Using Linux KVM and KSM in scientific computing environments

Red Hat Inc.

Andrea Arcangeli
aarcange at redhat.com

CERN virtualization workshop - Geneve
24 June 2009

Agenda

- RHEV-H/RHEL 5.4 and RHEV-M
- KVM philosophy
- KVM features
- KVM design vs other virtualization designs
- KVM numbers with HINT scientific computing benchmark
- KSM @ CERN & benchmark
- Conclusions
- Ideal virtualiaztion @ CERN

RHEV-H

- Dedicated hypervisor
  - The minimum OS needed to run/manage virtual machines
  - Well defined management interfaces/APIs
- Small footprint
  - 100MB image size
  - 750MB disk space required for installation
  - 512MB run-time RAM footprint

RHEV-H

- Built from software distributed as part of RHEL
  - Builds upon well tested base OS
  - Supports the same hardware as RHEL
  - Leverages hardware certification and partner testing
- Utilizes KVM (Kernel-based Virtual Machine)
- Includes VDSM (or RHEV-M host agent) provides high level API
- RHEL 5.4 onward will support KVM and RHEV-M in addition to RHEV-H

RHEV-H

- RHEV-H image is a LiveOS created using livecd-tools
- Image provided in ISO format (livecd)
- Tools provided to convert image to:
  - PXE images (kernel and initrd)
  - USB Flash Drive
- Boots completely stateless
- Installable to local hard disk or flash drive
- Minimal state information persisted to config partition

