Using Linux as Hypervisor with KVM

Qumranet Inc.

Andrea Arcangeli
andrea@qumranet.com
(some slides from Avi Kivity)
CERN - Geneve
15 Sep 2008

Agenda

- Overview/feature list
- KVM design vs other virtualization designs
- Shadow pagetables in vmx/svm
- Integration with Linux kernel VM
- QCOW2 image format
- Paravirtualization
- Pci-passthrough

KVM Overview

- KVM is a Linux kernel module that turns Linux into a hypervisor
- Requires hardware virtualization extensions

  egrep 'vmx|svm' /proc/cpuinfo
- Supports multiple architectures: x86 (32- and 64-bit) s390 (mainframes), PowerPC, ia64 (Itanium)
- Competitive performance and feature set
- Advanced memory management (full swapping)
- Supports nested full virtualization on SVM (AMD)

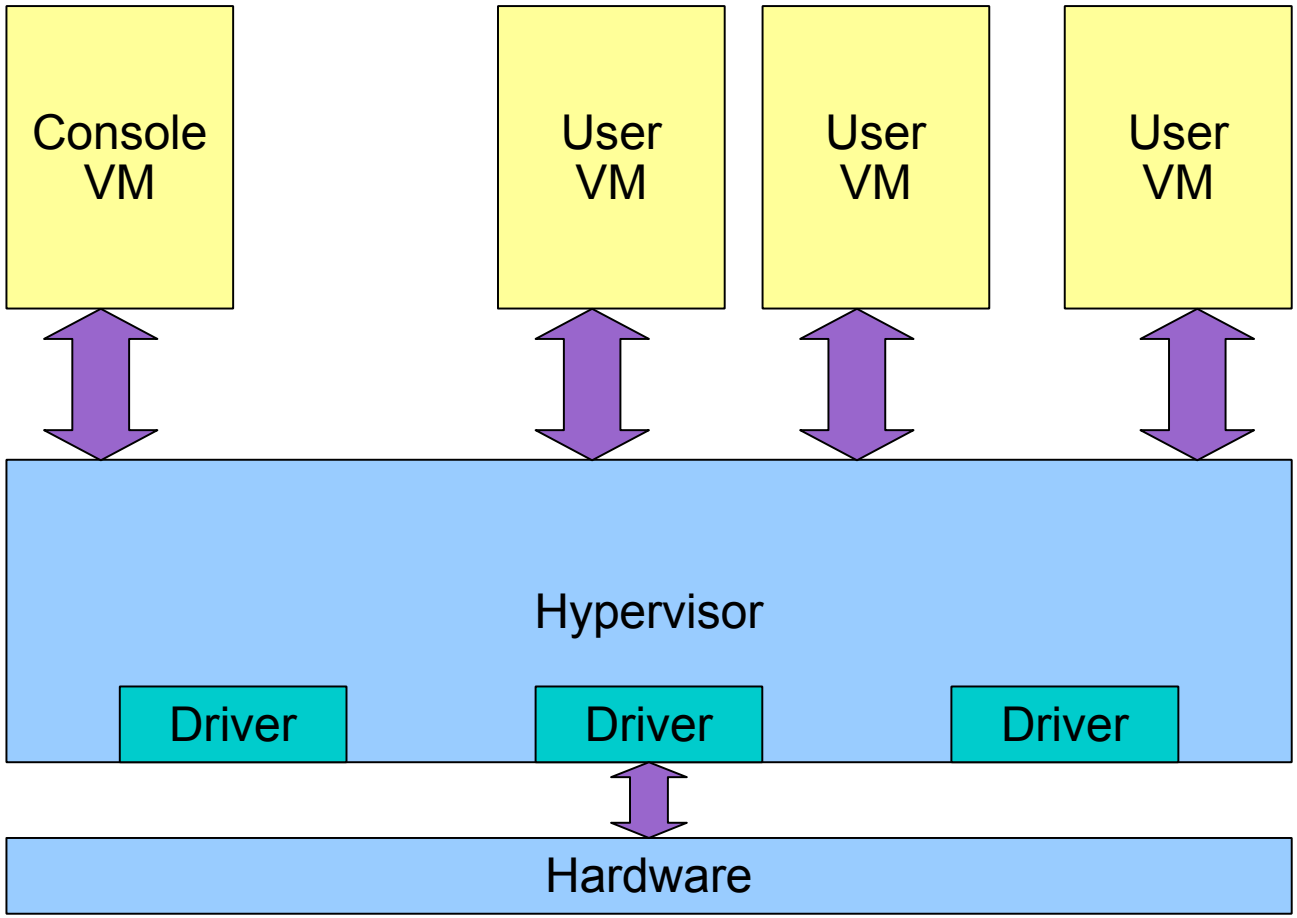
KVM Features

- NPT/EPT support (server boost)
- KSM (share memory with COW)
- Disk image cloning, sharing, snapshot
- Ballooning
- Live migration (nfs as shared storage)
- Save and restore VM
- Virtio paravirtualization
- PCI-passthrough VT-D/IOMMU support

