Virtualization Architecture & KVM

Pradeep Kumar Surisetty < psuriset@redhat.com>

Principal Software Engineer

Why we believe KVM is the best virtualization platform

Performance

KVM holds the **Top 6/11** virtual machine consolidation scores on SPECvirt (1)

Lower Cost

customers report up to **70%** savings by using KVM (2)

Cross Platform

Support and certification for leading x86_64 operating systems including **RHEL** and **Microsoft Windows** (4)

Security

EAL4+ Certification (3) plus SE Linux enabling Mandatory Access Control between virtual machines

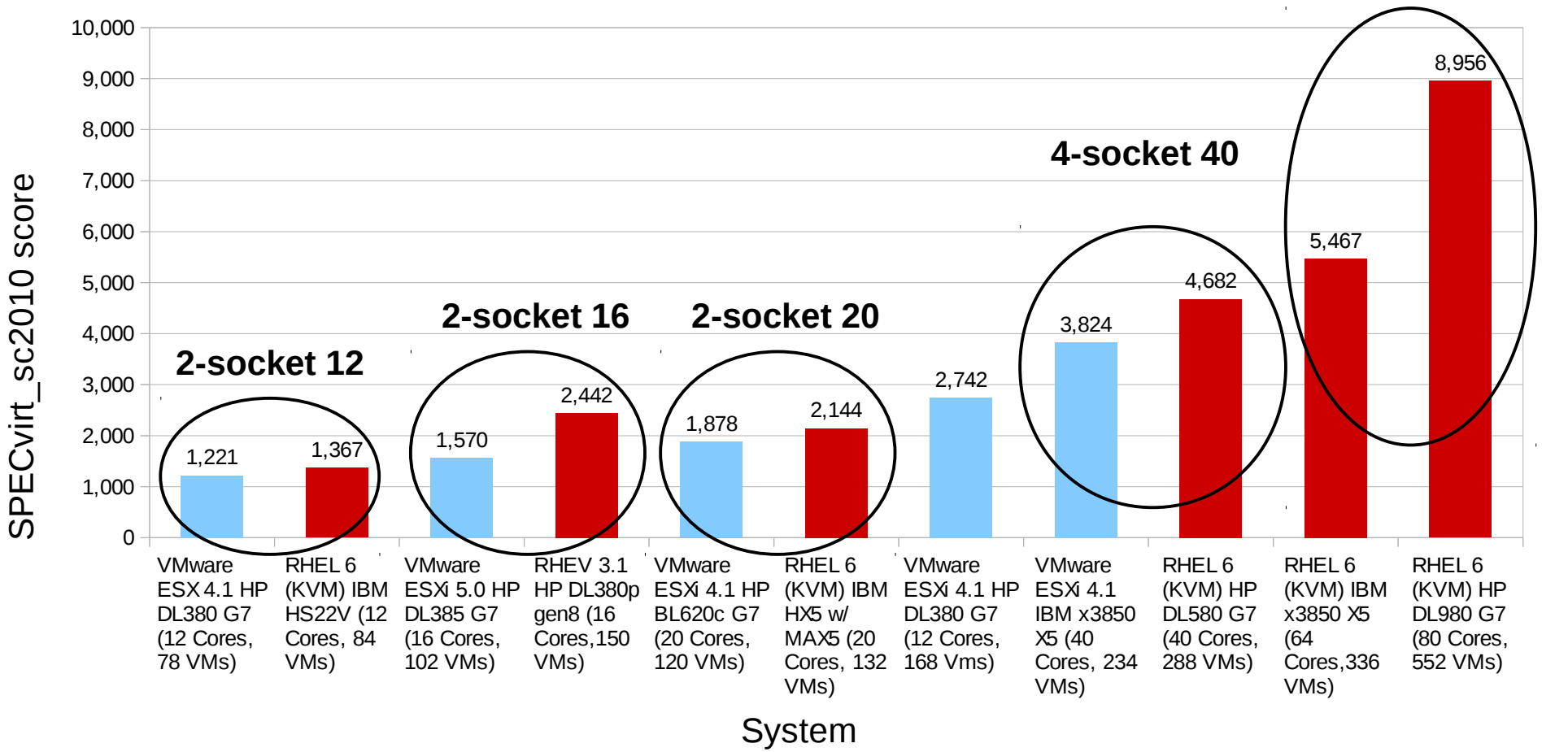
Cloud & Virtualization Management

Red Hat Open Stack for Cloud Virtualization and Red Hat Enterprise Virtualization for data-center Virtualization

(1) Source: SpecVirt_sc2010 results: http://www.spec.org/virt_sc2010/results/specvirt_sc2010_perf.html
(2) Source: Case study on Canary Islands Government migration from VMware to RHEV: http://www.redhat.com/resourcelibrary/case-studies/canary-islands-government-migrates-telecommunications-platform-from-vmware-to-red-hat
(3) Source: http://www.redhat.com/solutions/industry/government/certifications.html
(4) Source: http://www.redhat.com/resourcelibrary/articles/enterprise-linux-virtualization-support

Best SPECvirt_sc2010 Scores by CPU Cores

(As of May 30, 2013)

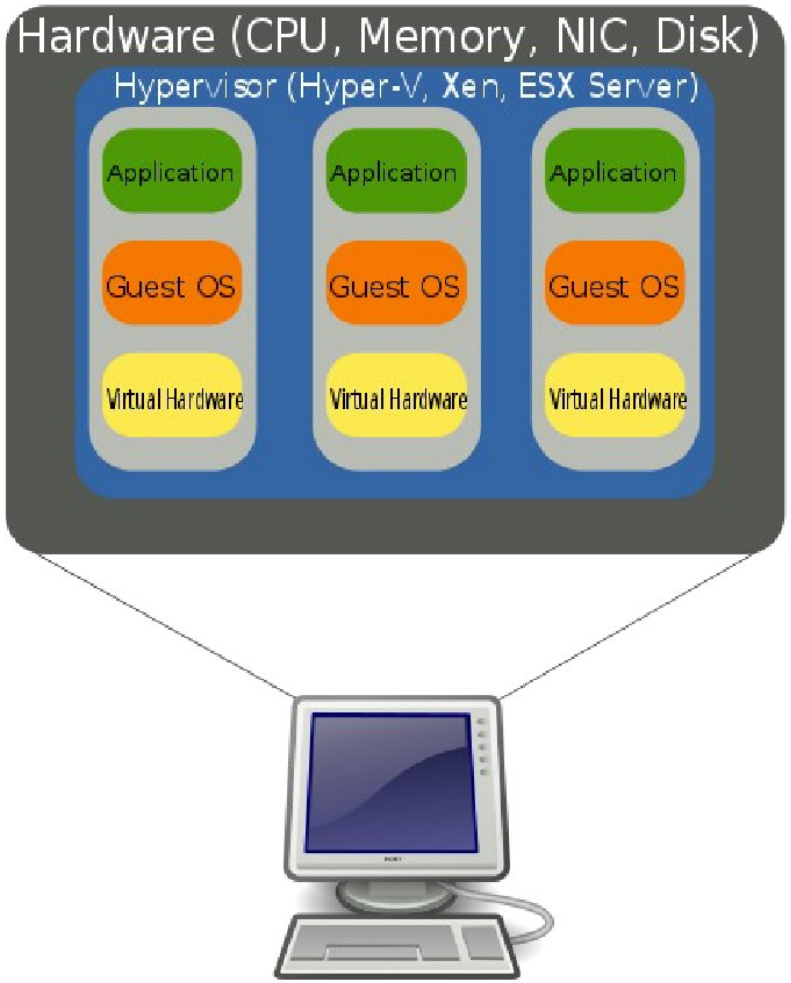


Comparison based on best performing Red Hat and VMware solutions by cpu core count published at www.spec.org as of May 17, 2013. SPEC® and the benchmark name SPECvir_sct® are registered trademarks of the Standard Performance Evaluation Corporation. For more information about SPECvirt_sc2010, see www.spec.org/virt_sc2010/.

Virtualization 101

- Run a computer
- With virtual computers inside

- Shared use
- Security isolation
- Hardware isolation
- Power saving
- Development & testing
- Legacy OS on new hardware



PC Architecture

- Monitor, keyboard, mouse & magic box

- Processor
- Memory
- Disk
- BIOS
- Video card
- USB ports
- Sound card
- PCI bus
- Disk controller
- Clock sources
- Interrupt controllers
- ...