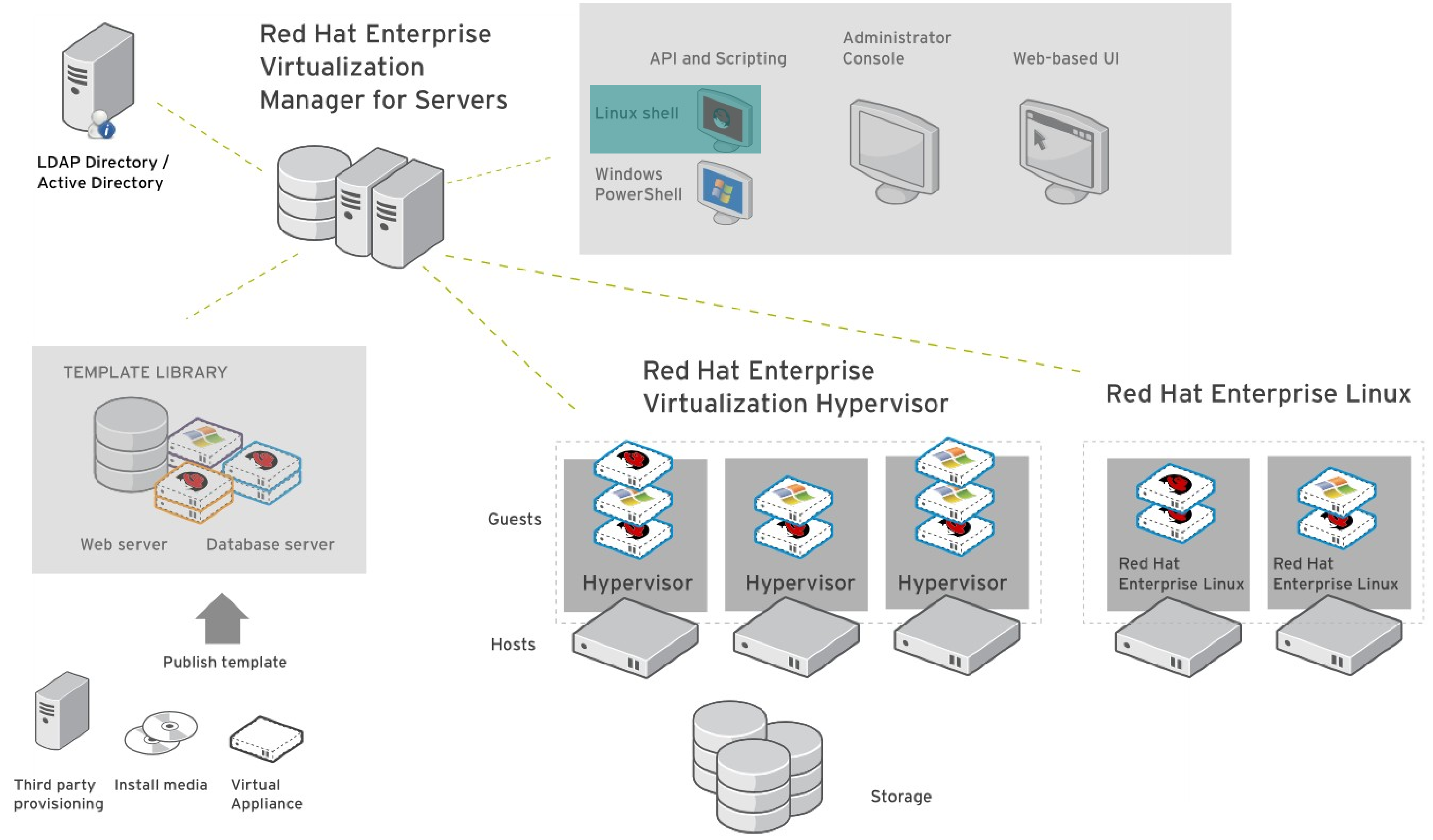
RHEV-M

- **RHEV-M** provides a centralized management server for coordinating groups of RHEV-H Nodes
  - Nodes are grouped in logical clusters which define migration domains
  - Utilize shared storage for guest filesystem
  - VM management done by VDSM daemon
  - **Migrate VM during idle times and shutdown idle nodes to save power**

RHEV-M

- **Physical Resources**
  - Data Center
  - Storage
  - Networking
  - Clusters
  - Hosts
- **Virtual Resources**
  - Virtual Servers
  - Virtual Desktops

RHEV


LDAP Directory / Active Directory
Red Hat Enterprise Virtualization Manager for Servers
API and Scripting
Administrator Console
Web-based UI
Linux shell
Windows PowerShell
TEMPLATE LIBRARY
Web server
Database server
Publish template
Third party provisioning
Install media
Virtual Appliance
Red Hat Enterprise Virtualization Hypervisor
Red Hat Enterprise Linux
Guests
Hypervisor
Hypervisor
Hypervisor
Red Hat Enterprise Linux
Red Hat Enterprise Linux
Hosts
Storage

KVM philosophy

- Reuse Linux code as much as possible
- Focus on virtualization only, leave other things to respective developers
  - VM
  - cpu scheduler, cpu bindings
  - Drivers
  - Numa
  - Powermanagement
  - Cgroups, host partitioning (RHEL6)
- Integrate well into existing infrastructure
  - just a kernel module + mmu notifier

KVM philosophy

- KVM not microkernel approach
- In monolithic Linux we trust anyway
- If Dom0 kernel crashes things go bad even if microkernel-hypervisor memory can't be corrupted by Dom0:
  - Network loss
  - Disk I/O loss
  - Risk of I/O data corruption
- Microkernel approach to virtualization doesn't bring significant reliability advantages and it forces the hypervisor to reinvent inferior wheels

KVM features

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

```
egrep 'vmx|svm' /proc/cpuinfo
```

- Supports multiple architectures: x86 (32- and 64-bit) s390 (mainframes), PowerPC, ia64 (Itanium)
- Advanced memory management (full swapping)
- Nested full virtualization on SVM (AMD)
- Disk image cloning, sharing, snapshot
- Live migration (nfs/SAN [iScsi/FC] as shared storage)

