RED HAT :: CHICAGO :: 2009
SUMMIT

## FOLLOW US:

TWITTER.COM/REDHATSUMMIT

## TWEET ABOUT US:

ADD #SUMMIT AND/OR #JBOSSWORLD TO THE END OF YOUR EVENT-RELATED TWEET


# Open Source Virtualization: KVM and Linux

Chris Wright
Principal Software Engineer, Red Hat
September 4, 2009

presented by


redhat
Virtualibation


# Increased capacity

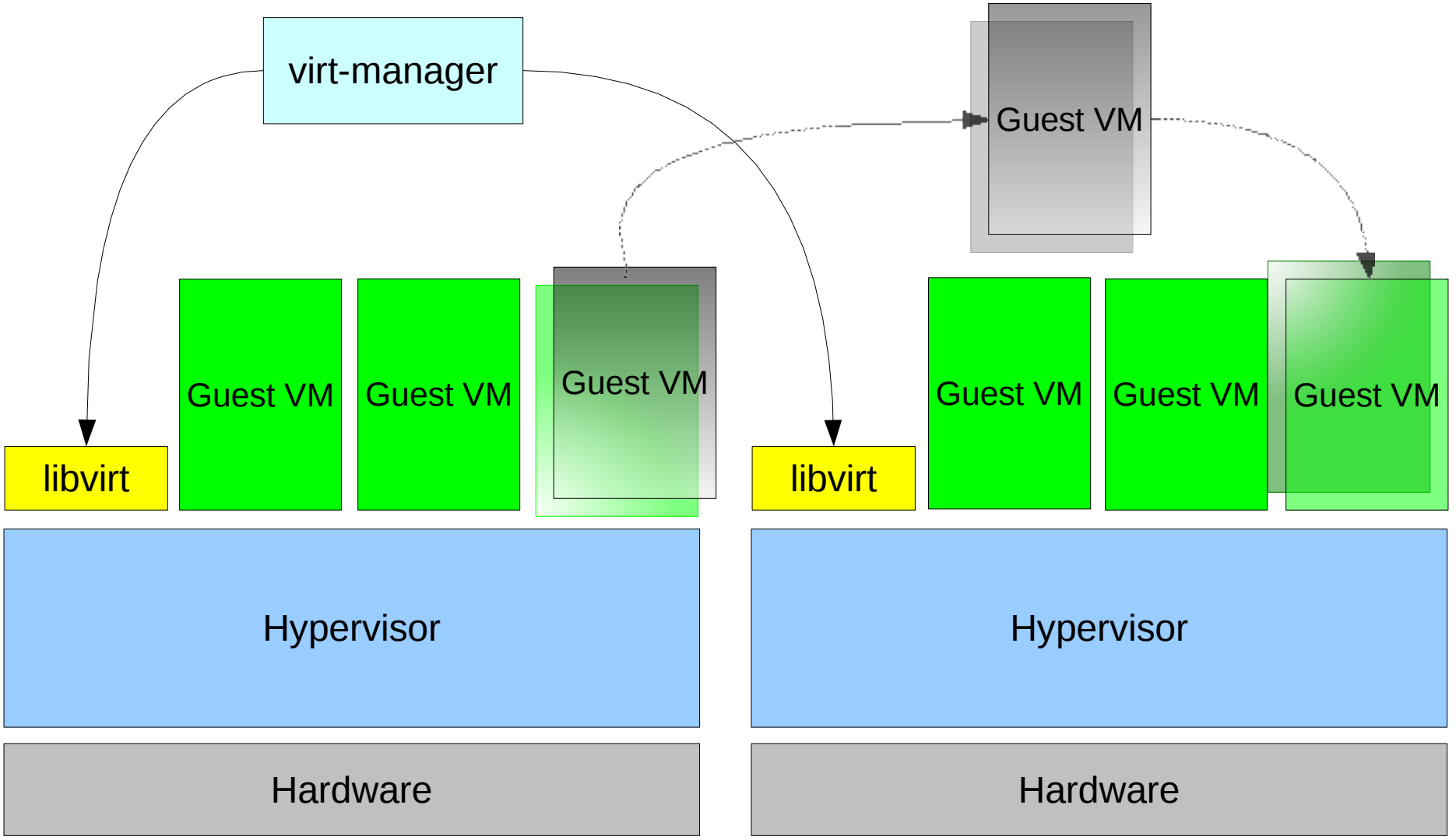
# Agenda

- RHEL 5.4 Virtualization
- Virtualization Stack
- KVM
  - KVM at a glance
  - KVM Execution loop
  - Memory management
  - Paravirtualization
  - I/O
  - Roadmap
  - Community
- Conclusions

# RHEL 5.4 Virtualization

- Xen enhancements
  - 32-on-64 PV, HVM guest timekeeping, 192 CPUs, VPID, 1GB pages, PCI device assignment, SR-IOV
- libvirt update to 0.6.3
  - KVM support, PCI hotplug, PCI device assignment
- Introduction of KVM
  - more on this later
- RHEV
  - Based on RHEL 5.4 and KVM (SVVP, WHQL)