Introducing KVM virtualization

KVM hypervisor runs virtual machines on Linux hosts

- Mature on x86, recent progress on ARM and ppc

Most popular and best supported hypervisor on OpenStack

- https://wiki.openstack.org/wiki/HypervisorSupportMatrix

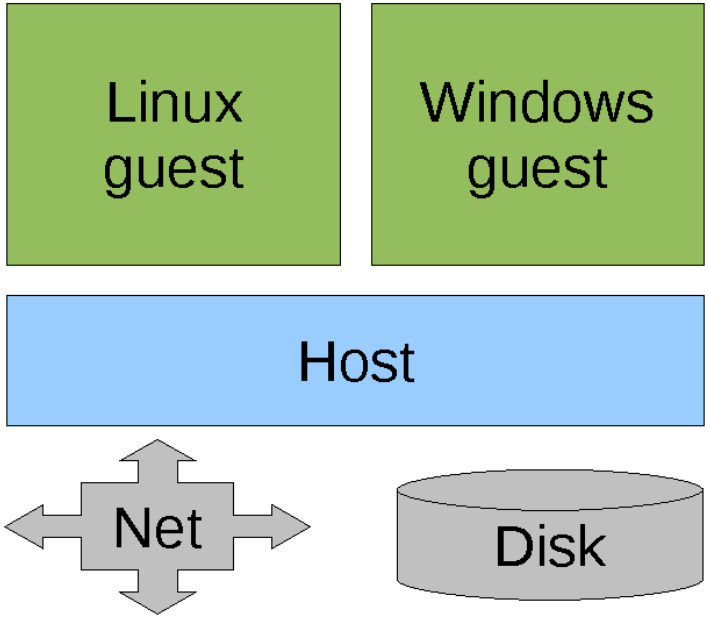
Built in to Red Hat Enterprise Linux

- Qumranet startup created KVM, joined Red Hat in 2008

Virtualization goals

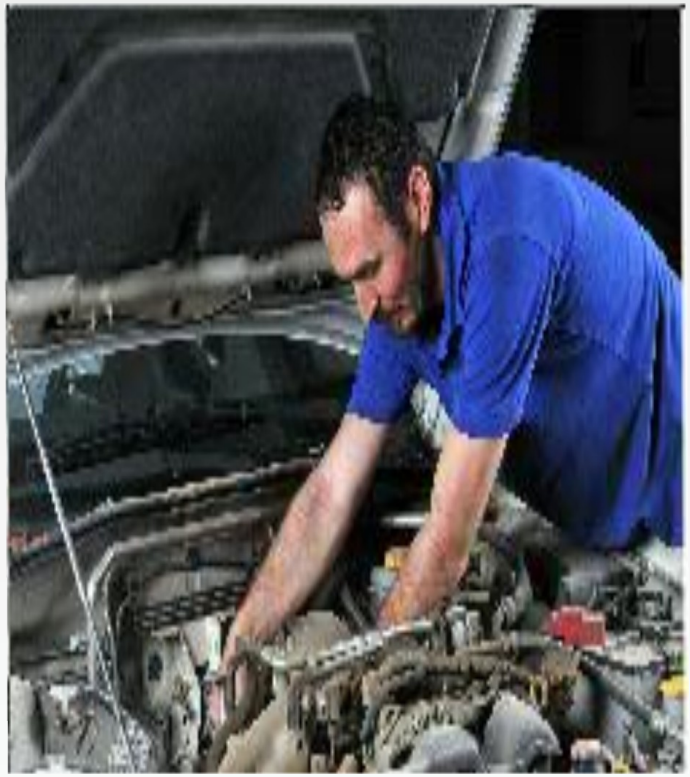
Efficiently and securely running virtual machines on a Linux host

- Linux, Windows, etc guest operating systems
- Access to networking and storage in a controlled fashion



Virtualization goals

- How to Virtualize CPU?
- How to Virutalize Memory?
- How to Virtualize IO?



Virtualizing the X86 Architecture

- x86 architecture is difficult to virtualize
- CPU implements 4 privilege levels or “rings” - 0 thru 3
  - Privileged kernels calls run in ring 0
  - Applications / userspace run in ring 3

Virtualizing the X86 Architecture

- To virtualize x86 operating systems we need to handle privileged calls
  - “Ring compression” or “de-privileging”
- Four techniques for virtualizing x86 platform
  - Full emulation
  - Binary Translation
  - Paravirtualization
  - Hardware Assisted Virtualization

Virtualizing the X86 Architecture

- Full Emulation

  Emulate the entire machine, including CPU using software

  - Yields very more performance
  - Rarely used due to large overhead
  - Some niche use cases

    eg. Running x86 software on PPC platforms

Virtualizing the X86 Architecture

- Binary Translation

  On the fly translation of privileged kernel instructions

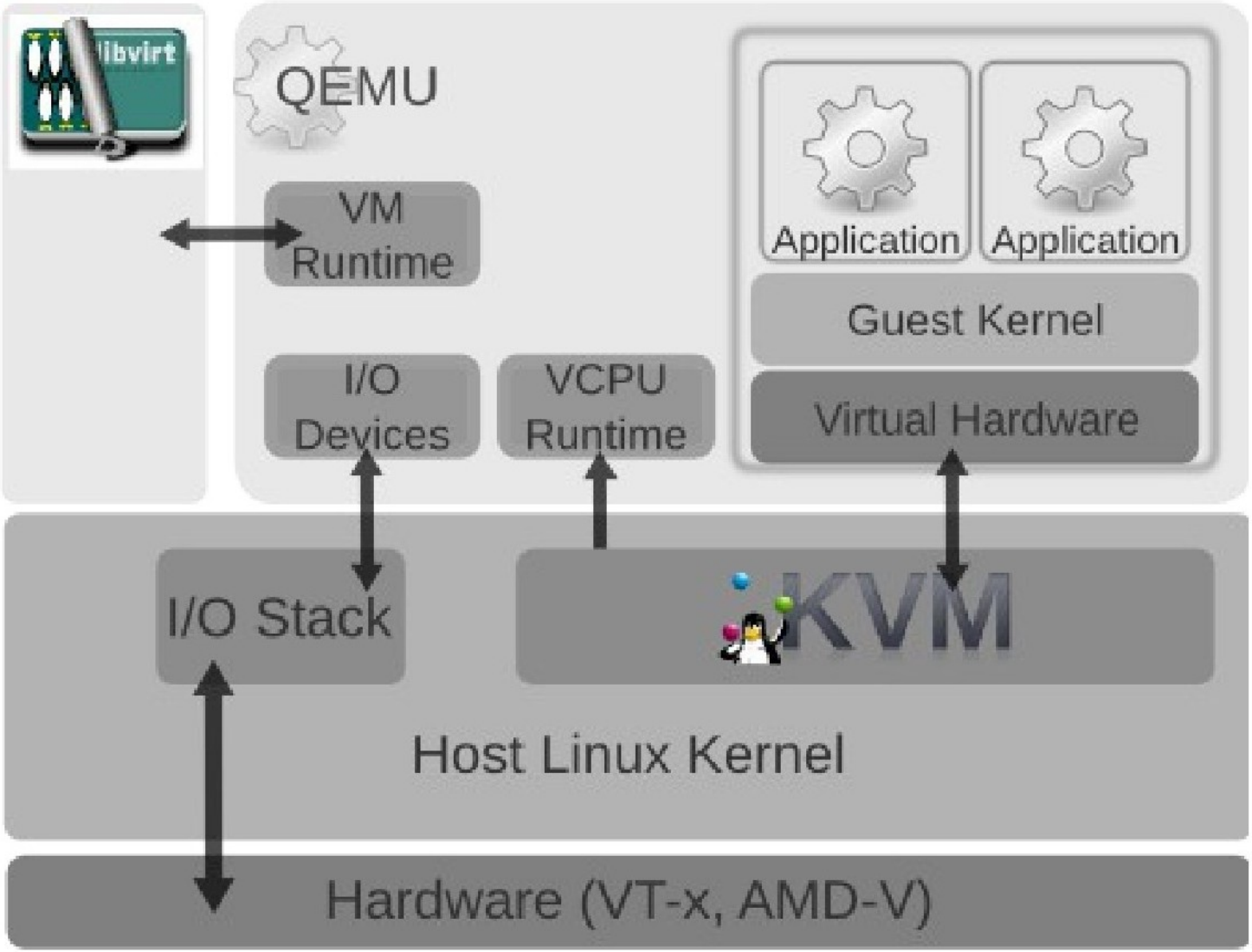
  - Unprivileged instructions run directly on CPU
  - Hypervisor reads ahead and re-writes privileged instructions
  - Redirects calls to the hypervisor
  - Also known as “Scan Before Executing”

Virtualizing the X86 Architecture

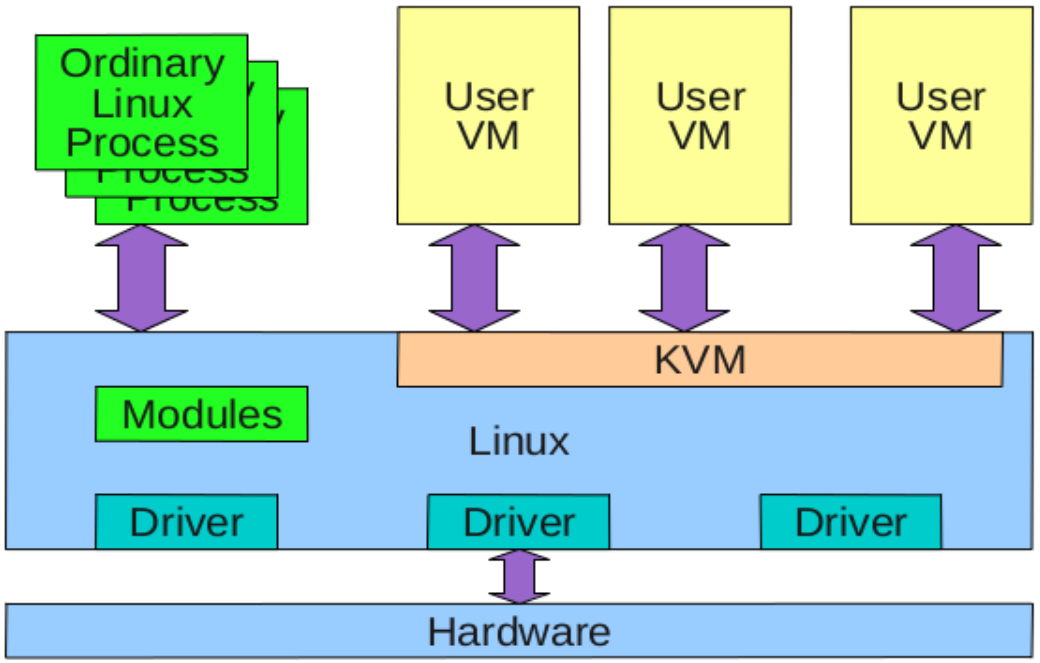
- Paravirtualization

  Modify guest operating system to talk directly to the hypervisor
  - Guest OS kernel is modified to remove privileged instructions
    - Replaced with direct calls to the hypervisor
  - Advantages
    - Improved IO and resource scheduling -> Improved performance
  - Disadvantages
    - Requires changes to the guest operating system -> new OS Kernel
      - Another kernel/OS to test and certify

KVM Architecture



Quick Overview - KVM Architecture



Qemu

- Quick EMUlator
- Written by Fabrice Bellard
- Emulates everything inside a PC, or other architecture
    - CPU, memory, disk, video, etc...
- Typically runs 10x as slow as running on bare metal
    - Emulated
    - Read instruction, simulate what it would do, etc...
- Emulation not desirable
- GPLv2 licensed
- Basis for KVM, Xen, etc...