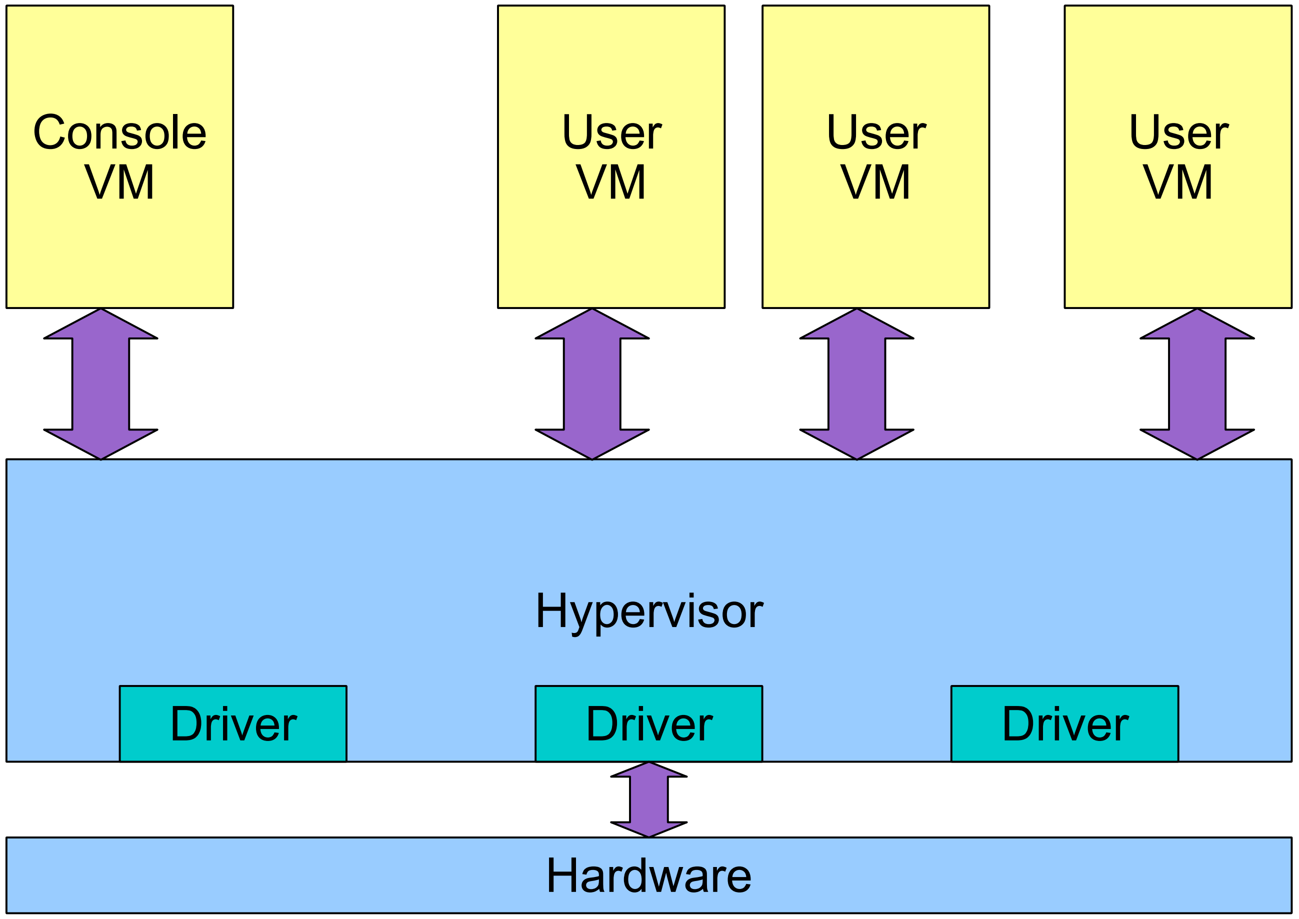
KVM features

- Out of sync shadow ptes (faster on intensive guest pte modifications)
- NPT/EPT support (server/mm-switch boost)
- KSM (shares guest memory creating COW pages)
- Disk image cloning, sharing, snapshot
- Ballooning
- Save and restore VM
- Virtio paravirtualization (net/blk/clock)
- PCI-passthrough VT-D/IOMMU support