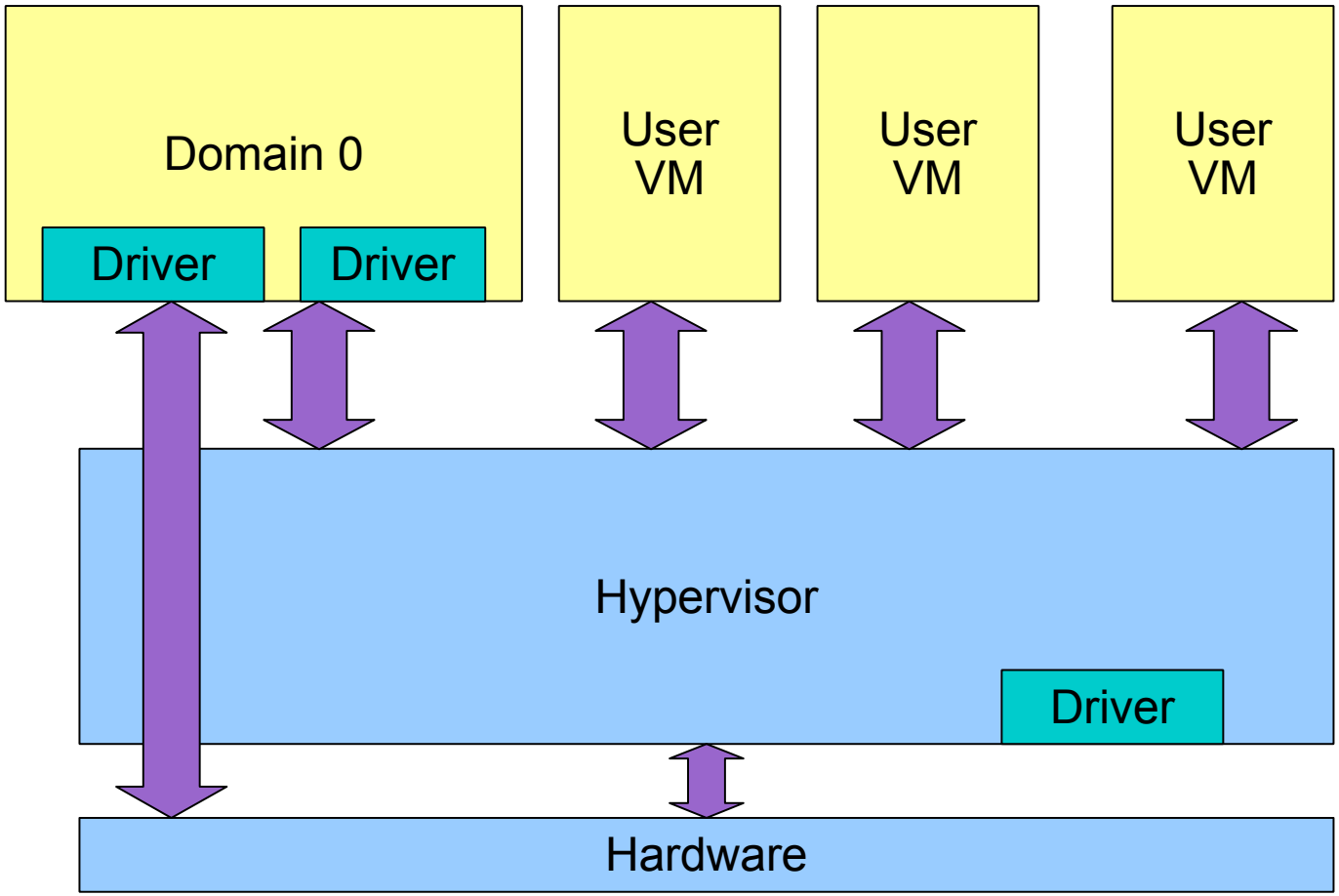
KVM Philosophy

- Reuse Linux code as much as possible
- Focus on virtualization only, leave other things to respective developers
  - VM
  - cpu scheduler
  - Drivers
  - Numa
  - Powermanagement
- Integrate well into existing infrastructure
  - just a kernel module