Virtualizing the X86 Architecture

- Hardware Assisted Virtualization

  Extensions to x86 architecture to support virtualization

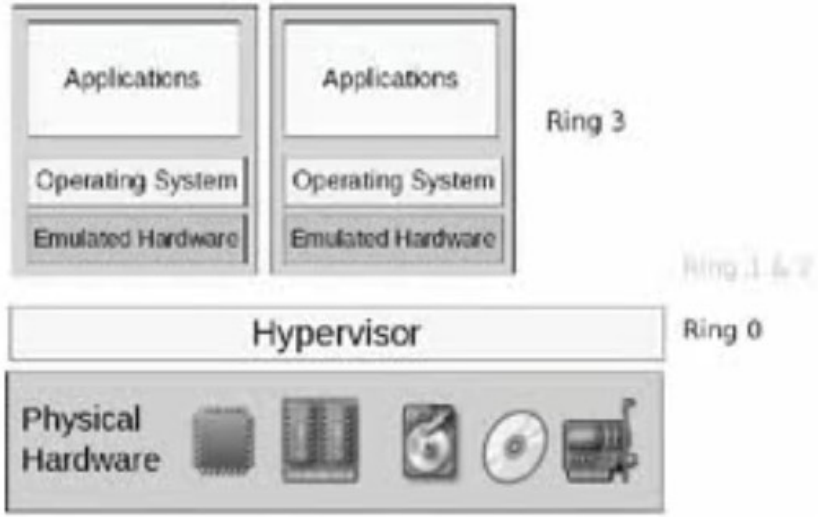
  - AMD-V
  - VT-X
  - Available since 2006, now in all mainstream platforms

- Offloads “Ring compression” to CPU
  - Effectively provides new privilege level
  - Used by hypervisor to help trap and handle privileged instructions

Virtualizing the X86 Architecture

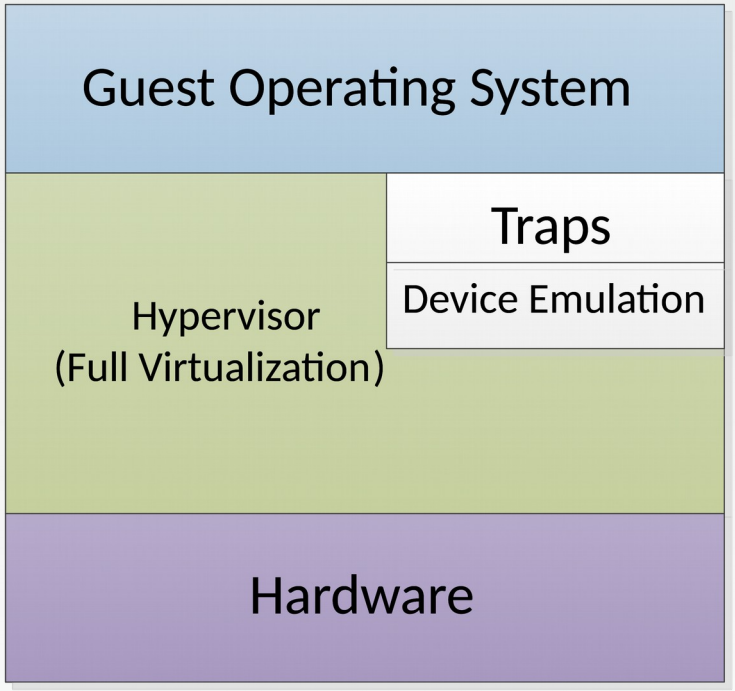
- Hypervisor runs in ring 0
- Virtual machines run in ring 3

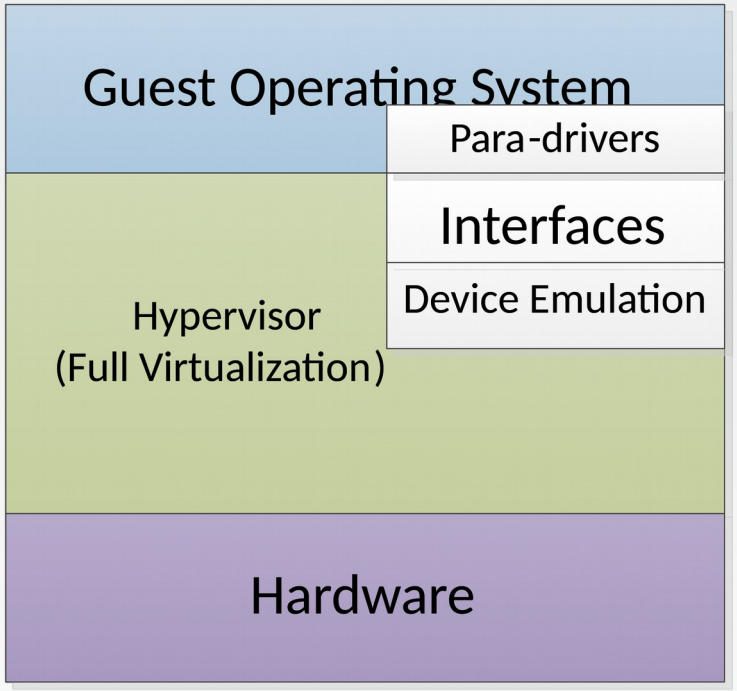
Operating system in Virtual machine makes calls privileged instructions
Will cause a machine fault



Hardware Assisted Virtualization

- First generation – CPU Virtualization
- Second generation – Memory Management
  - Offloads memory page table management to CPU / chipset
  - Provides significant performance improvement
  - Intel : Extended Page Tables (EPT)
  - AMD : Rapid Virtualization Indexing (RVI) - previously called NPT


Guest Operating System
Traps
Device Emulation
Hypervisor
(Full Virtualization)
Hardware

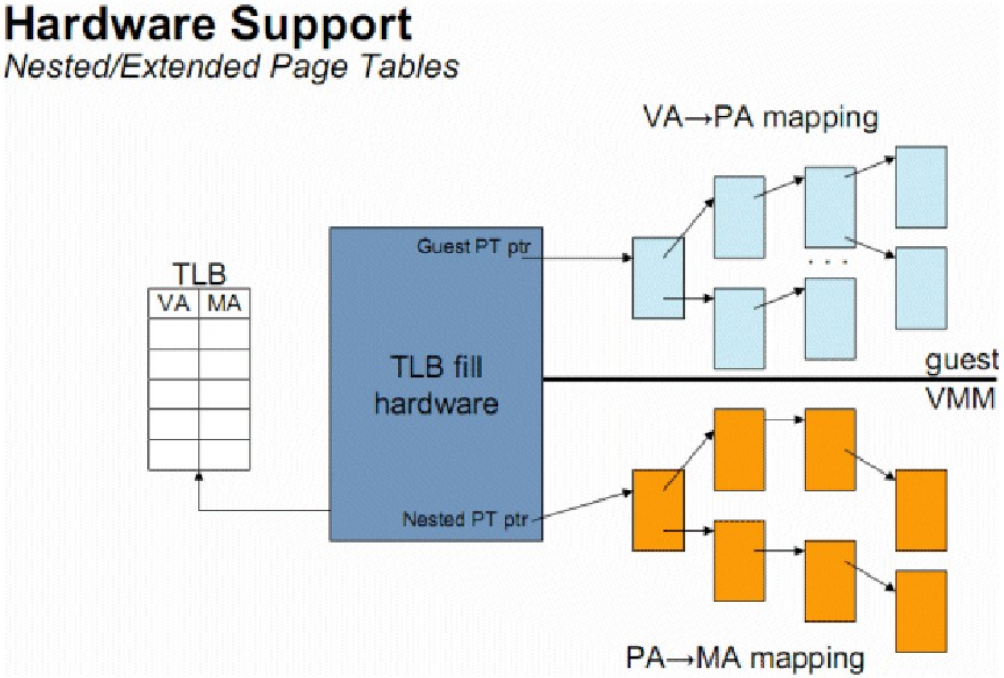

Guest Operating System
Para-drivers
Interfaces
Device Emulation
Hypervisor
(Full Virtualization)
Hardware

Shadow Page Tables

- KVM keeps a second set of page tables
  - From virtual memory in guest
  - To physical memory address on real hardware
- One set of page tables for every process in every guest
- Page table memory in guest marked read-only
  - When the guest writes it,
  - A page fault is triggered,
  - The host emulates the write,
  - And writes a corresponding entry in the shadow page table
- High overhead
  - Especially for fork/exec heavy workloads

EPT / NPT

- A hardware solution to the problem, by Intel and AMD
- Two sets of page tables
  - Guest process tables, maintained by guest OS
  - VM tables, maintained by host OS
- CPU does two lookups on TLB miss
  - Some overhead, but better than shadow PT



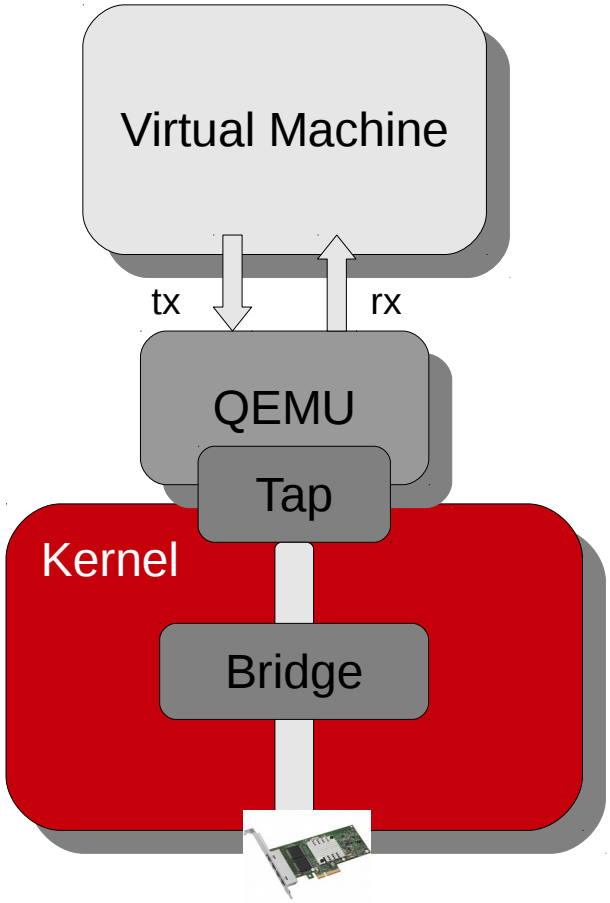
KVM Philosophy

- KVM is a Linux kernel module, used with a modified QEMU binary
  - KVM benefits from Linux performance enhancements
- Use QEMU when possible
  - KVM started out as QEMU + minimal kernel driver
  - Code already exists
  - qemu-kvm binary for use with kvm
  - Improvements shared with wider qemu community
- Implement things in kernel when needed
  - Only possible in kernel
  - Performance requires kernel code

