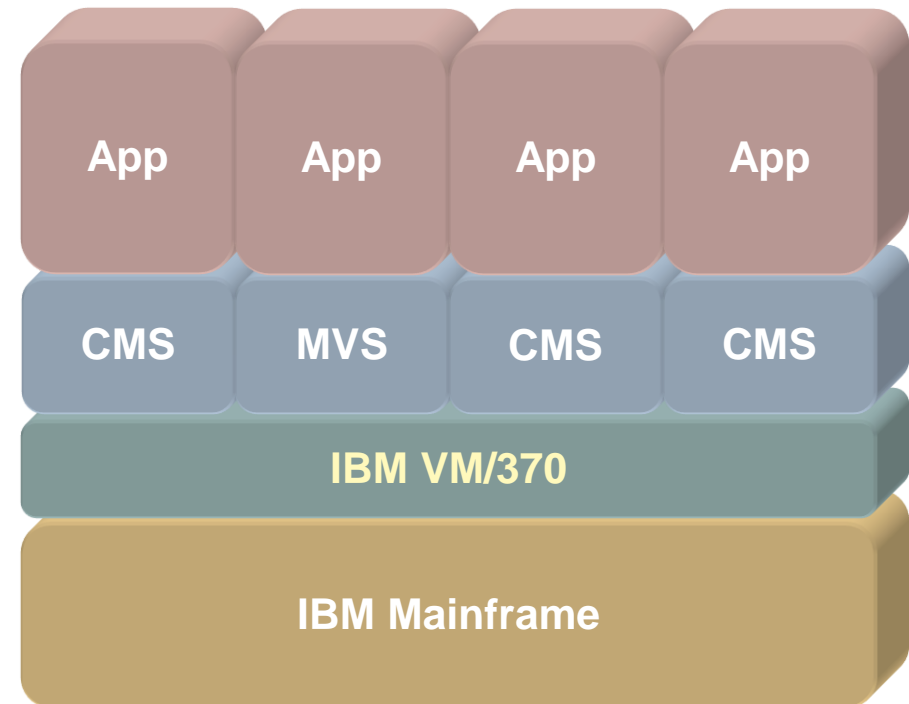
Spring 2023

Virtual Machines



Virtual Machine

- A fully protected and isolated copy of the underlying physical machine's hardware (definition by IBM)
- Virtual machine monitor (VMM)
 - A thin software layer that sits between hardware and the operating system – virtualizing and managing all hardware resources
 - “Hypervisor”



History: Old Idea from 1960s

- **IBM VM/370 – A VMM for IBM mainframe**
 - Multiple OS environments on expensive hardware
 - Desirable when few machines around
- **Popular research idea in 1960s and 1970s**
 - Entire conferences on virtual machine monitors
 - Hardware/VMM/OS designed together
 - Robert Goldberg, Architectural Principles for Virtual Computer Systems, Ph.D. Thesis, Harvard University, 1972.
- **Interest died out in the 1980s and 1990s**
 - Hardware got cheap
 - OS got more powerful (e.g., multi-user)