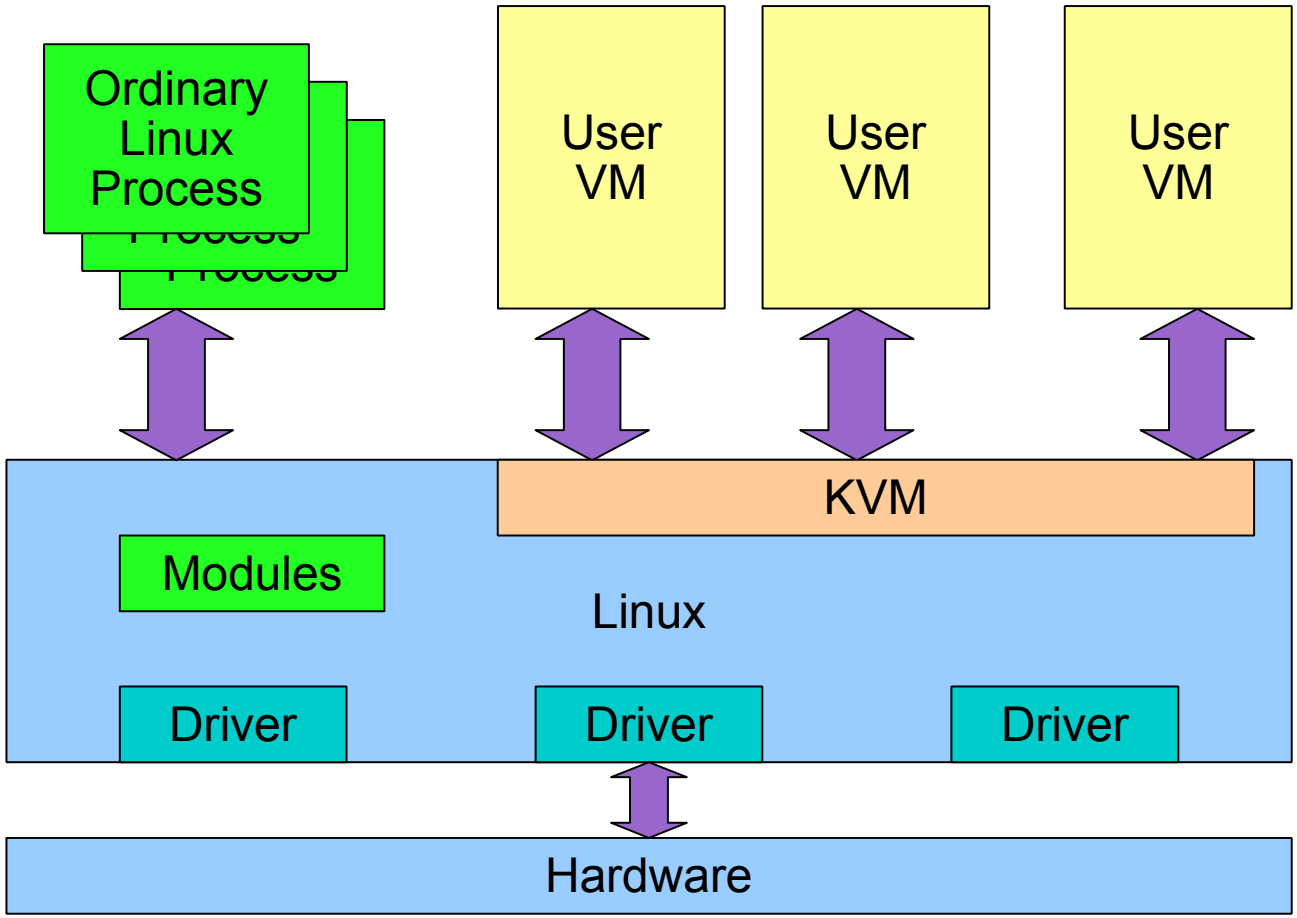
Some closed source proprietary VM design