KVM I/O Architecture

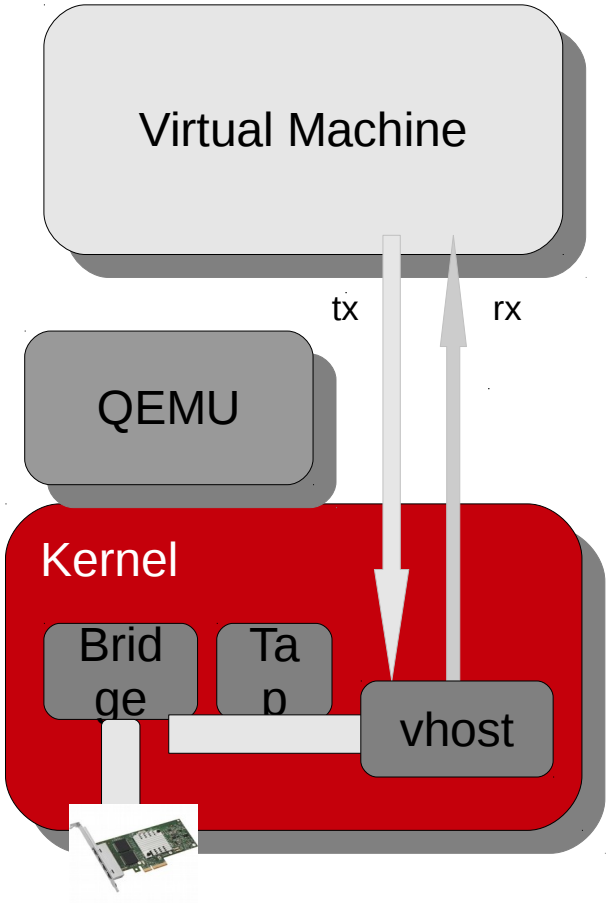
Emulated Devices

- Native drivers
- Compatibility over performance



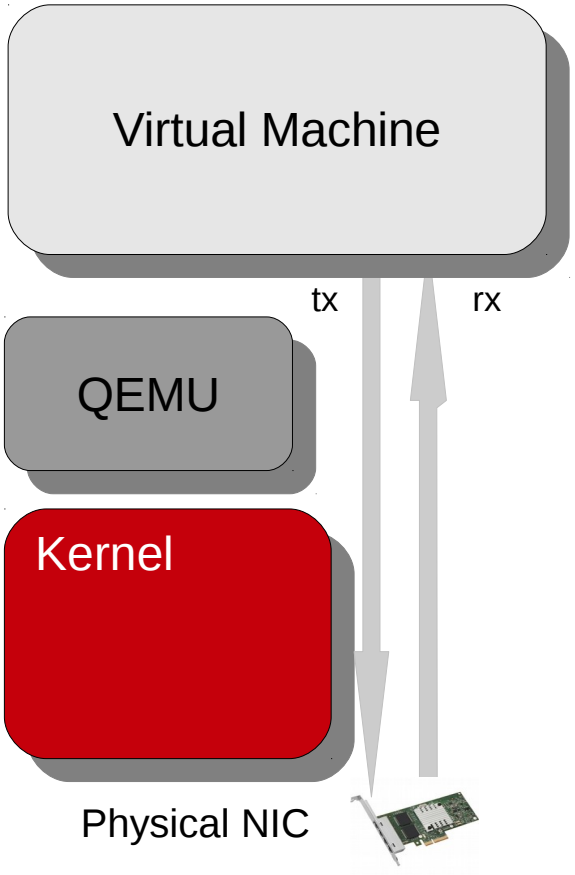
Virtio Devices

- Paravirtualized
- Performance over compatibility

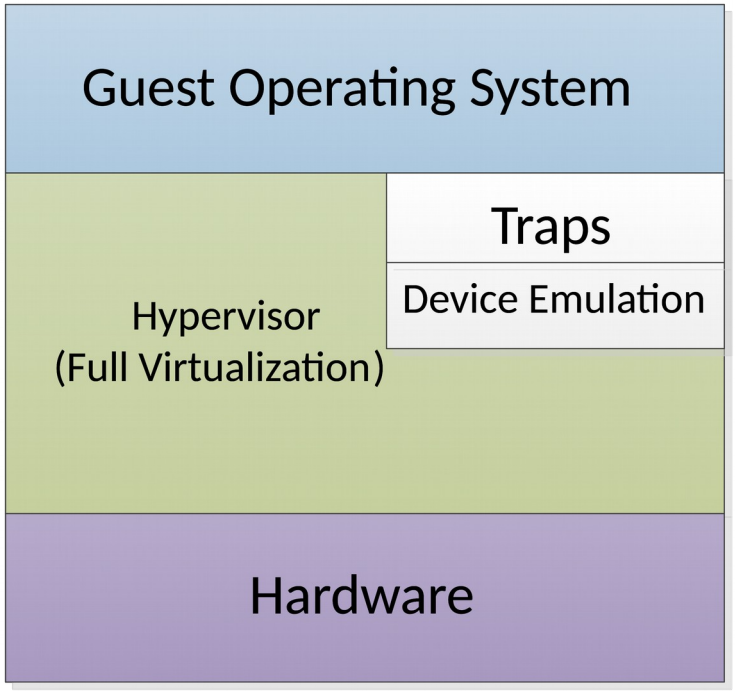


Device Assignment

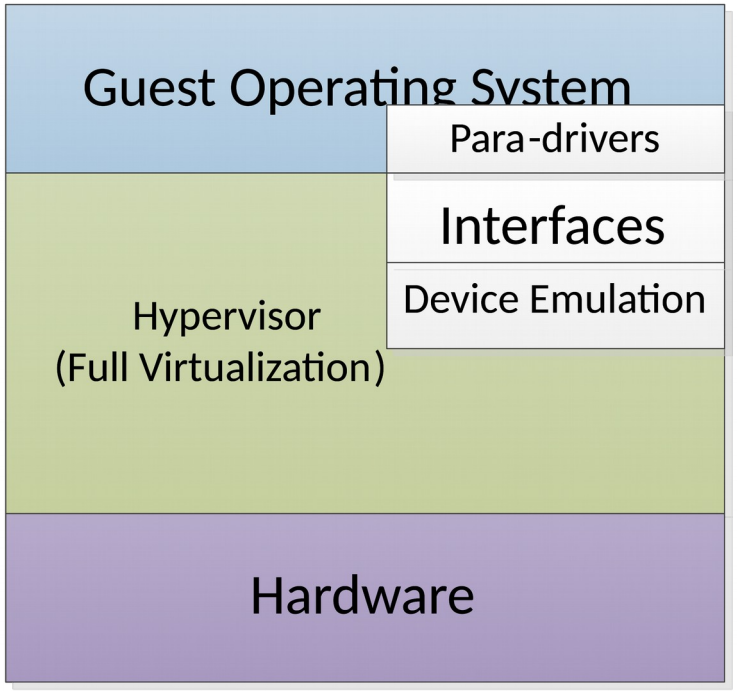
- Native drivers
- Compatibility and Performance



Storage transport choices in KVM



Full virtualization



Para-virtualization

Storage transport choices in KVM

- **Full virtualization : IDE, SATA, SCSI**
  - Good guest compatibility
  - Lots of trap-and-emulate, bad performance
- **Para virtualization: virtio-blk, virtio-scsi**
  - Efficient guest ↔ host communication through virtio ring buffer (virtqueue)
  - Good performance
  - Provide more virtualization friendly interface, higher performance.
  - In AIO case, io_submit() is under the global mutex

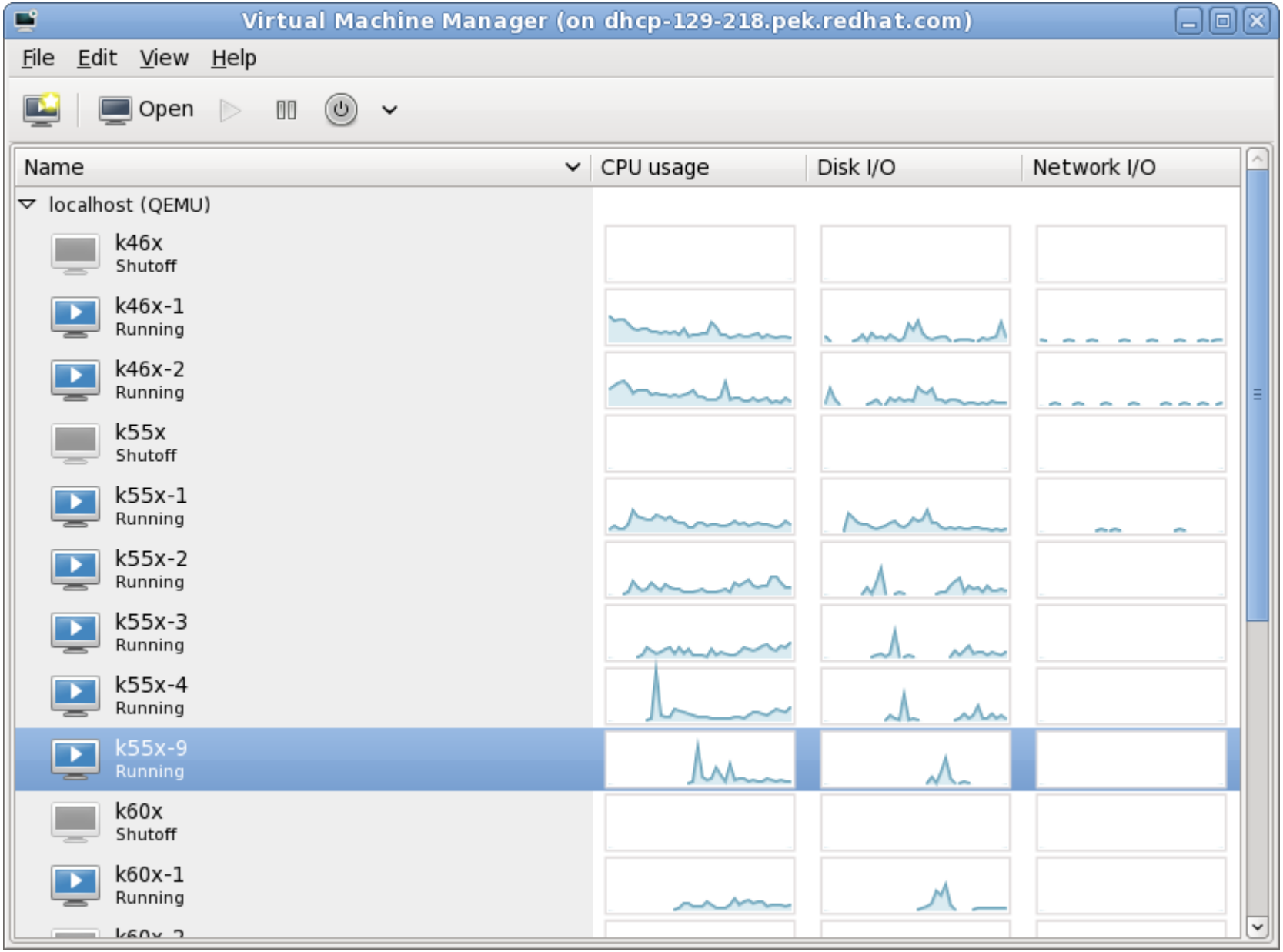
Storage transport choices in KVM

- **Device assignment (Passthrough)**
  - Pass hardware to guest, high-end usage, high performance
  - Limited Number of PCI Devices
  - Hard for Live Migration