  - Dedicated Virtual Machine hosting platform
  - Graphical Management

# Virtualization stack



# libvirt features

- Provisioning, lifecycle management
  - Hypervisor agnositc
    - Xen, KVM, QEMU, LXC, UML, OpenVZ, VMware, IBM Power
  - 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

# KVM features

Leverage HW virtualization support
    VT-x/AMD-V, VPID/ASID, EPT/NPT, VT-d/IOMMU

CPU and memory overcommit

High performance I/O

Hotplug (CPU, Block, NIC, PCI)

SMP guests

Live migration

PCI Device Assignment and SR-IOV

Page sharing

NUMA

Power Management

SPICE

# The Kernel-based Virtual Machine (KVM)

# KVM at a glance

A Linux kernel module turning Linux into a hypervisor
- Tightly integrated into Linux

Advanced memory management

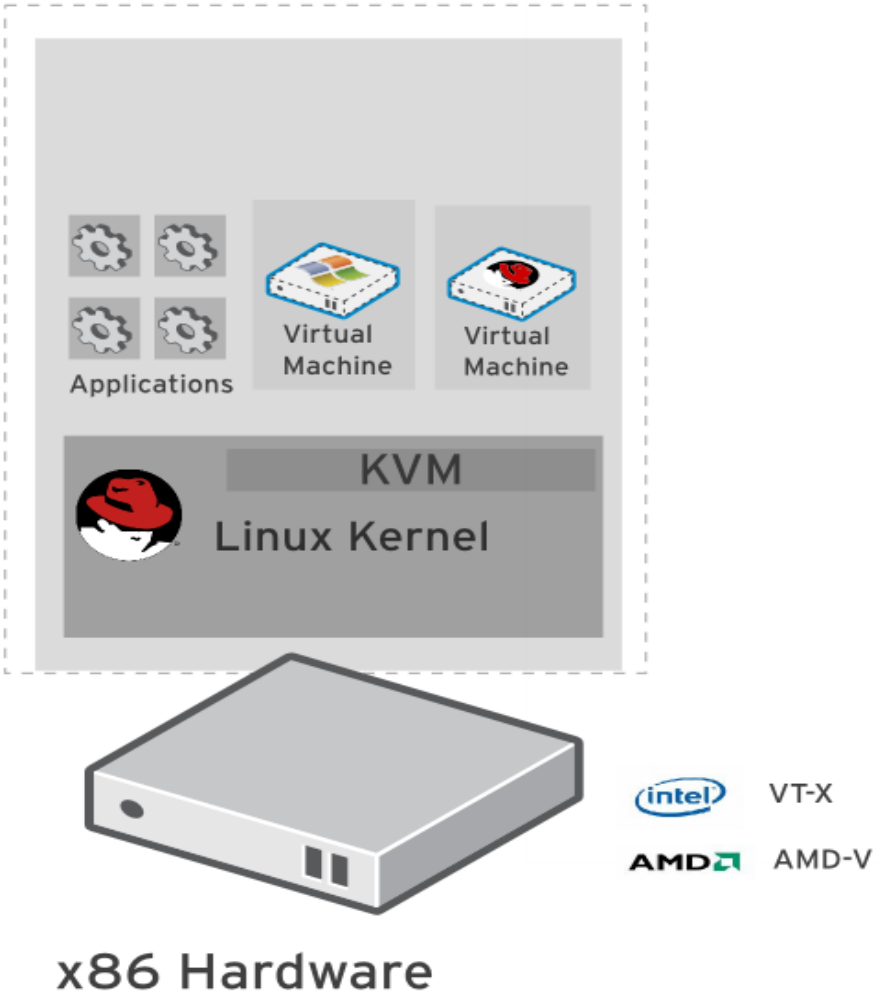
Supports multiple architectures
- x86, ia64, s390, PowerPC