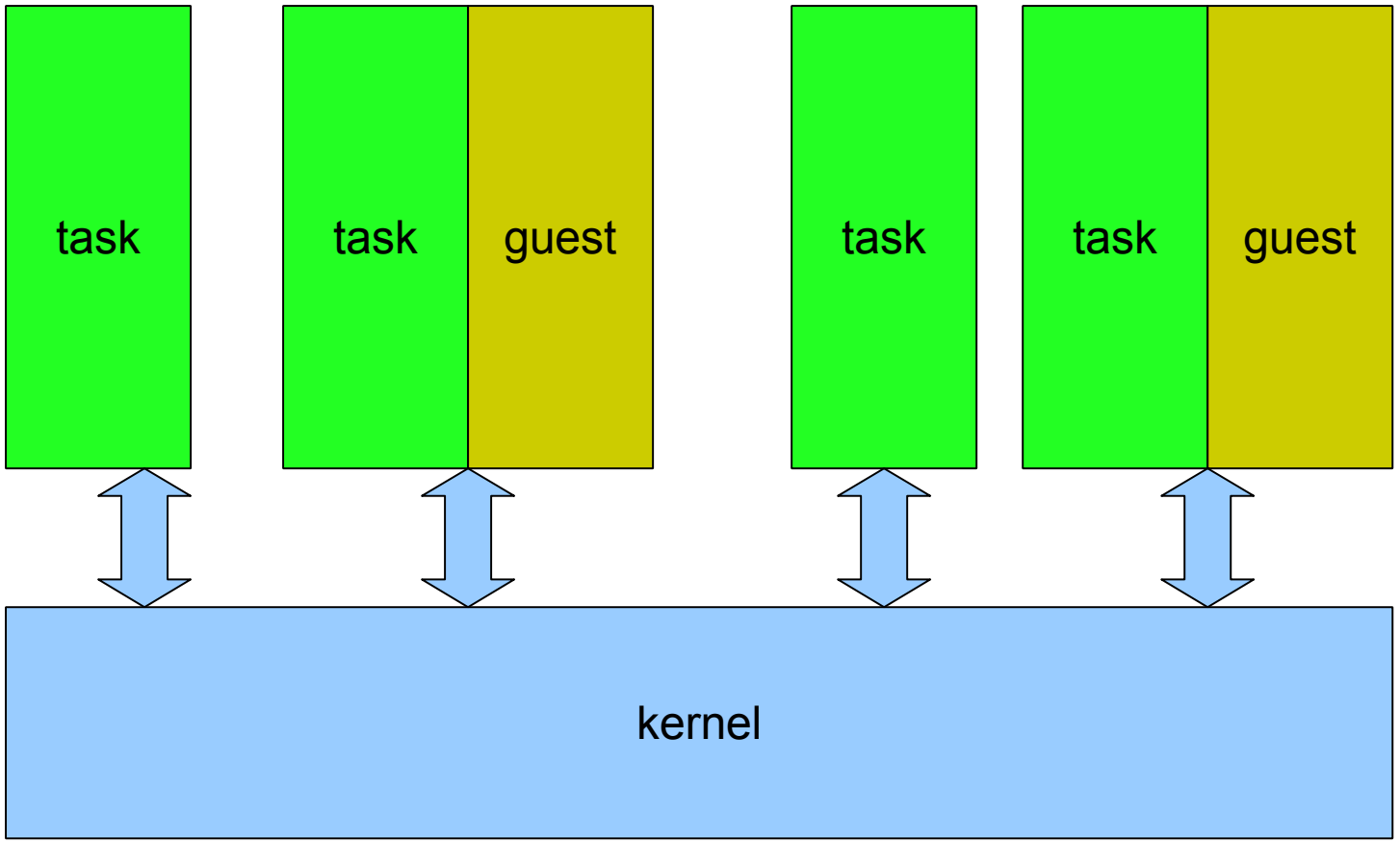
xen design



KVM design... way to go!!