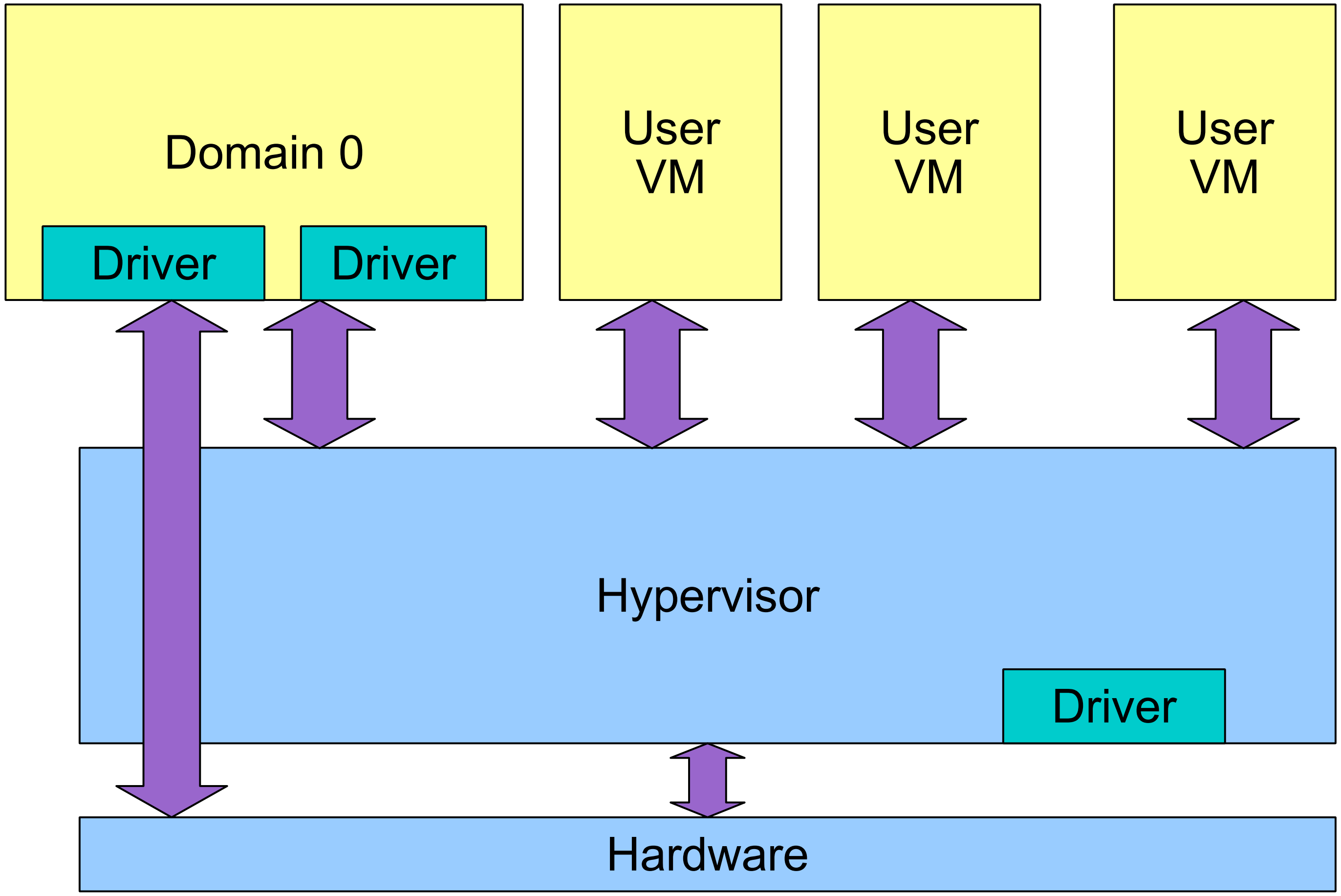
KVM no lockstep

- Lockstep huge slowdown and complexity in SMP, not suitable for computing
- HA shouldn't be provided by the VM in the cloud
- More than one node can must be allowed to go down (like in a department power loss) and grid must notice the job failed and restart the node on a functional part of the grid
- Guest OS/apps should handle HA, the KVM Linux Hypervisor not
- Grid needs open source, maximum flexibility and simplicity to manage or customize, and maximum speed at computing (not HA)