Requires hardware virtualization extensions
- paravirtualization where makes sense

Supports many guests
- RHEL 3/4/5, Windows XP/Server 2003/Server 2008

Competitive performance and feature set

# 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 guest OSes
- KVM architecture provides high "feature-velocity" – leverages the power of Linux

# The KVM approach

is best characterized by

**PRAGMATISM**

# The KVM approach

is best characterized by

~~PRAGMATISM~~

**LAZINESS**

# The KVM approach

A hypervisor needs

- A scheduler and memory management
- Platform support code
- An I/O stack
- Device drivers
- A management stack

# The KVM approach

A hypervisor needs

- A scheduler and memory management
- Platform support code
- An I/O stack
- Device drivers
- A management stack

Linux has world-class support for all this, so why reinvent the wheel?

Focus on virtualization, leave other things to respective developers

# 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

# Leverage Linux

Power Management – a good example of how Linux integration helps

- An especially rough area in operating systems

KVM has

- Automatic frequency scaling
  - with several governors
- Suspend/resume support
  - with running virtual machines
- Advanced I/O support
  - LVM, Multipath, Bonding

All with a small amount of glue code

# Leverage Linux: Security

KVM inherits security features of Linux

Includes support for SELinux

- Provides protection and isolation for virtual machines and host
- Compromised virtual machine cannot access other VMs or host

sVirt Project

- Sub-project of NSA's SELinux community
- Provides “hardened” hypervisors
- Multilevel security
- Isolate guests
- Contain any hypervisor breaches