A Return to Virtual Machines in 90's

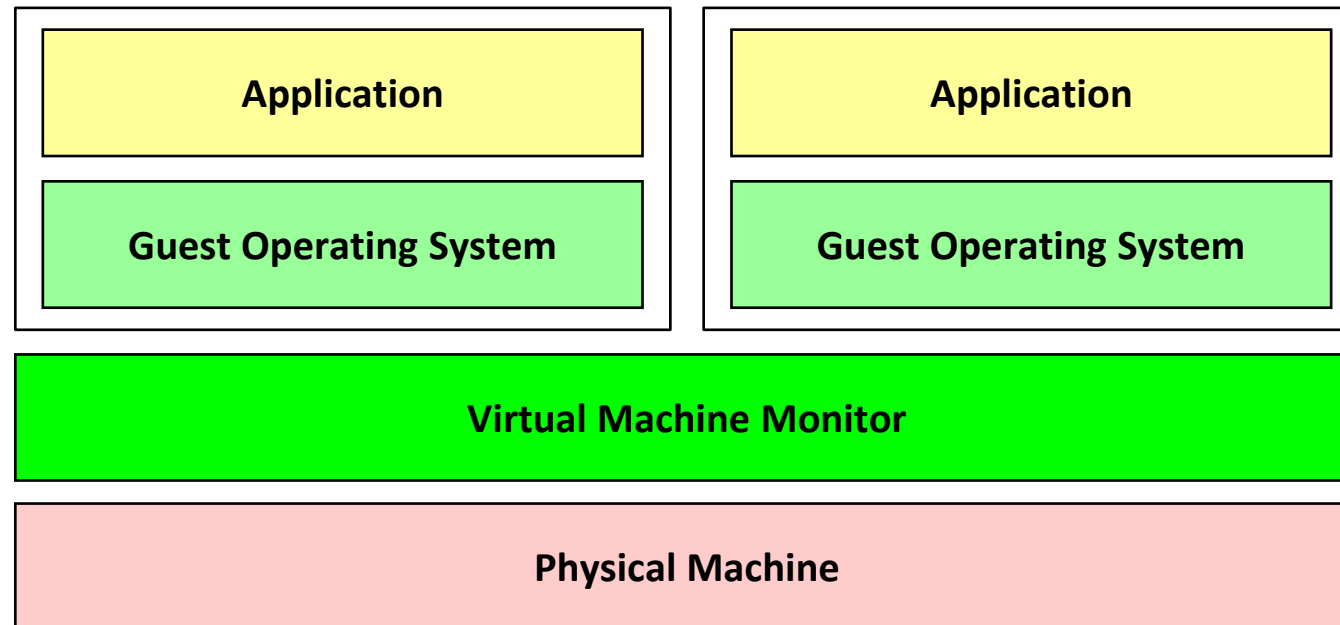
- **Disco: Stanford research project (SOSP '97)**
 - Run commodity OSes on scalable multiprocessors
 - Focus on high-end: NUMA, MIPS, IRIX
- **Commercial virtual machines for x86 architecture**
 - VMware Workstation (→ EMC/Dell → Broadcom) (1998 -)
 - Connectix VirtualPC (now Microsoft)
- **Research virtual machines for x86 architecture**
 - Xen (SOSP '03), plex86
- **OS-level virtualization**
 - FreeBSD Jails, Linux Docker

Why Virtual Machines?

- Create the illusion of multiple VMs
- Strong isolation between VM instances
- Software compatibility
- Logical partitioning and server consolidation
- Convenient environment for debugging OSes

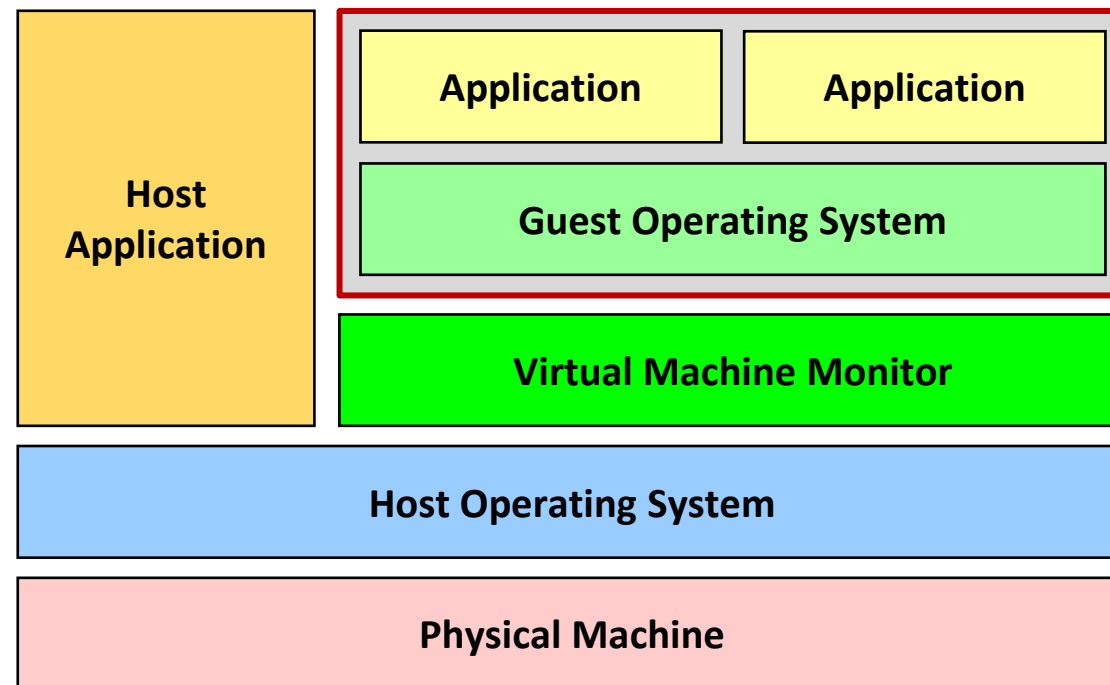
Type I VMM: Bare-Metal Hypervisors

- VMM is implemented directly on the physical hardware
- VMM performs the scheduling and allocation of the system's resources
- IBM VM/370, Disco, VMware ESX Server, Xen, Hyper-V



Type II VMM: Hosted Hypervisors

- VMMs are built completely on top of a host OS
- A guest OS runs as a process on the host
- VMware Workstation/Player, Virtual Box, Parallels Desktop for Mac, KVM?