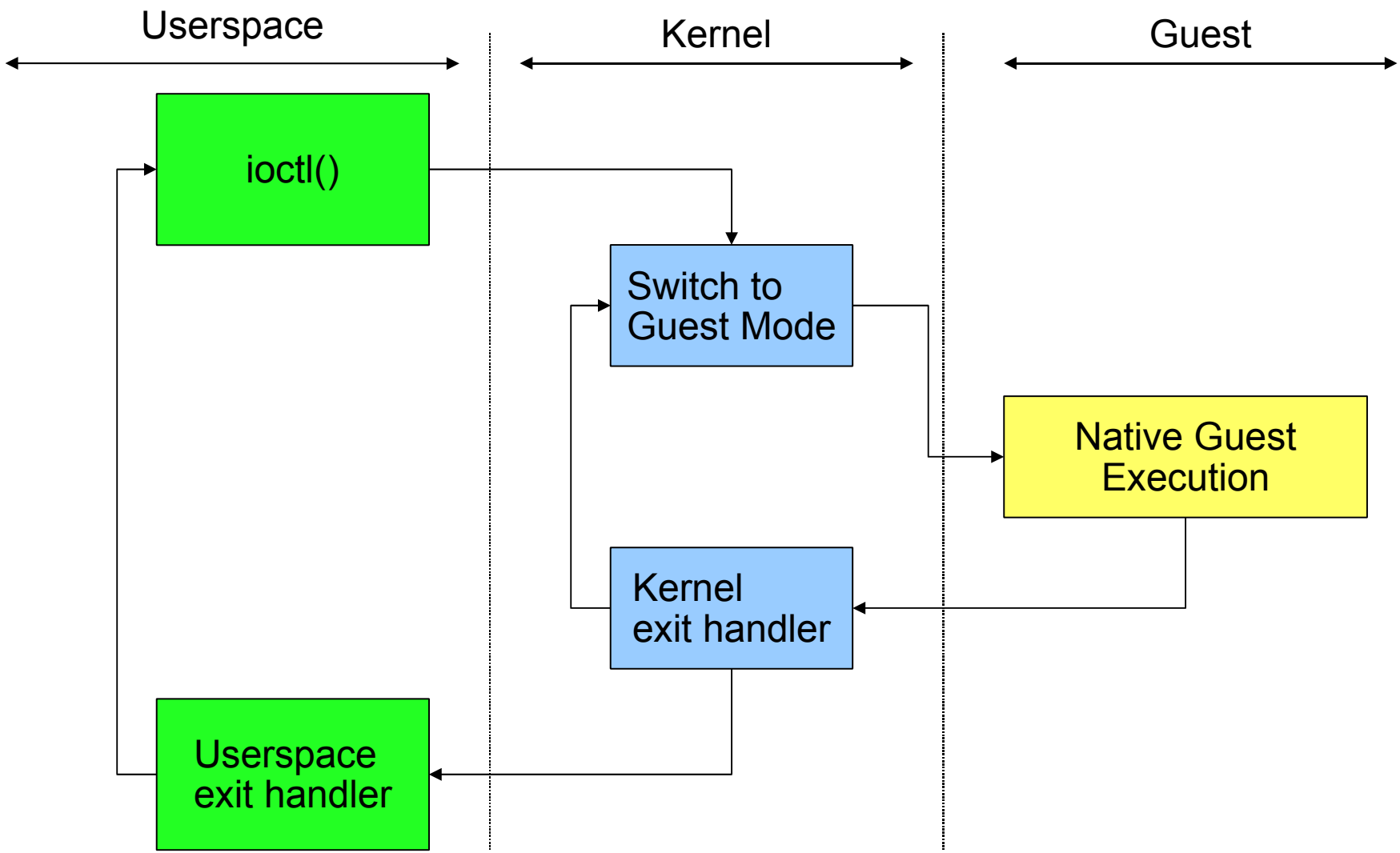
KVM task model



KVM gust mode

- Three modes for thread execution instead of the traditional two:
  - User mode
  - Kernel mode
  - Guest mode (new!)
- A virtual CPU is implemented using a Linux thread (each thread has its own guest mode)
- The Linux scheduler is responsible for scheduling a virtual cpu, as it is a normal thread

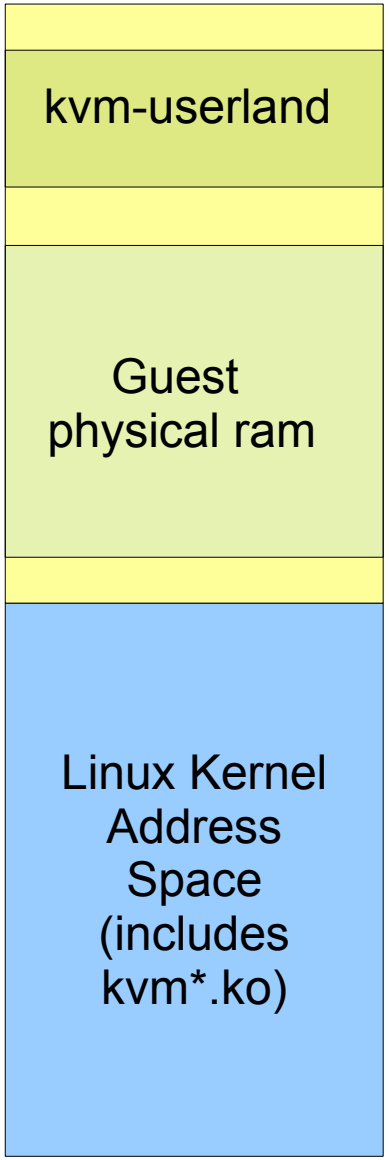
KVM userland <-> KVM kernel



KVM emulation

- Most code executes natively but there is instruction emulation (interpreted) in these cases:
  - wrprotect shadow pagetable faults (will mostly go away with out-of-sync)
  - MMIO on emulated devices
  - Big real mode on vmx (vmx has no real mode support, and vm86 misses the big real mode)
- Emulator is in the kernel to avoid round trip to userland and to support well SMP
- Because of big real mode emulator tries to emulate most instructions