# 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

# 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

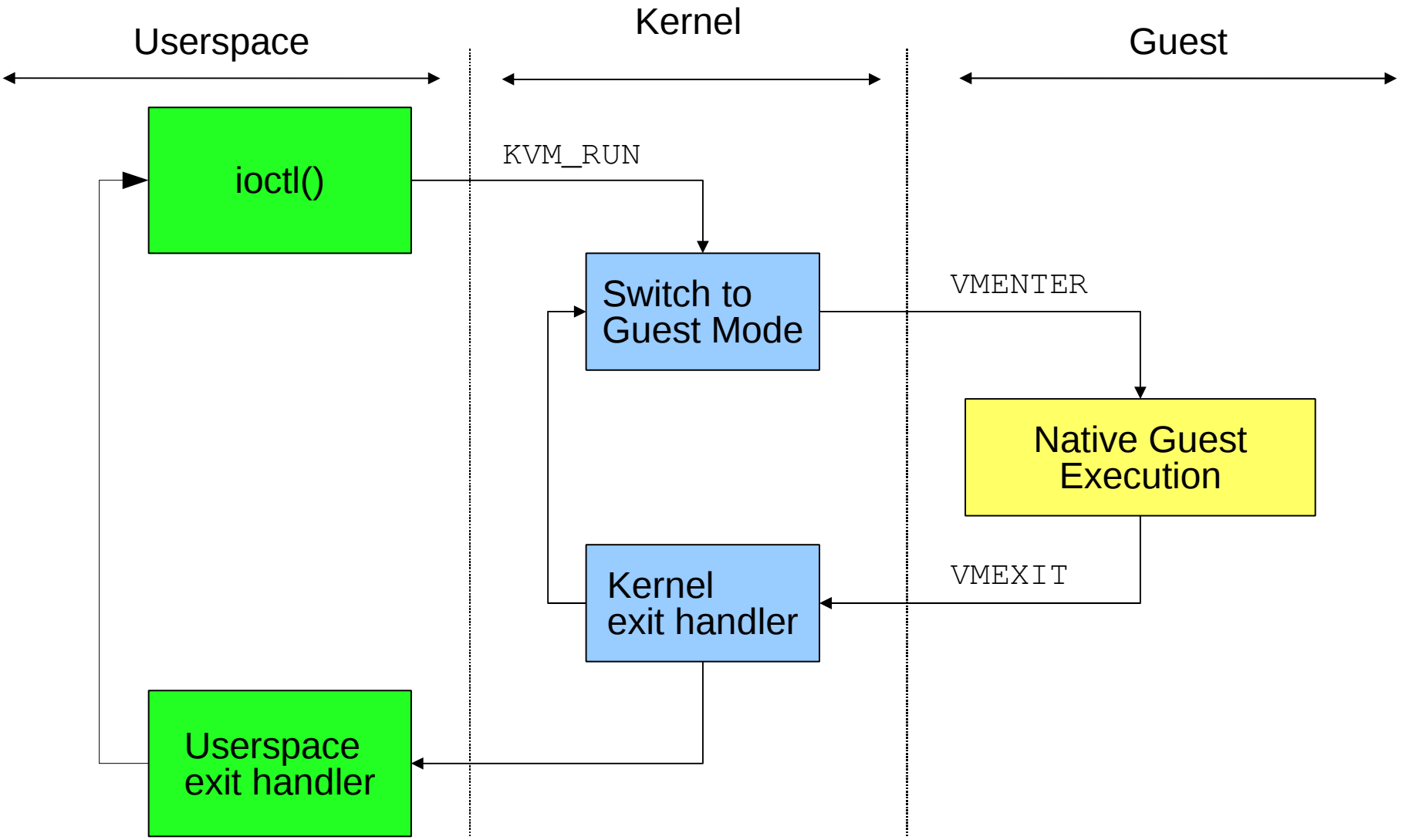
# KVM Execution Model

Guest code executes natively
- Apart from trap and emulate instructions

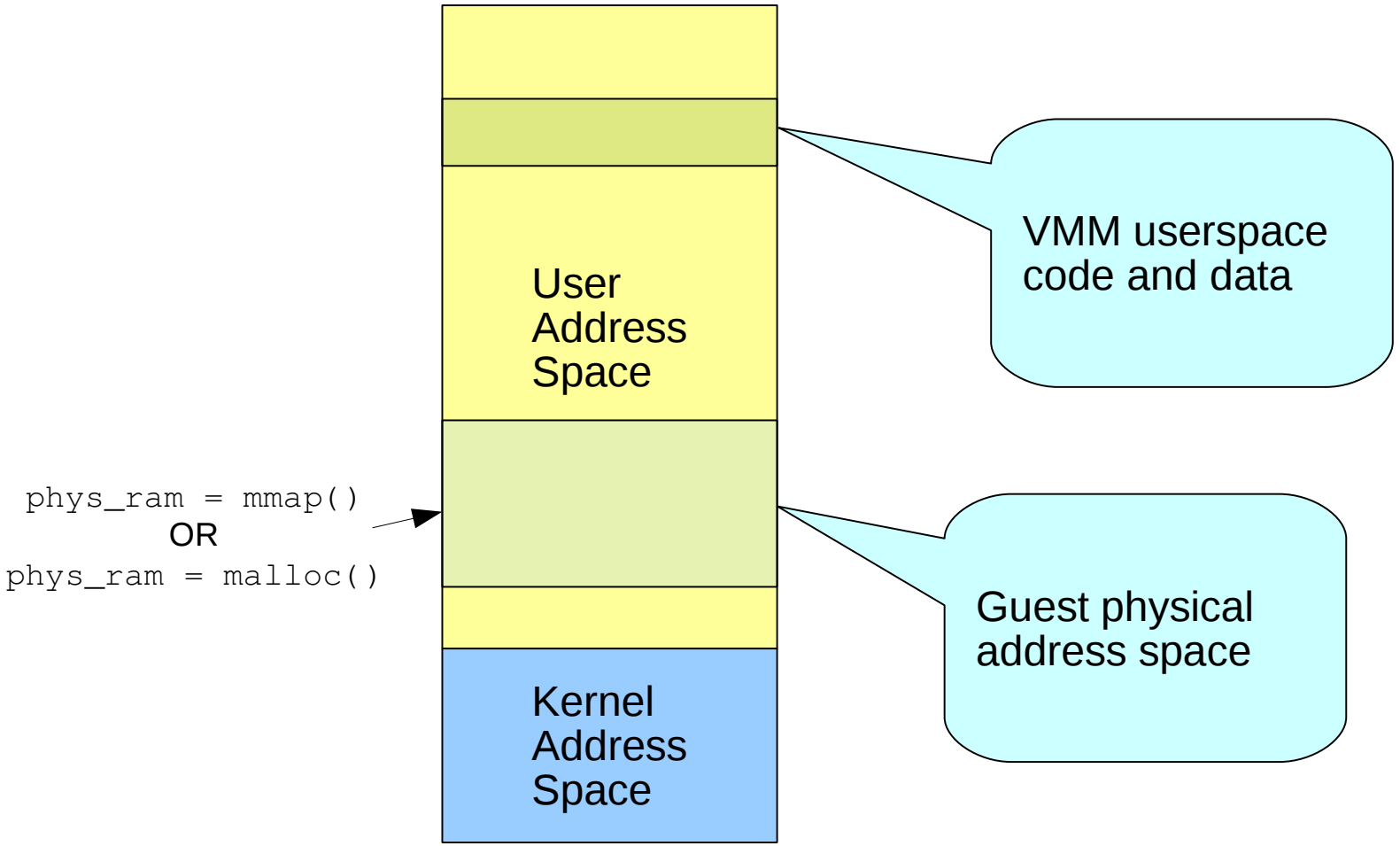
Performance critical or security critical operations handled in kernel
- Mode transitions
- Shadow MMU

I/O emulation and management handled in userspace
- QEMU-derived code base
- Other users welcome

# KVM Execution Model



# KVM Memory Model



# 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

# Memory Page Sharing

Implemented in loadable kernel module

- Kernel SamePage Merging (KSM)

Kernel scans memory of virtual machines

- Looks for identical pages
- “Merges” identical pages
- Only stores one copy (read only) of shared memory
- If a guest changes the page it gets it's own private copy