Related Technologies

■ Complete machine simulators

- Bochs (x86), SimOS (MIPS R4000/R10000), SimICS (x86, Alpha, ARM, IA-64, MIPS, PowerPC, Sparc), Qemu (Alpha, ARM, x86, MIPS, Sparc, RISC-V, PowerPC, ...)
- Portable: Runs instructions purely in software
- Slow (e.g., 100x slow down for Bochs)
- Portability vs. performance

■ ABI/API emulators

- WINE (Windows Emulator or Wine Is Not an Emulator): Port of Windows API to X-windows/Unix.
- Focuses on getting system call for a particular operating system's interface.

■ High-level language VMs: Java VM, etc.

Popek/Goldberg Theorem

For any conventional third-generation computer, a virtual machine monitor may be constructed if the set of **sensitive instructions** for that computer is a subset of the set of **privileged instructions**.

-- G. Popek and R. Goldberg, "Formal Requirements for Virtualizable Third-Generation Architectures," CACM, 1974.

- An instruction is control-sensitive if it can update the system state
- An instruction is behavior-sensitive if its semantics depend on the actual values set in the system state
- An instruction is privileged if it can only be executed in supervisor mode and causes a trap when attempted from user mode

$$\{\textit{control-sensitive}\} \cup \{\textit{behavior-sensitive}\} \subseteq \{\textit{privileged}\}.$$

Violations in IA-32

- 17 problematic instructions that are sensitive and yet unprivileged

Group	Instructions
Access to interrupt flag	pushf, popf, iret
Visibility into segment descriptors	lar, verr, verw, lsl
Segment manipulation instructions	pop <seg>, push <seg>, mov <seg>
Read-only access to privileged state	sgdt, sldt, sidt, smsw
Interrupt and gate instructions	fcall, longjump, retfar, str, int <n>

Intel Virtualization Technology (VT-x)

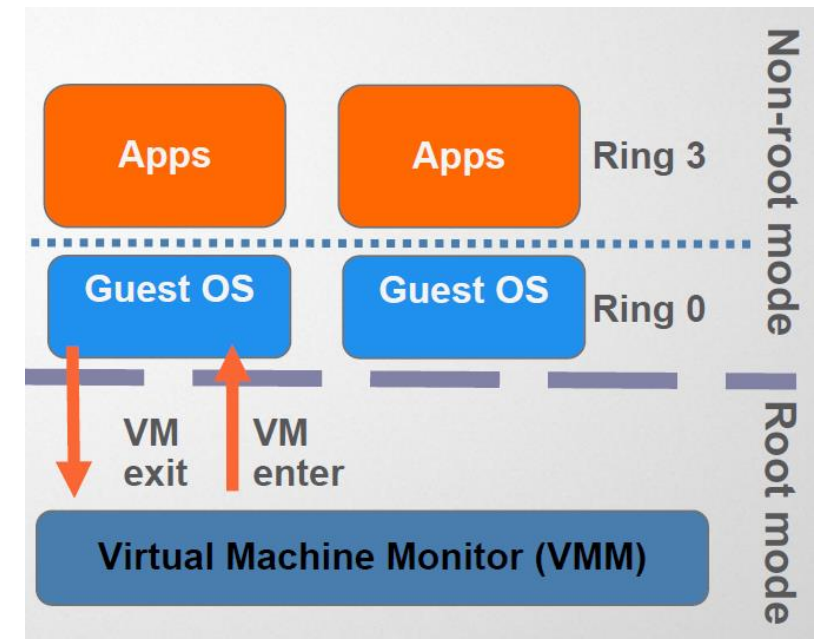
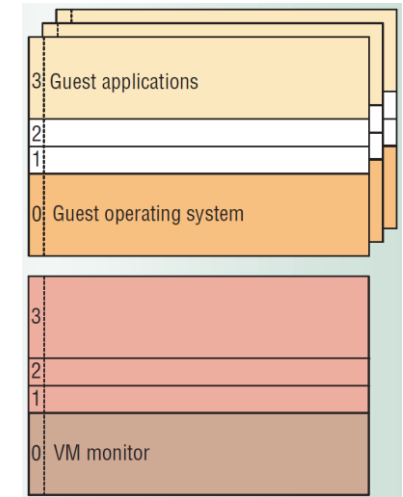
A central design goal for Intel Virtualization Technology is to eliminate the need for CPU paravirtualization and binary translation techniques, and thereby enable the implementation of VMMs that can support a broad range of unmodified guest operating systems while maintaining high levels of performance.