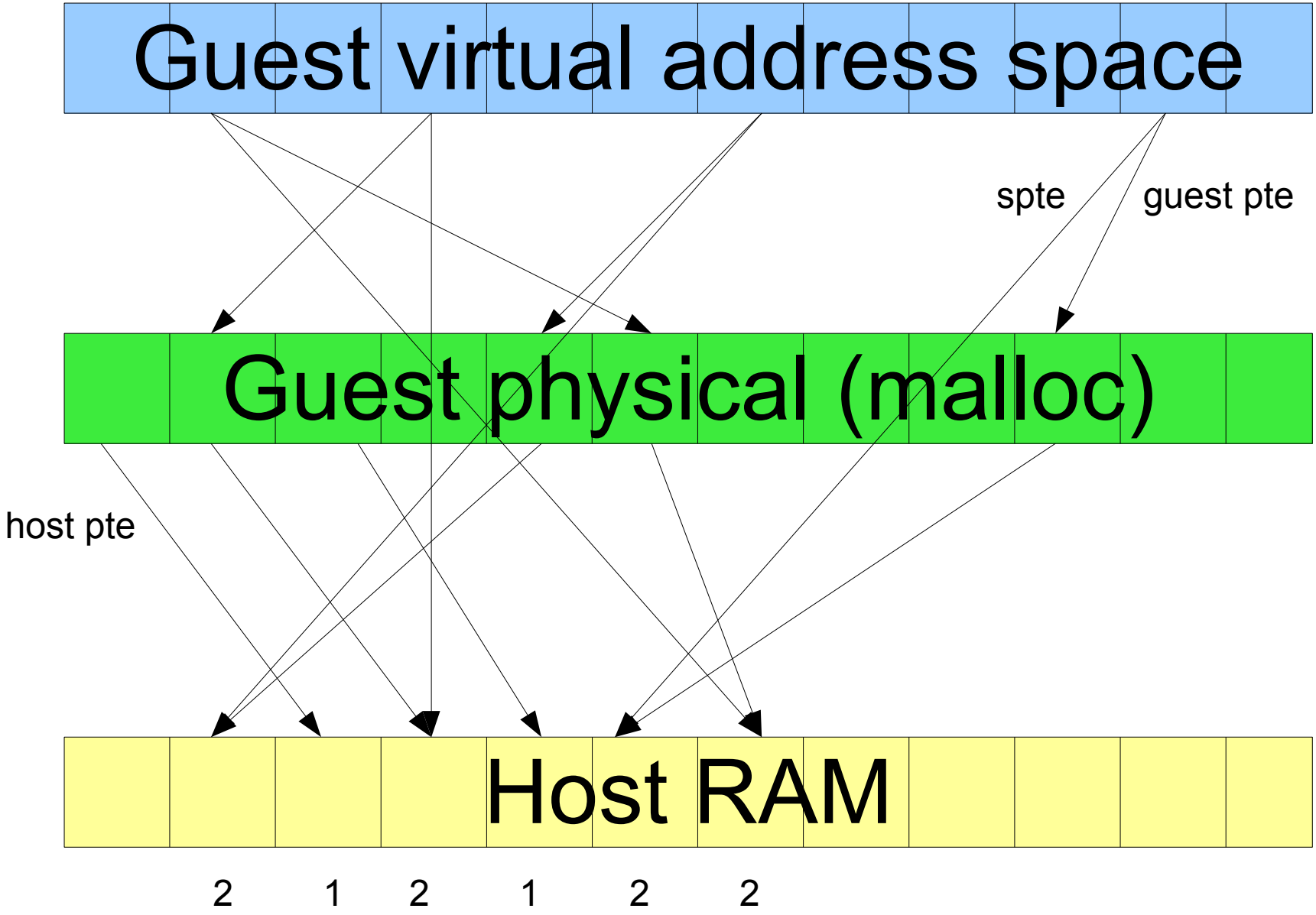
KVM process memory layout



KVM page fault and sptes

- As the CPU enters guest mode, the KVM page fault will be invoked and it'll establish the shadow pagetables
- Shadow pagetables simulates a secondary TLB refilled by software
- The primary TLB maps “host virtual” to “host physical”
- Shadow pagetables map “guest virtual” to “host physical”
- To establish sptes KVM must do “guest virtual” -> “guest physical” -> “host virtual” -> “host physical”