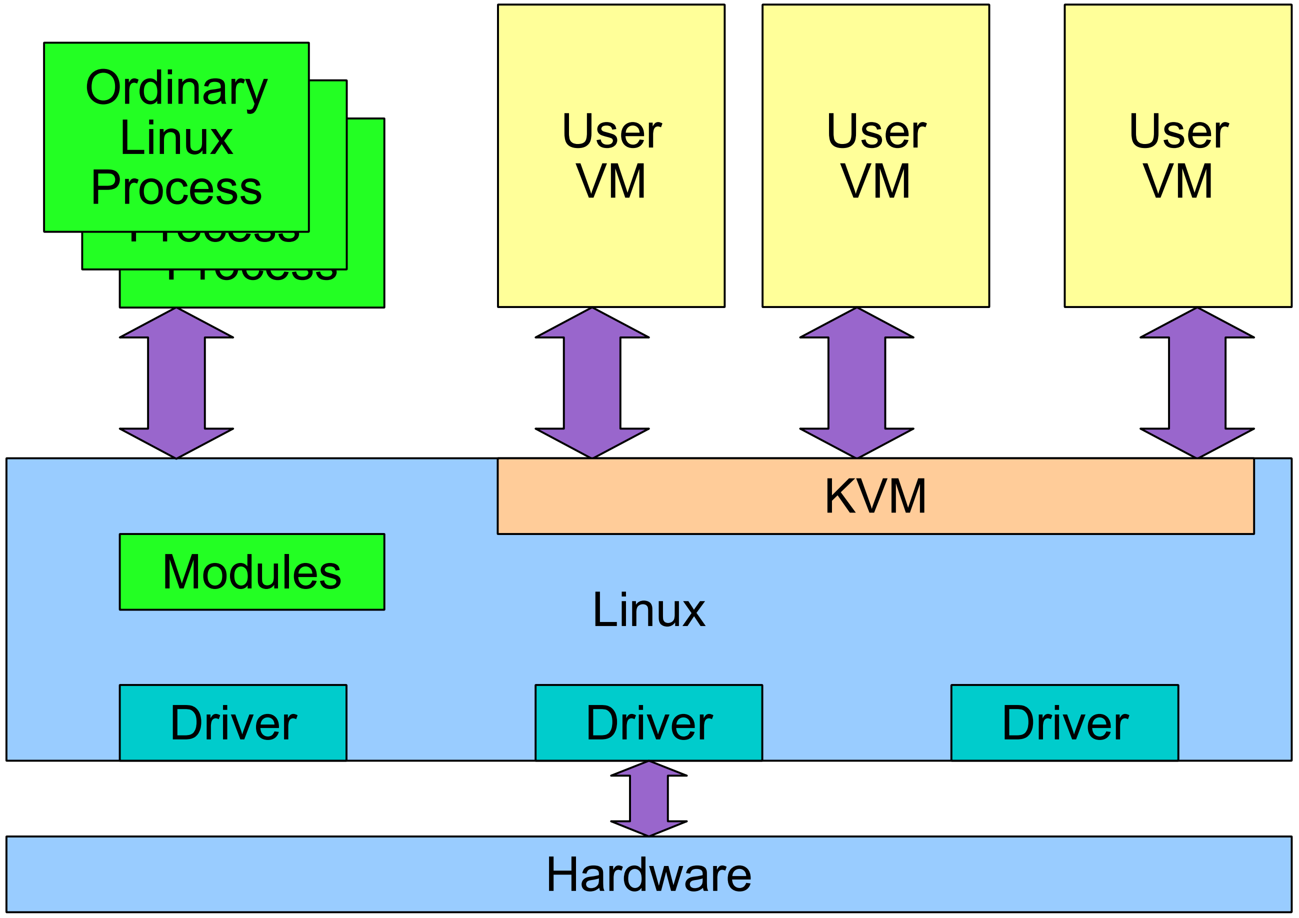
Some proprietary VM design



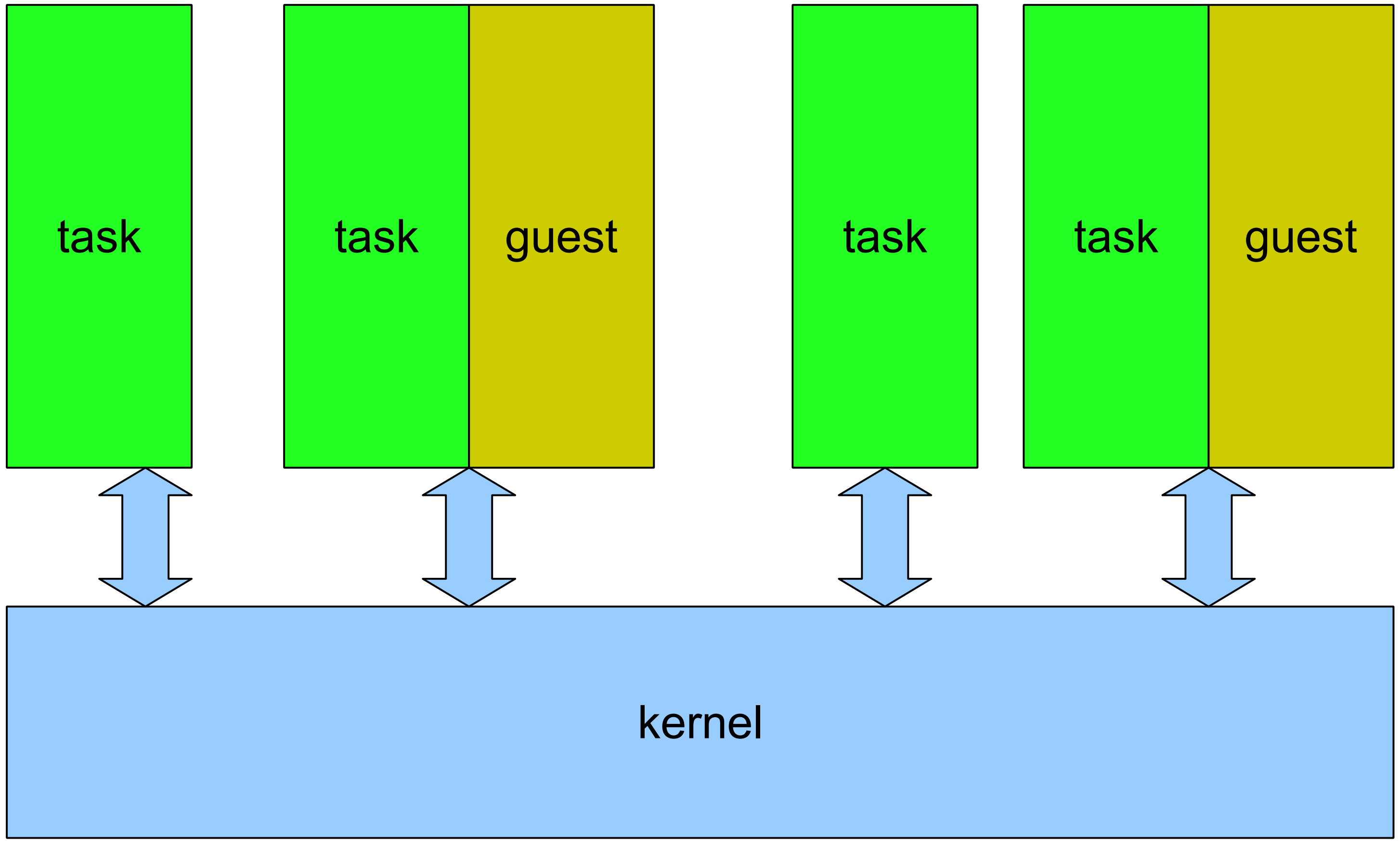
xen design GPL



KVM design... way to go!! GPL



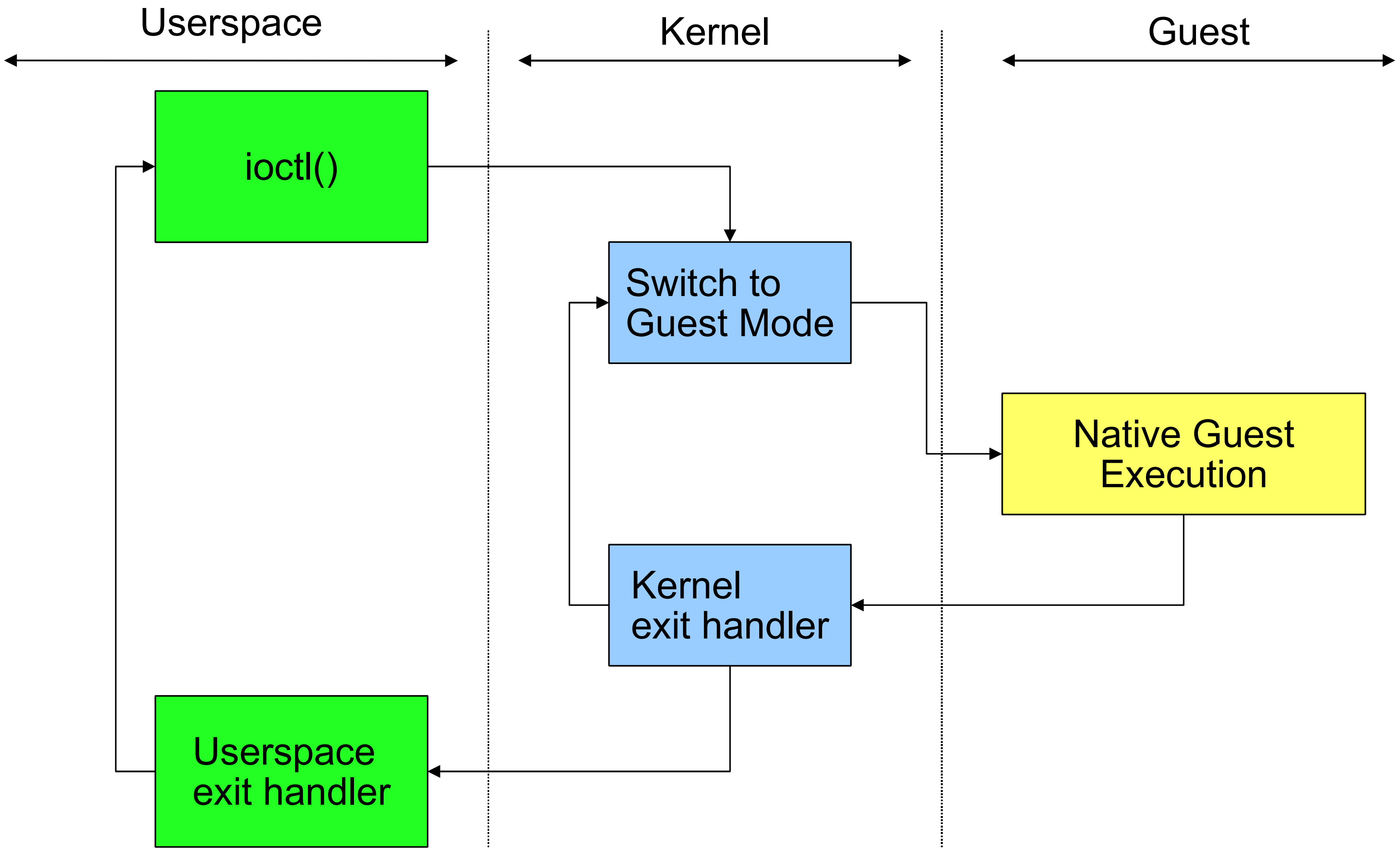