Significant hardware savings

- Better consolidation ratio
  - Allows more virtual machines to run per host

# 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

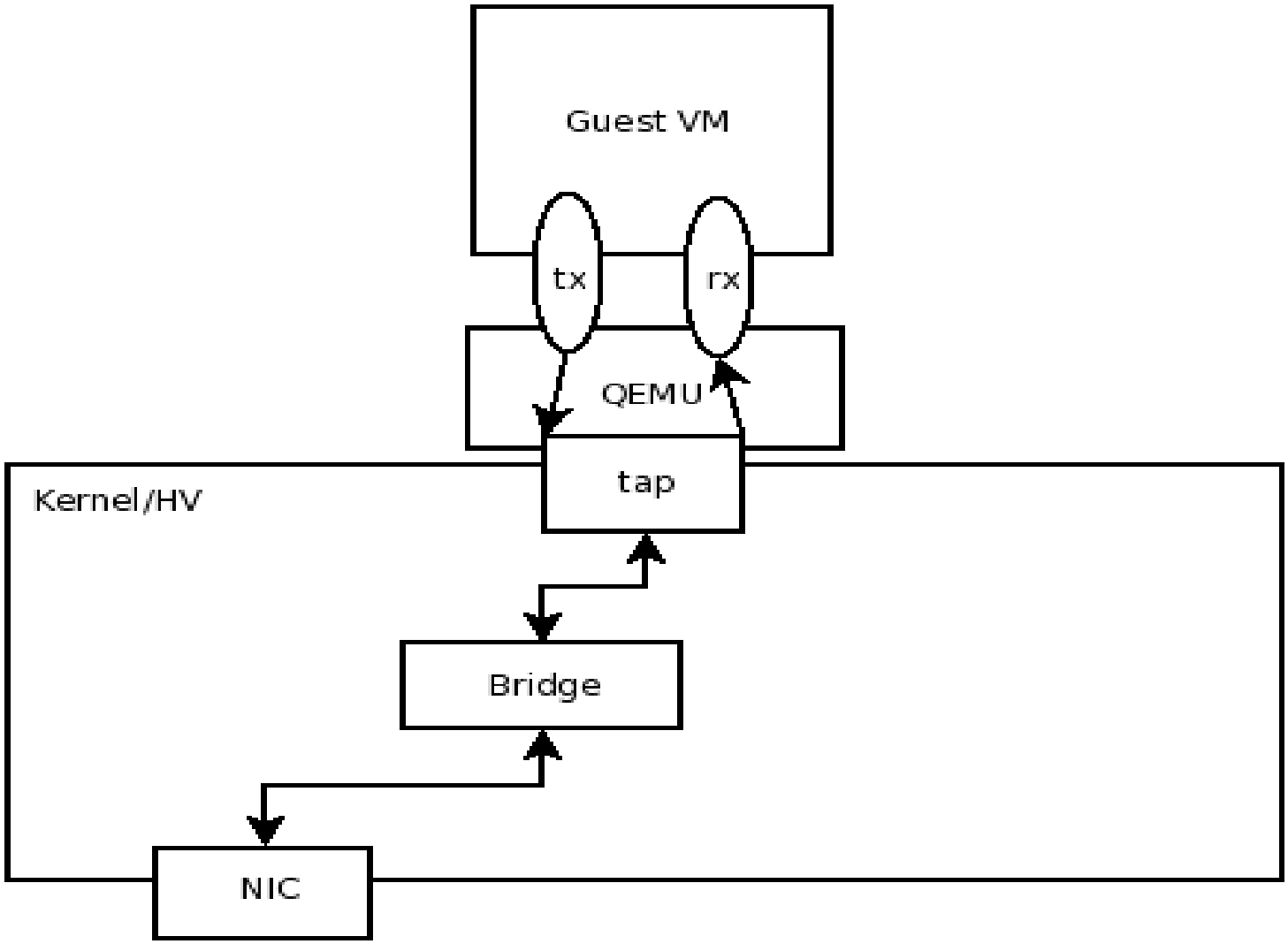
Supported areas

- Hypercall-based, batched mmu operations
- Clock
- I/O path (virtio)