Conclusions

- KVM uses QEMU for non-performance critical things
  - KVM developers part of QEMU community
- KVM uses Linux as a hypervisor
  - KVM developers part of Linux community
  - KVM gets Linux hardware support
  - KVM gets Linux performance improvements
- Intel and AMD help solve some of the issues
  - Great performance
  - Still a lot done in software
- KVM has some of the best hardware support and performance
- Everything tested by a large community
  - Stable software

Virt-Manager



Special Thanks
to
Andrew Theurer
Stefan Hajnoczj

Thanks

Irc: #psuriset

Blog: psuriset.com

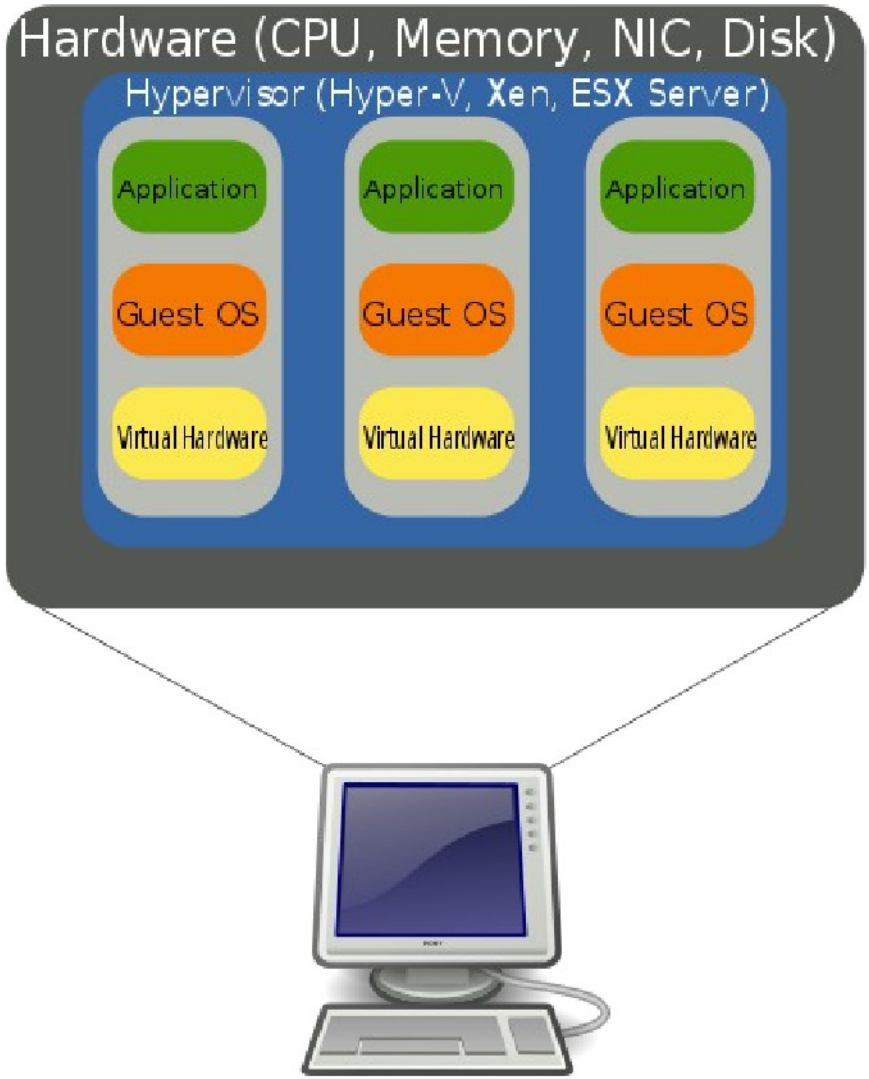
Agenda

- Virtualization 101
- PC Architecture
- Qemu
- KVM Architecture
- X86 Hardware Virtualization Enablers
- KVM Advanced Features
- Conclusions

Virtualization 101

- Run a computer
- With virtual computers inside

- Shared use
- Security isolation
- Hardware isolation
- Power saving
- Development & testing
- Legacy OS on new hardware

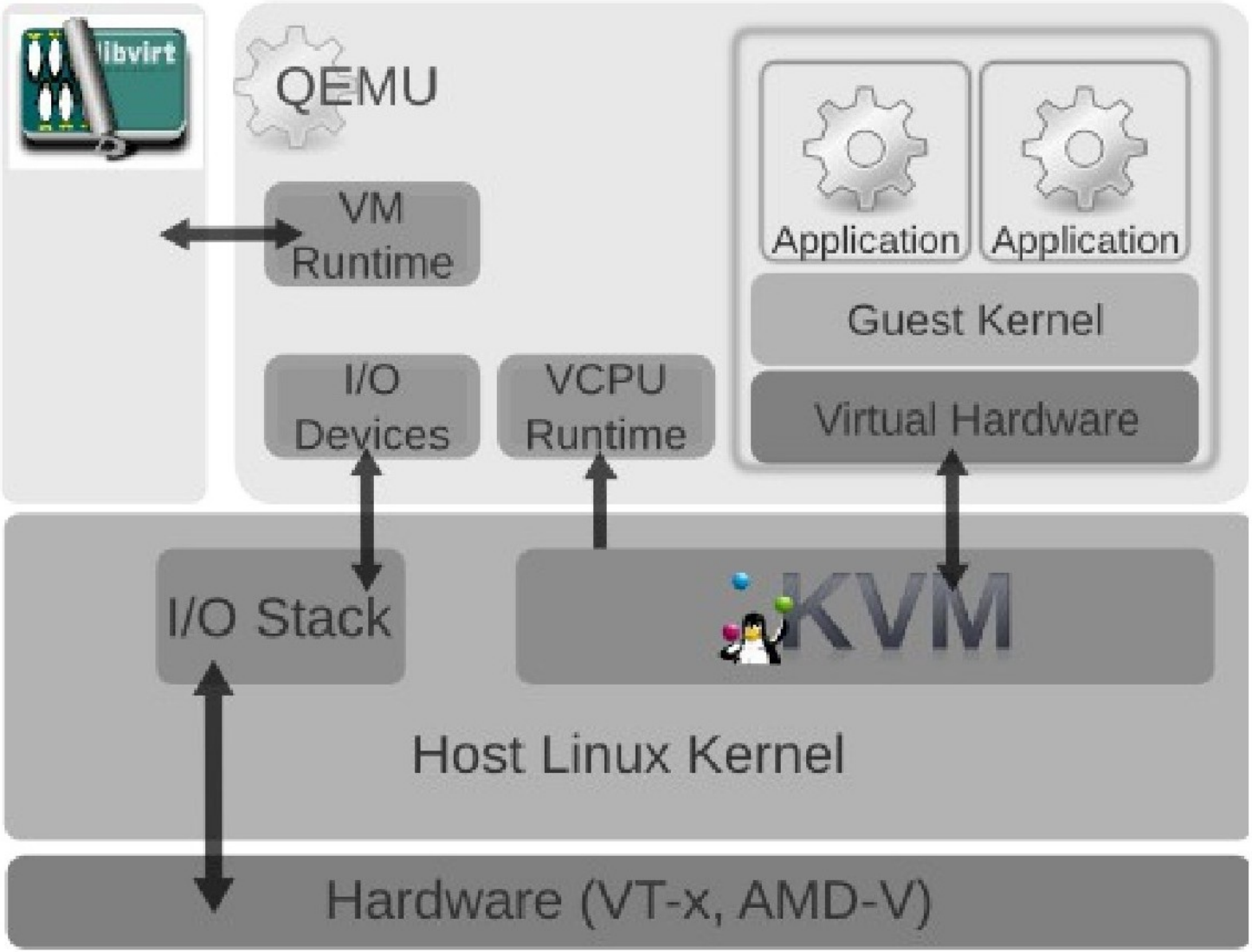


PC Architecture

- Monitor, keyboard, mouse & magic box

- Processor
- Memory
- Disk
- BIOS
- Video card
- USB ports
- Sound card
- PCI bus
- Disk controller
- Clock sources
- Interrupt controllers
- ...

KVM Architecture



KVM Philosophy

- KVM is a Linux kernel module, used with a modified QEMU binary
  - KVM benefits from Linux performance enhancements
- Use QEMU when possible
  - KVM started out as QEMU + minimal kernel driver
  - Code already exists
  - qemu-kvm binary for use with kvm
  - Improvements shared with wider qemu community
- Implement things in kernel when needed
  - Only possible in kernel
  - Performance requires kernel code

Linux as a Hypervisor

- KVM uses Linux as the hypervisor
- KVM gets Linux performance improvements “for free”
  - Transparent Huge Pages
    - ~5-20% faster for some workloads
  - Network stack improvements
    - Can use 10Gbit from a virtual machine
  - NUMA placement
    - Numad, userland NUMA placement daemon
    - Numa/core, NUMA placement in kernel
    - Proper NUMA placement can get 10-20% gain on some workloads
- KVM gets Linux hardware support for free
  - >8TB RAM, >1024 CPUs
  - Latest disk & network controllers
  - KVM runs on many systems

Qemu

- Quick EMUlator
- Written by Fabrice Bellard
- Emulates everything inside a PC, or other architecture
    - CPU, memory, disk, video, etc...
- Typically runs 10x as slow as running on bare metal
    - Emulated
    - Read instruction, simulate what it would do, etc...
- Emulation not desirable
- GPLv2 licensed
- Basis for KVM, Xen, etc...