-- R. Uhlig et al., "Intel Virtualization Technology," IEEE Computer, 2005

- Virtual machine extensions (VMX) introduced in 2005
- 13 new instructions are added
- Two new VT-x operating modes: VMX non-root and VMX root
- Two new transitions: VM entry and VM exit
- Extended Page Tables (EPT) added in 2008 for memory virtualization

VMX

- **VMX root/non-root operations**
 - A VMM runs in VMX root operation
 - Guest OSes run in VMX non-root operation
 - Both support all four privilege levels
- **Transitions**
 - VM entry: VMX root \rightarrow VMX non-root
 - VM exit: VMX non-root \rightarrow VMX root

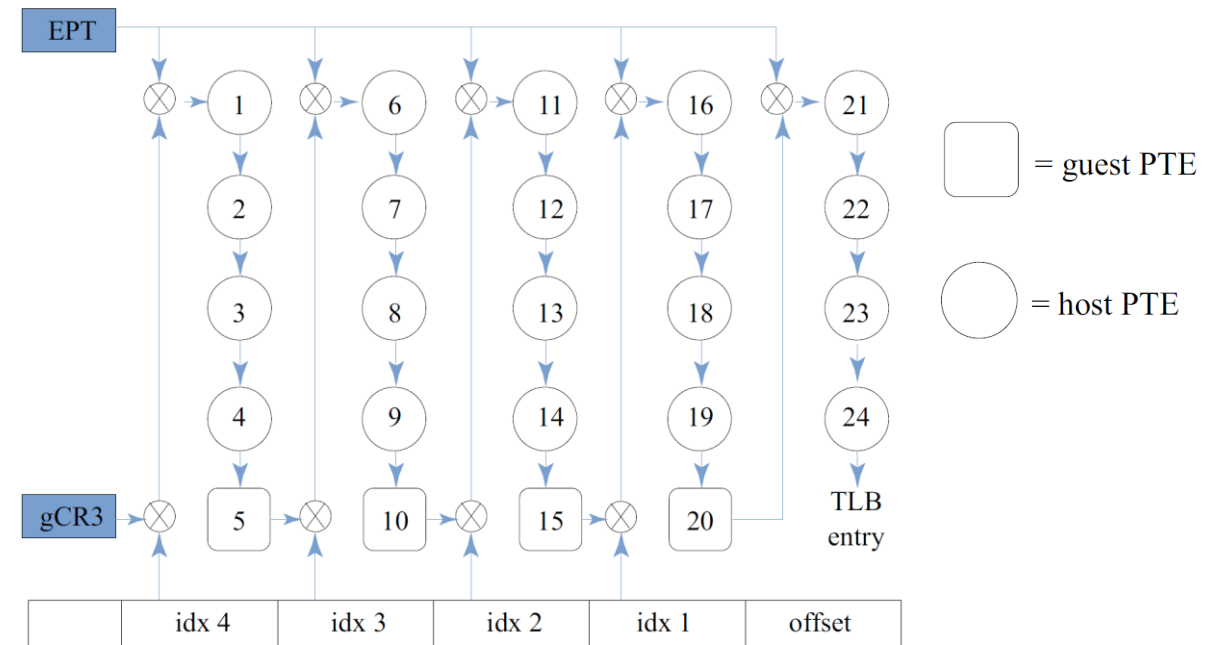
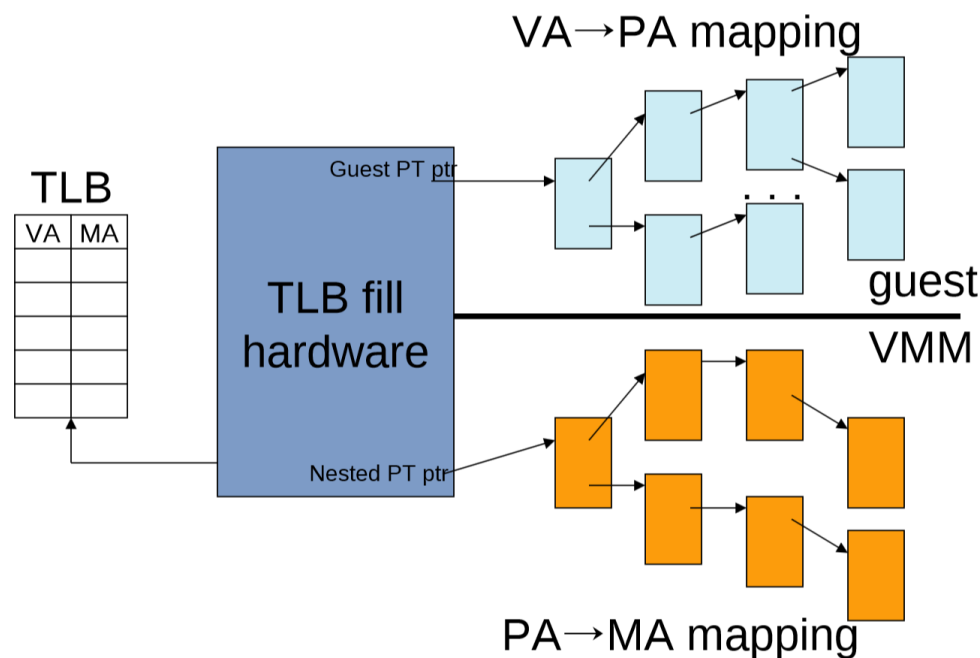