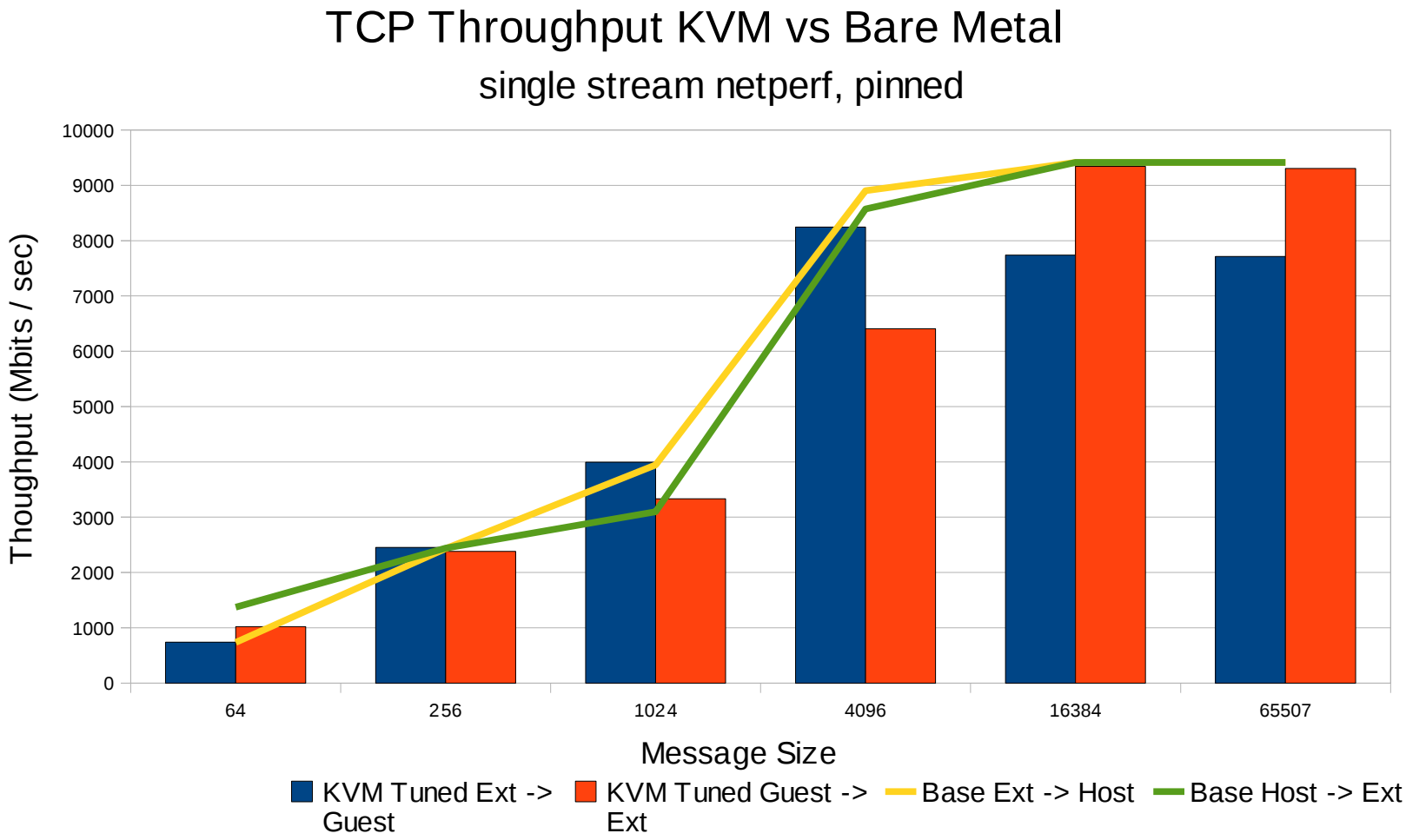
# I/O – virtio

- 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.)
    - RHEL 3/4/5, Windows XP/Server 2003/Server 2008
  - Hypervisor specific host support