KVM task model



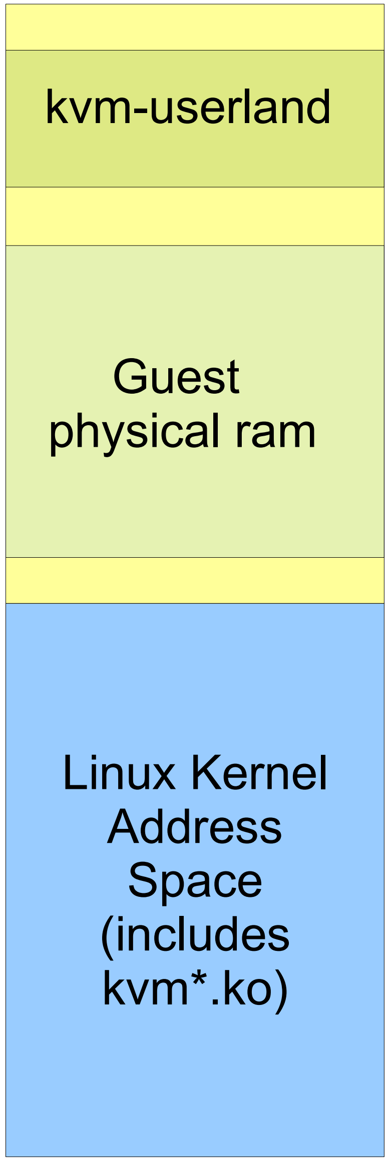
KVM gust mode

- 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 (each thread has its own guest mode)
- The Linux scheduler is responsible for scheduling a virtual cpu, as it is a normal thread
- The activity of the VM can be analyzed in the host with ptrace/strace/tcpdump -i tun0 etc...