VMCS

- **Virtual-Machine Control Structure**
 - A new data structure that manages VM entries and VM exist and processor behavior in VMX non-root operations
 - Guest-state area vs. host-state area
 - VM entries load processor state from the guest-state area
 - VM exits save processor state to the guest-state area and then load processor state from the host-state area
- **Processor behavior changes in VMX non-root operation**
 - Some instructions cannot be executed in VMX non-root operation because they cause VM exists unconditionally
 - Other instructions, interrupts and exceptions can be configured to cause VM exists conditionally (using VM-execution control fields in VMCS)

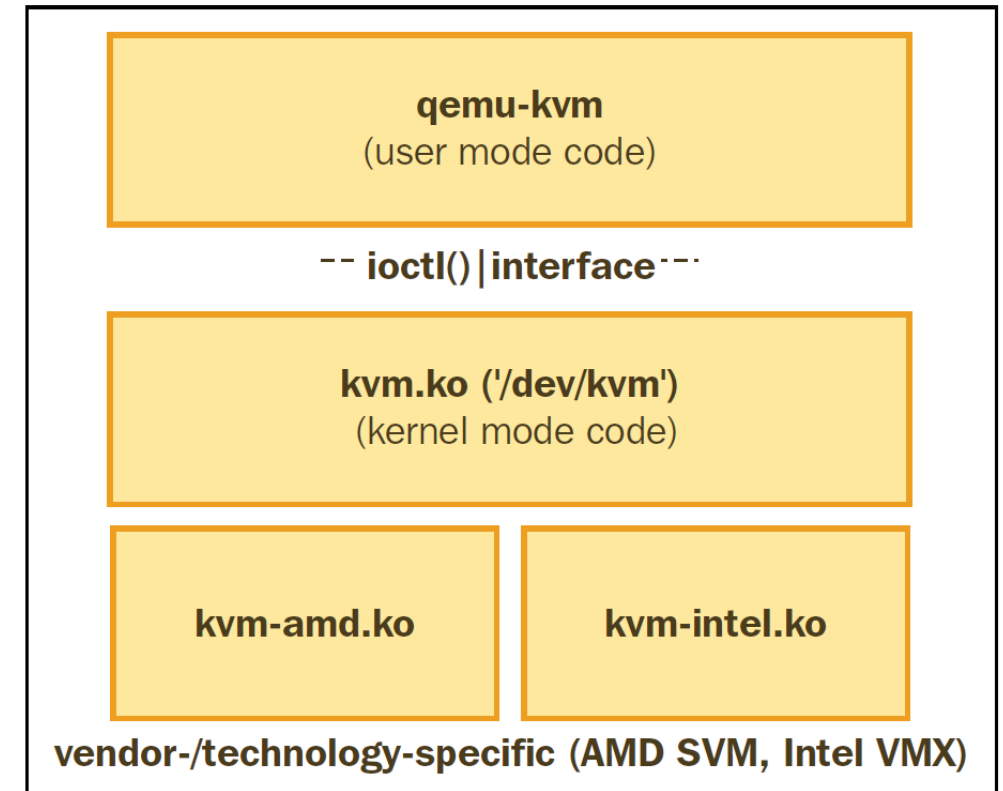
Extended Page Tables (EPT)