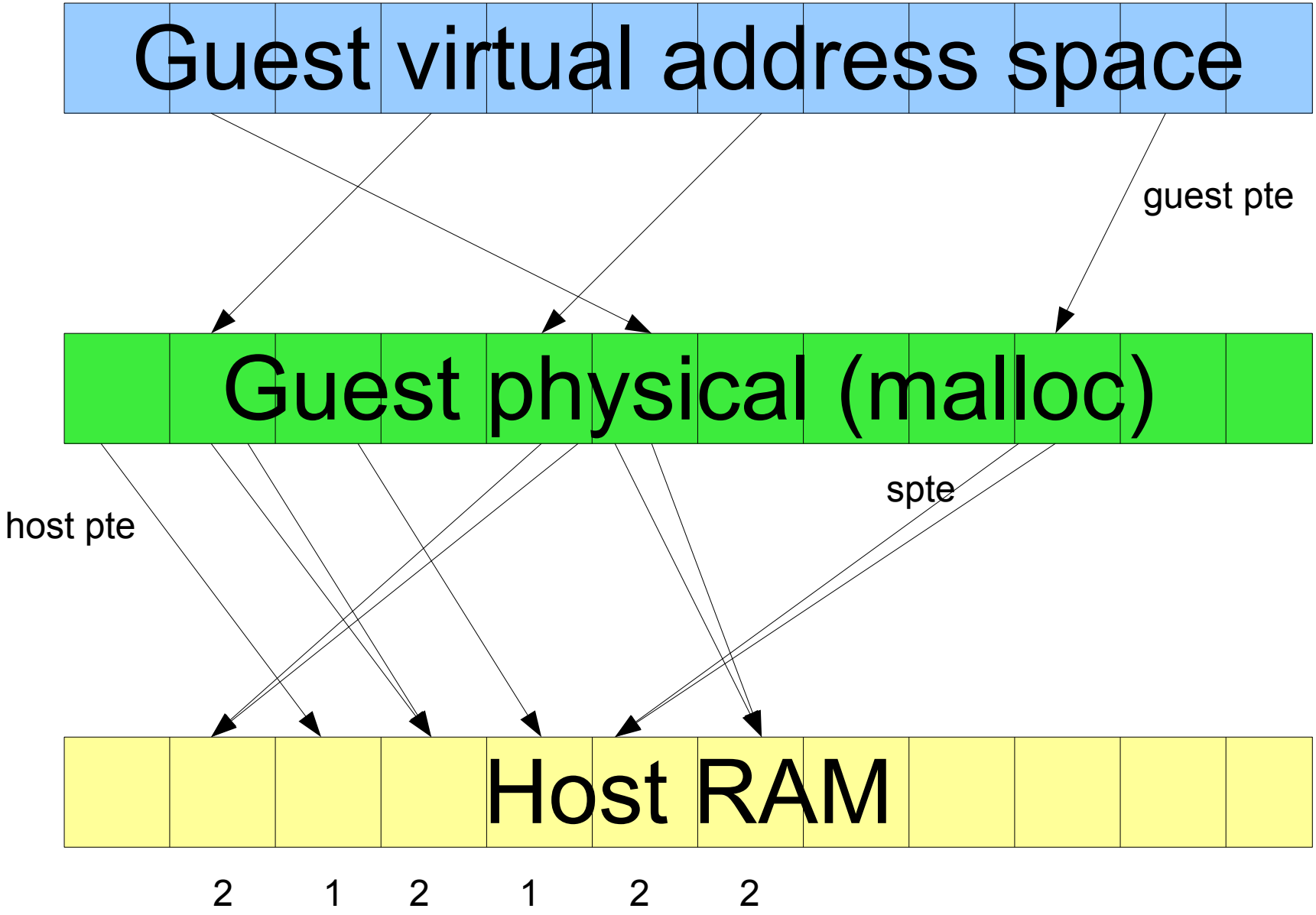
VM layout with sptes



Guest virtual changes

- When guest virtual changes the shadow pagetable must change
- KVM caches shadow pagetables if guest pte wasn't modified:
  - Not like the hardware TLB that would throw away all information when guest issues a TLB flush
- Frequent pte changes in the guest requires many VM-exits and shadow pagetable mangling
- EPT/NPT shadow paging extension avoids all the guest pte mangling virtualization overhead by allowing shadow pagetables to map “guest physical” to “host physical”