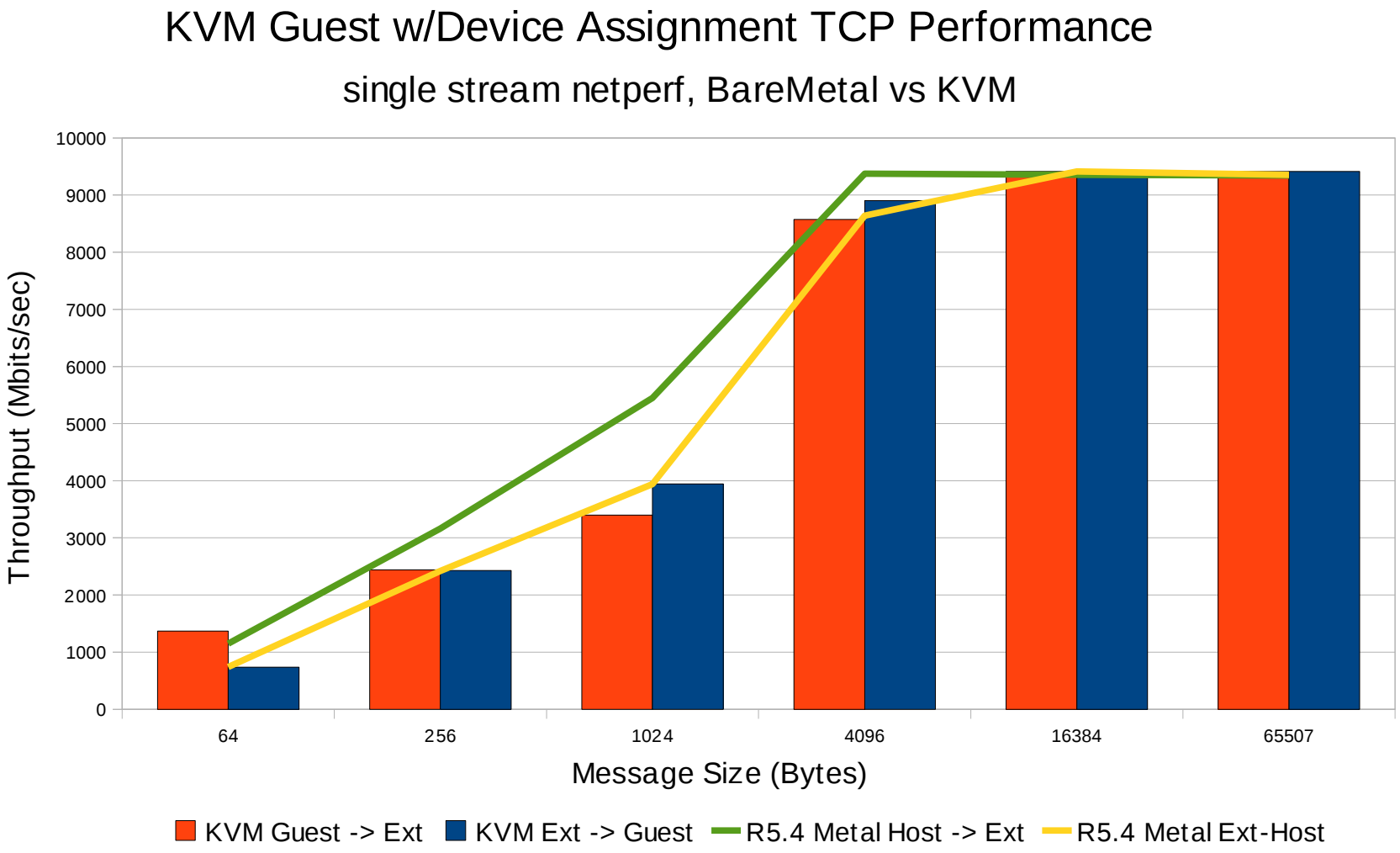
# Virtio network architecture



# KVM netperf w/ virtio



# KVM netperf w/ device assignment



# Roadmap

QEMU improvements and integration

- libmonitor, machine description

qxl/SPICE integration

Scalability work

- Qemu and kvm

Performance...

# Roadmap cont'd

- Performance work
  - Block
    - I/O using linux aio
  - Network
    - GRO, multiqueue virtio, latency reduction, zero copy
  - MMU
    - Page hints
  - Scheduler
    - Improve SMP guest scaling
- Resource management
  - cgroups

# Community

Main contributors

- AMD, IBM, Intel, Novell, SGI, Siemens, Red Hat

Typical open source project

- Mailing lists, IRC

Your participation is most welcome!

http://linux-kvm.org

# Conclusions

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

http://linux-kvm.org


redhat
Virtualibation


redhat