- VMM maintains PPN→MPN mappings in “nested page tables”
 - For every PPN (guest-physical) accessed during guest page table walk, the hardware also walks nested page tables to determine the corresponding MPN (host-physical)
 - TLB still maps guest-virtual pages to host-physical pages



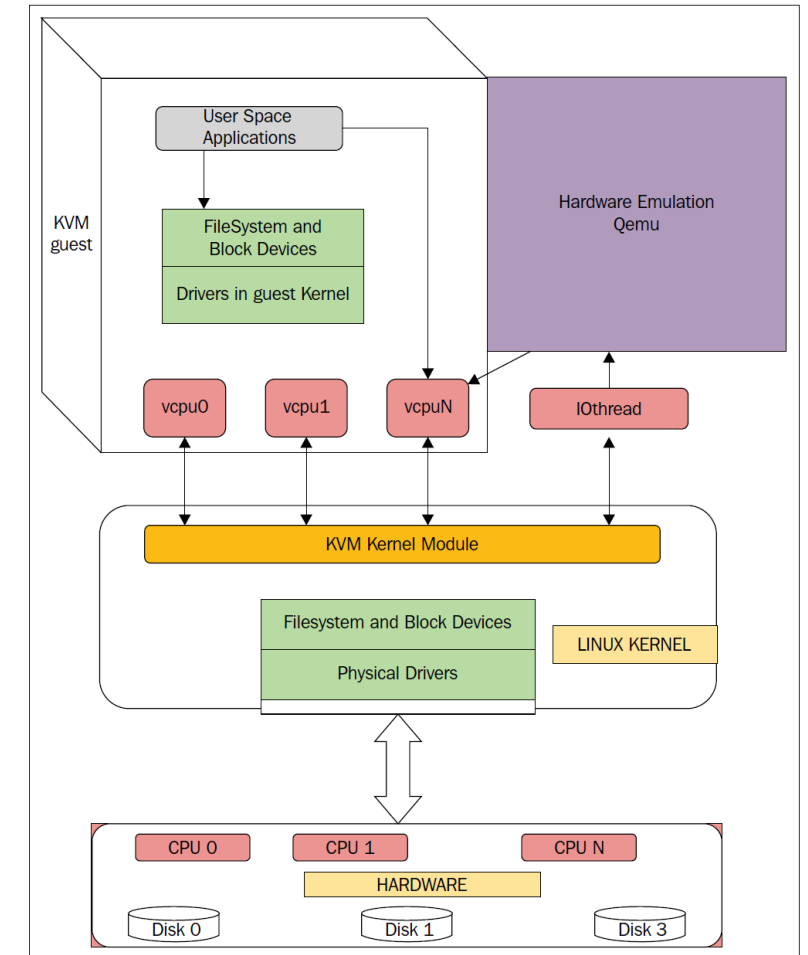
KVM

- Developed by Qumranet
 - Has been part of the Linux kernel since v2.6.20
 - Later Qumranet was acquired by Red Hat
 - Officially supported hypervisor of major commercial Linux distributions
- Requires hardware virtualization capable processors to operate
- KVM turns the standard Linux kernel into a hypervisor



QEMU

- Open source machine emulator and virtualizer
 - Developed by Fabrice Bellard
 - Runs OSes and programs for another CPU ISA using dynamic binary translation or direct execution
 - Emulates a set of devices: disks, networks, VGA, PCIe, serial & parallel ports, USB, ...
 - Runs other management tasks: creating and initializing a virtual machine, BIOS, VM management, etc.



KVM Architecture

- **KVM kernel module (kvm.ko)**
 - Handles the basic CPU platform emulation issues
 - CPU / memory / interrupt virtualization
 - Some chipset emulation (APIC, IOAPIC, etc.)
- **QEMU-KVM**
 - For each and every VM, there is a QEMU process running in the host system
 - Virtual CPUs are executed in the host kernel as POSIX threads
 - Guest RAM is assigned inside the QEMU process's virtual address space
 - Worker threads (iothreads) for virtual network and disk devices
 - QEMU talks to the KVM kernel module using ioctls on /dev/kvm

Execution Flow