Libvirt API

- Handles all management and interaction with QEMU
- (VMs) are defined in Libvirt via XML; referred to a “domain”
- Translates XML to command line options for calling QEMU
- Become comfortable with ‘virsh’
- Libvirt XML reference: http://tinyurl.com/libvirt-xml

Processor Virtualization

- X86 architecture was very difficult to virtualize
  - CPU has kernel & user execution mode (rings 0 & 3)
  - Some instructions behave differently in kernel vs user mode
  - Emulation is slow
  - Binary translation is complex
- Hardware assisted virtualization
  - Adds new privilege levels below kernel mode
  - Can run guests directly on the CPU
  - Traps to host for any exception
- Intel VT-x & AMD-V
  - VMX in /proc/cpuinfo flags
  - SVM in /proc/cpuinfo flags

Intel VT-x overview

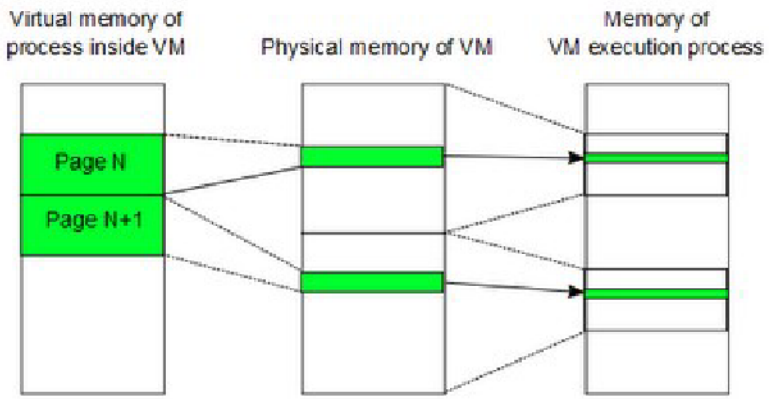
- Virtual Machine Control Structure (VMCS)
  - Guest State (registers, memory, interrupts, ...)
  - Host State (where to jump after exit, ...)
  - Configuration
  - ...
- VMPTRLD instruction loads a VMCS into a CPU
- VMPTRST unloads a VMCS from a CPU
- VMREAD / VMWRITE to inspect & modify a current VMCS
- VMLAUNCH instruction runs a guest from the host
  - VMCS prepared for current guest state
- VMRESUME instruction run on exit from guest
- VMXON enables VT-x
- VMXOFF disables VT-x

Intel VT-x trap to host

- With VT-x, the CPU can directly execute a virtual machine safely
- However, not everything is handled in hardware
  - Interrupt
  - Page fault
  - HLT
  - Pause
  - I/O port access
  - MSR access
  - ...
- On exceptions, CPU traps to host, saving state in VMCS
  - Linux & KVM handle the event
- Switching from the guest to the host is expensive
  - VMCS contains a lot of information
  - Reducing the number of traps help performance

Memory Virtualization

- A virtual machine appears to have physical memory
- Which really is virtual memory
  - Two layers of translation required
- Page tables inside a virtual machine point to virtual addresses
  - One layer of translation provided by guest OS
- Page tables for KVM process point to physical addresses
  - Second layer of translation provided by host OS
- KVM has to connect both layers of translation...
  - Shadow page tables, or hardware assist w/ EPT or NPT

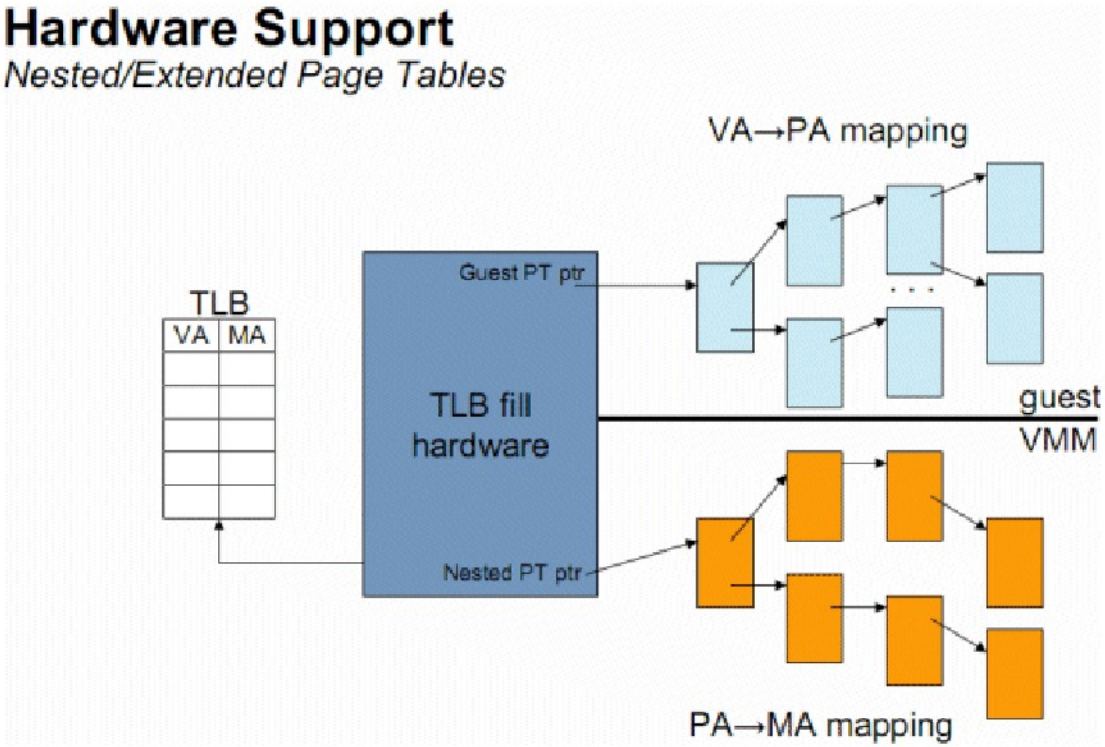


Shadow Page Tables

- KVM keeps a second set of page tables
  - From virtual memory in guest
  - To physical memory address on real hardware
- One set of page tables for every process in every guest
- Page table memory in guest marked read-only
  - When the guest writes it,
  - A page fault is triggered,
  - The host emulates the write,
  - And writes a corresponding entry in the shadow page table
- High overhead
  - Especially for fork/exec heavy workloads

EPT / NPT

- A hardware solution to the problem, by Intel and AMD
- Two sets of page tables
  - Guest process tables, maintained by guest OS
  - VM tables, maintained by host OS
- CPU does two lookups on TLB miss
  - Some overhead, but better than shadow PT



Paravirt devices

- VMCS contains a lot of state
  - VMEXIT is slow
- Normal hardware often takes multiple IO port reads and writes
- Special devices to reduce the number of VMEXITs
  - One communication cycle per action
  - Often multiple blocks/packets/... per action
- Paravirt disk
  - PCI or SCSI
- Paravirt network
- Kvm-clock
  - Paravirtualized for accuracy & steal time accounting
  - Shared memory area + TSC
  - Needs no VMEXIT for gettimeofday
- X2apic
- Memory balloon driver

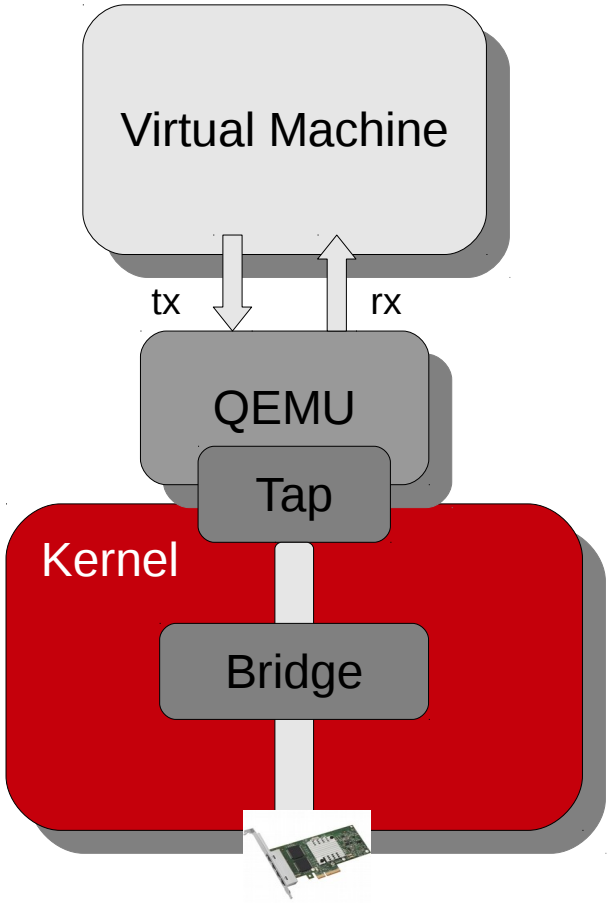
Device Passthrough

- A KVM guest can access hardware directly
  - Performance
  - Special devices
  - Dongles
- PCI device passthrough
  - Requires IOMMU (VT-D or SVM)
  - Address translation
  - Memory protection
- USB
  - Virtual USB bus in guest
  - USB messages get passed to and from the real USB device
  - Mostly for dongles and special data gathering devices