VM layout with EPT/NPT sptes



EPT/NTP and computing

- There are two hardware TLB:
  - primary for host: refilled by host ptes
  - secondary for guest: refilled by shadow ptes
- When a guest TLB miss occurs:
  - regular shadow pagetables maps the “guest virtual” to “host physical” and secondary TLB is filled by hardware in 4 memory accesses
  - With EPT/NTP the shadow pagetable maps “host virtual” to “host physical” so harware will have to walk “guest virtual” to “guest physical” first. Each guest pte read requires 5 reads and total will be 20 memory accesses for each TLB miss

EPT/NTP runtime switch?

- Regular shadow paging might be faster for number crunching with a thread per-cpu and not much guest VM activity
- EPT/NPT will surely be much faster for databases or similar apps as it eliminates lots of VM exits
  - NOTE: I/O activity is not relevant with regard to shadow paging
- Benchmarks are needed
- It's also possible for KVM to autodetect the workload and switch from regular shadow paging to NPT/EPT at runtime automatically

Linux Kernel integration

- Preempt notifiers:
  - CPU doesn't fully exit guest mode if scheduler invocation doesn't switch the task in the CPU
- MMU notifier:
  - Makes the guest physical ram totally swappable
  - Provides transparent aging and unmapping methods to remove secondary MMU references
  - Generic infrastructure: fits all kind of secondary MMUs, not just the KVM usage
  - Multiple secondary MMUs can work on the same “mm” simultaneously without interference

MMU notifier

- Linux doesn't know anything about the KVM MMU
- But the core Linux VM needs to:
  - Flush shadow page table entries when it swaps out a page (or migrates it, or inflates the balloon...)
  - Query the pte accessed bit to determine the age of a page to decide if it's part of the working set
- Every time Linux changes the primary MMU it notifies the secondary MMU drivers
- KVM ensures all relevant shadow pagetables are zapped before the MMU notifier method returns

QCOW2 format

- Divides the logical volume size in clusters
- Appends newly written blocks to the qcow2 image
  - Raw images also allocate blocks only after writes (holes), but the file size fixed
  - cp/scp/rsync by default won't recreate holes in destination, so small file is more user friendly
- Allows cloning an image with indefinite levels (qcow2 code is recursive)
  - All parent images must be readonly, child is COW
  - Snapshots are qcow2 images created on temporary files (changes be flushed to parent)

KVM Paravirtualization

- Emulated devices are the default with KVM/QEMU but they're slower
- MMIO accesses are emulated and they require an exit all the way down to userland if the driver is running in the I/O thread kvm-userland context
- This can result in dozen of exists for each packet delivered
- Paravirtualization is provided by the linux guest common code with a generic driver infrastructure
- KVM provides the paravirt support so the guest will enable paravirtualization during boot

KVM with Linux virtio

- Timer
  - More robust to get the time from the host with the equivalent of gettimeofday than to emulate PIT/HPET/RTC
- I/O
  - Avoids IDE/SCSI emulation (still has to go to userland for qcow2 etc..)
- Networking
  - A single VM-exit can deliver the packet to the virtio network device, can support GSO to deliver multiple packets in one exit, can run zerocopy

PCI passthrough

- New hardware provides VT-d/IOMMU to prevent PCI devices to DMA anywhere they want in RAM
- VT-d/IOMMU allow to securely associate a PCI device to a VM without risking to destabilize host kernel or other guests
- Without VT-d/IOMMU for pci-passthrough to work (insecurely):
  - the spte mappings must become an identity
  - pvdma must be supported by the guest
- VT-d/IOMMU also requires pvdma support to allow swapping (if guest is malicious the VM will be killed when VT-d throws an async exception)

KVM ideal for cloud computing

- VDE virtual distributed ethernet can be bridged or routed to the real ethernet
- Using tap-fd it's possible to create a p2p encrypted mac-enforced (or routed) secure virtual ethernet
- Qcow2 base image distributed to all clients
- Applications unpacked at boot on top of qcow2 base image, or distributed as child images
- Transparent environment
- Would like to run KVM on end user workstations with CPUShare to create lots of VM on demand and an omogeneous environemnt

Q/A

- You're very welcome!