KVM I/O Architecture

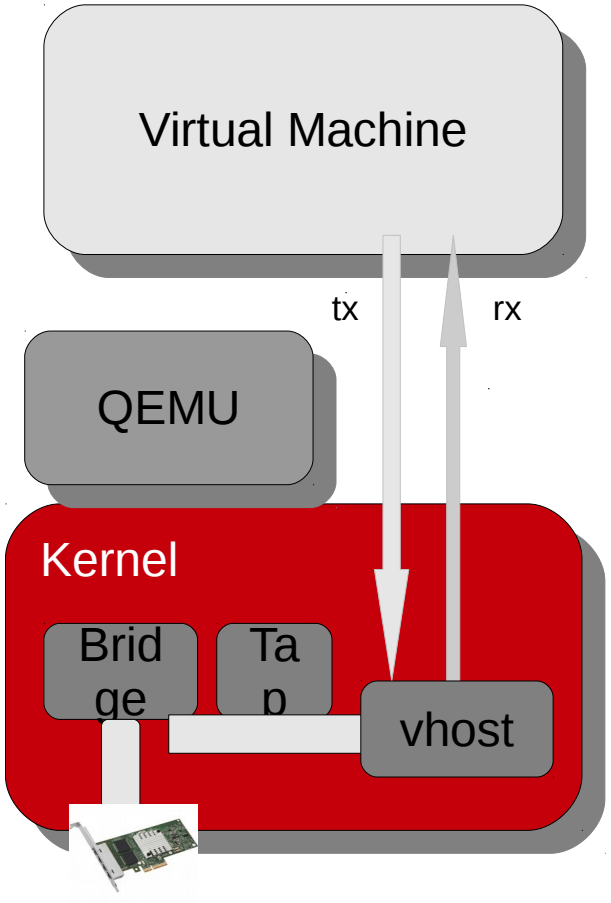
Emulated Devices

- Native drivers
- Compatibility over performance



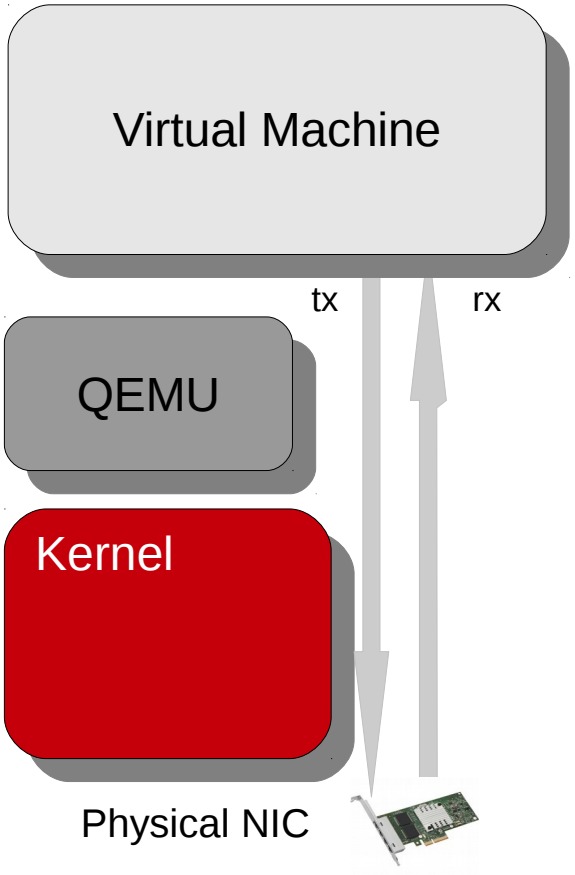
Virtio Devices

- Paravirtualized
- Performance over compatibility



Device Assignment

- Native drivers
- Compatibility and Performance



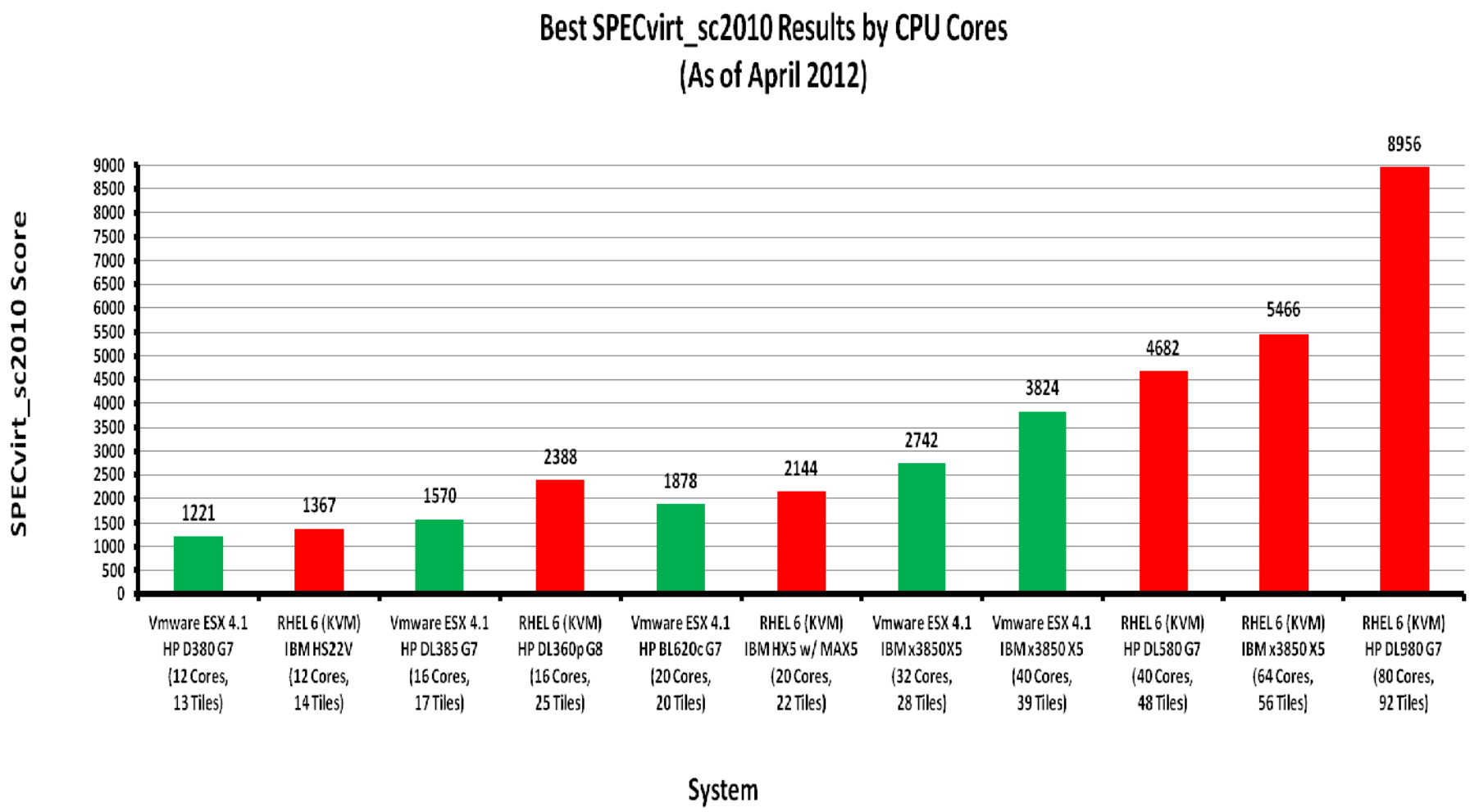
Save, Restore & Migration

- KVM guests can be saved to disk, and restored from disk
- KVM guests can also be migrated
  - Copy all the memory from one place to another
  - Continue running the guest at destination
- Live migration
  - Copy all the memory from one place to another
  - Then, copy the memory that changed since
  - Repeat, until almost all memory has been copied
  - Stop the guest, copy over the last bits of guest state
  - Continue running the guest at destination
- Used for load balancing & hardware maintenance

KVM Performance

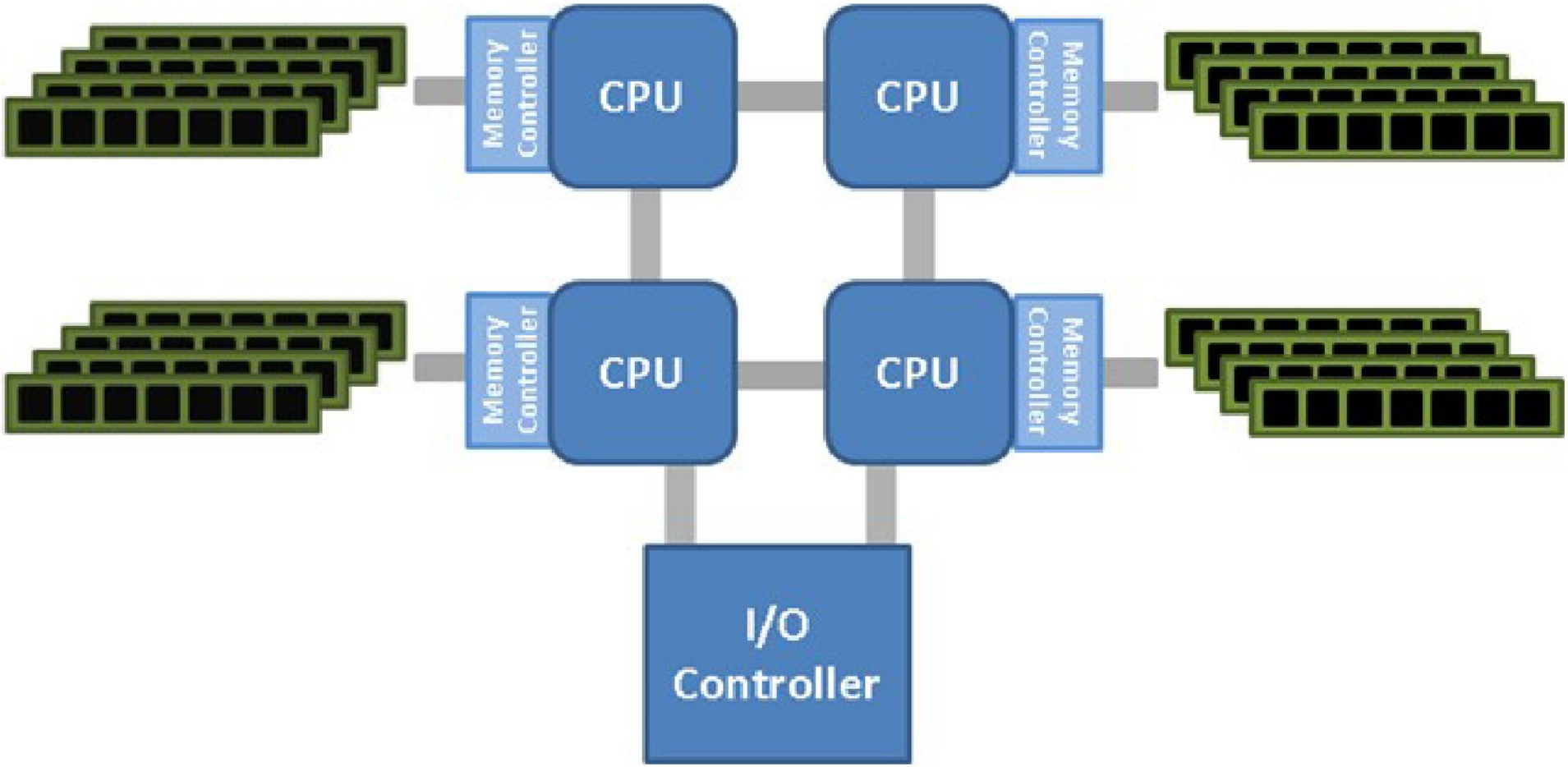
- What does it all add up to?
- Scales to very large systems
  - 1024 host CPUs, 8TB host memory
  - 160 guest CPUs, 2TB guest memory
  - Up to thousands of virtual disks per guest
- SpecVIRT
  - Standardized virtualization benchmark
  - Diverse mixed workload
  - Web, database, mail, idle, etc all in different guests
- NUMA optimizations

SpecVIRT


Best SPECvirt_sc2010 Results by CPU Cores
(As of April 2012)
SPECvirt_sc2010 Score
9000
8500
8000
7500
7000
6500
6000
5500
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
1221
1367
1570
2388
1878
2144
2742
3824
4682
5466
8956
Vmware ESX 4.1 HP D380 G7 (12 Cores, 13 Tiles)
RHEL 6 (KVM) IBM HS22V (12 Cores, 14 Tiles)
Vmware ESX 4.1 HP DL385 G7 (16 Cores, 17 Tiles)
RHEL 6 (KVM) HP DL360p G8 (16 Cores, 25 Tiles)
Vmware ESX 4.1 HP BL620c G7 (20 Cores, 20 Tiles)
RHEL 6 (KVM) IBM HX5 w/ MAX5 (20 Cores, 22 Tiles)
Vmware ESX 4.1 IBM x3850X5 (32 Cores, 28 Tiles)
Vmware ESX 4.1 IBM x3850 X5 (40 Cores, 39 Tiles)
RHEL 6 (KVM) HP DL580 G7 (40 Cores, 48 Tiles)
RHEL 6 (KVM) IBM x3850 X5 (64 Cores, 56 Tiles)
RHEL 6 (KVM) HP DL980 G7 (80 Cores, 92 Tiles)
System

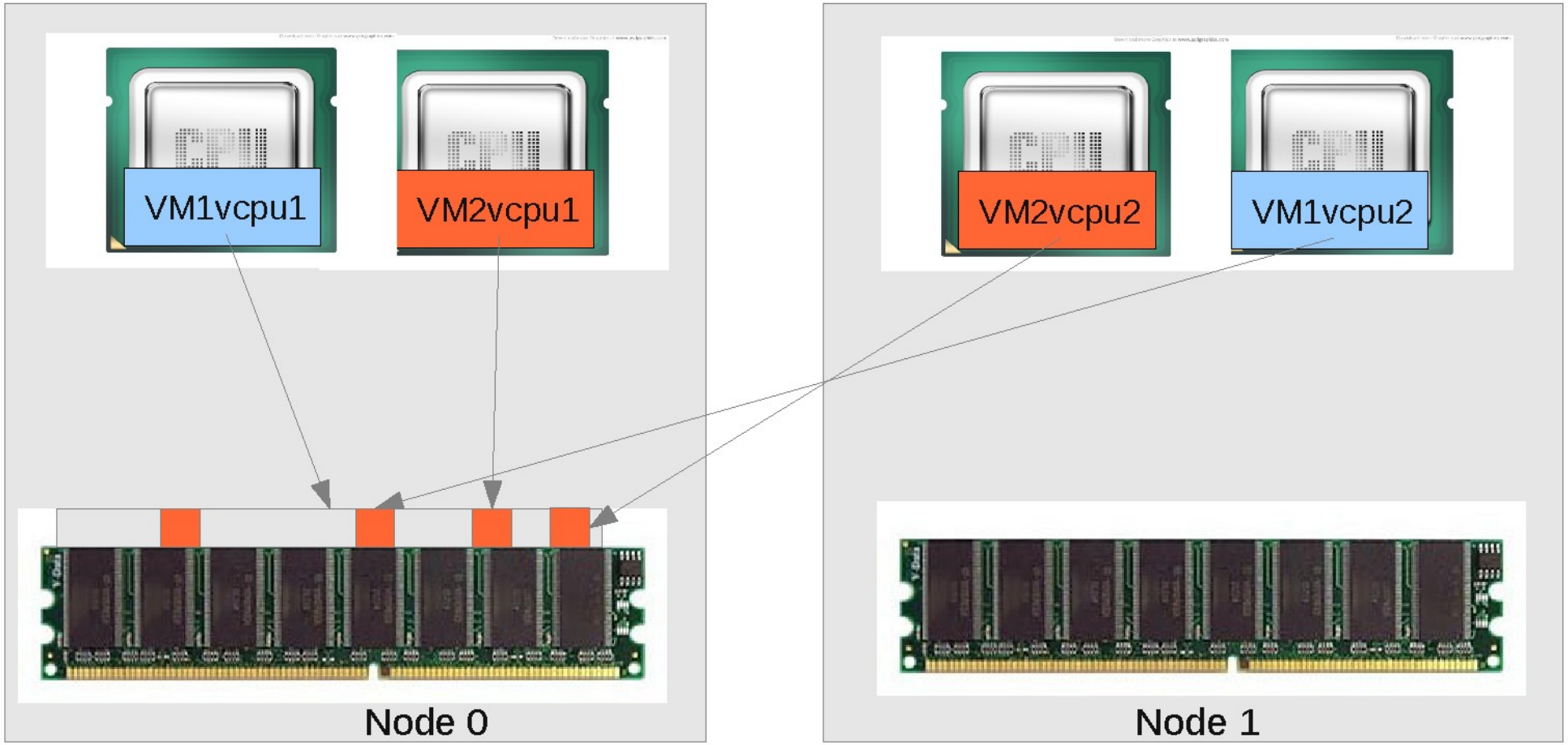
Non Uniform Memory Architecture

- Each CPU has its own memory (fast)
- Other memory accessed via other CPUs (slower)



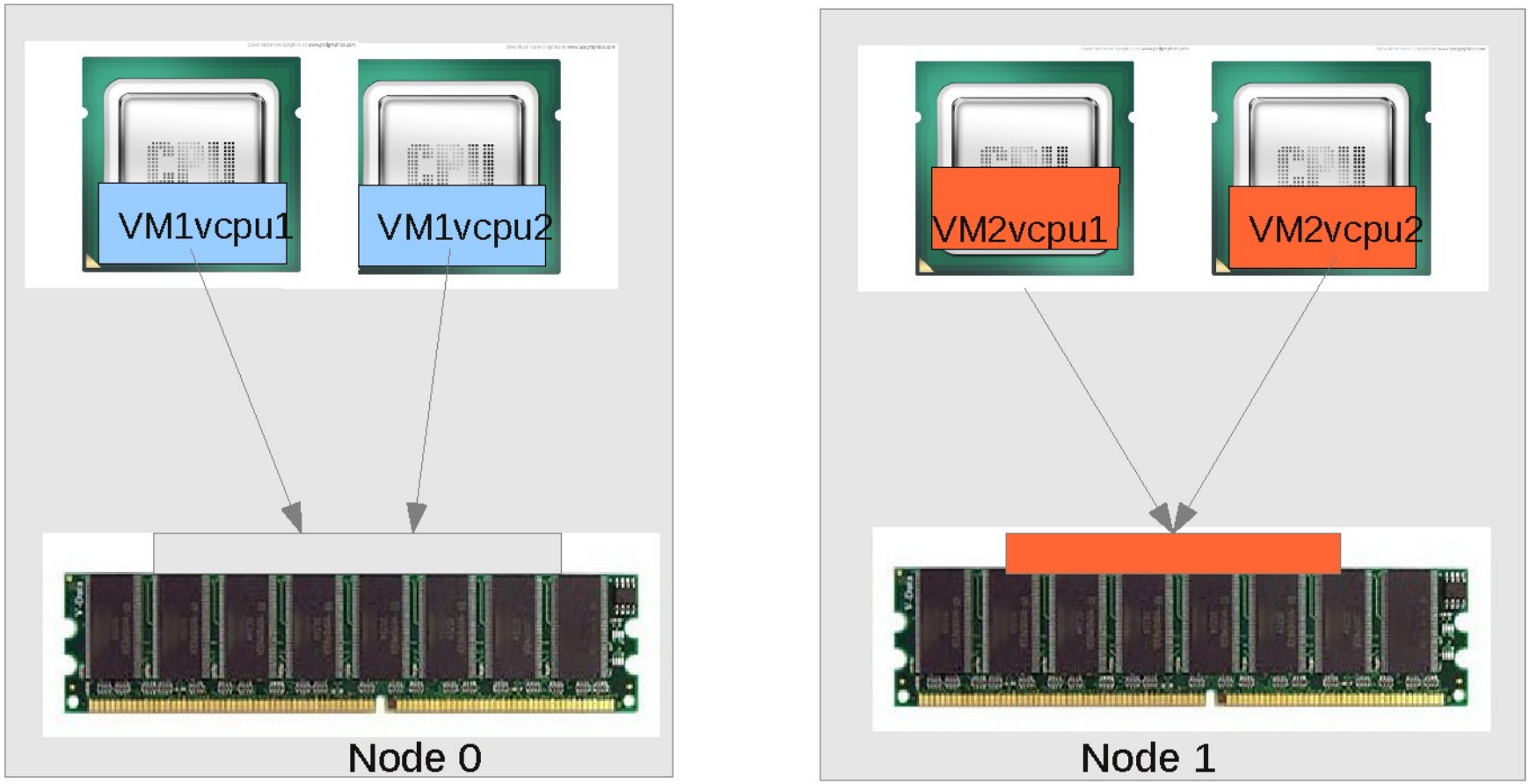
Unlucky NUMA Placement

- Without NUMA optimizations, this can happen



Optimized NUMA placement

- With numad or numa/core
- 3-15% performance improvement typical



Conclusions

- KVM uses QEMU for non-performance critical things
  - KVM developers part of QEMU community
- KVM uses Linux as a hypervisor
  - KVM developers part of Linux community
  - KVM gets Linux hardware support
  - KVM gets Linux performance improvements
- Intel and AMD help solve some of the issues
  - Great performance
  - Still a lot done in software
- KVM has some of the best hardware support and performance
- Everything tested by a large community
  - Stable software

Questions?