KVM userland <-> KVM kernel



KVM process memory layout



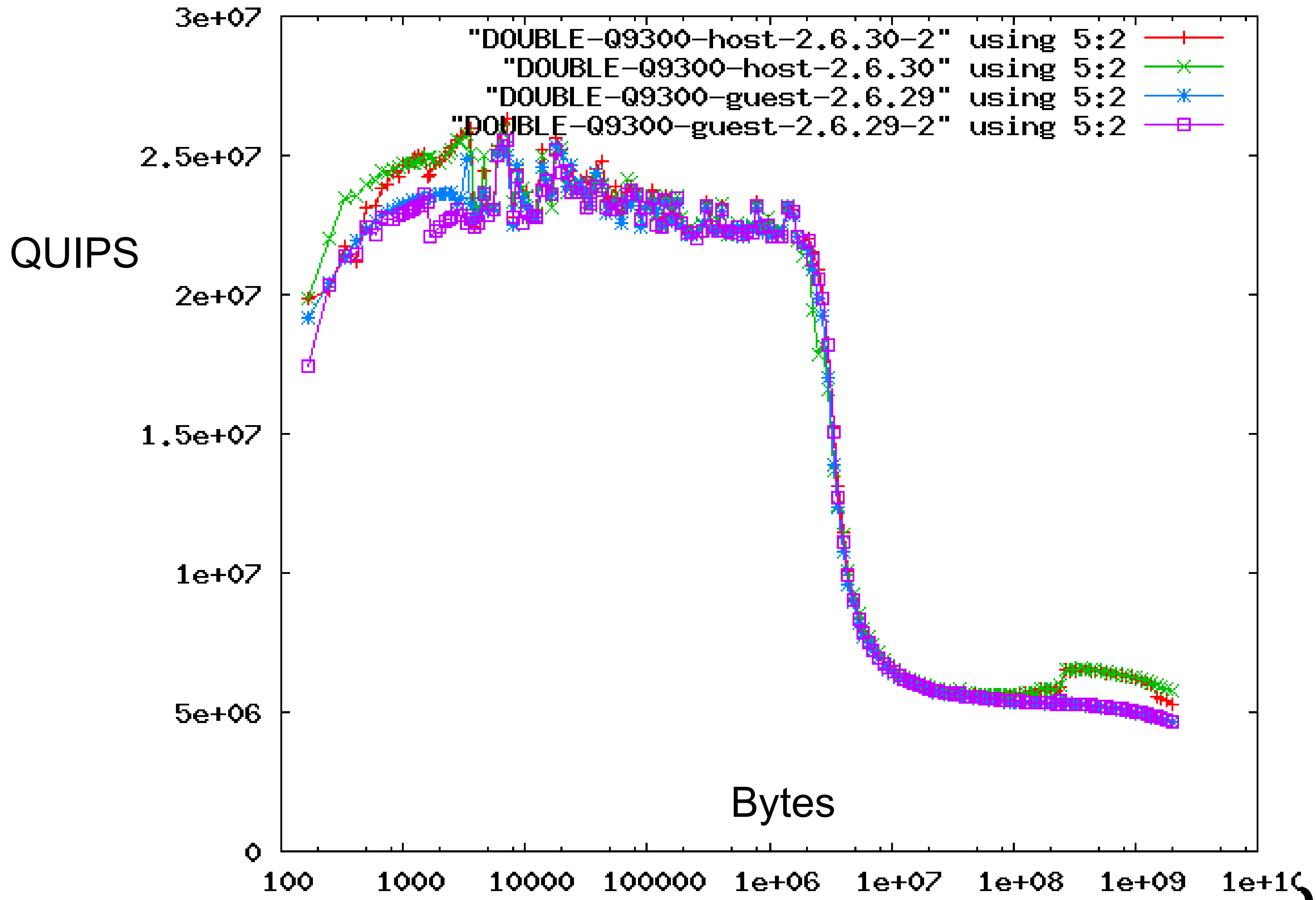
Linux Kernel integration

- 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

Hierarchical INTegration

- Quote from GPL package README
  - HINT reveals many aspects of computer performance: operation speed, precision, memory bandwidth, and usable memory size. It is completely unlike conventional computer benchmarks.
  - Although the human tendency is to try to turn performance into a single number, our hope is that machines will be compared using the entire QUIPS curve and not just the Net QUIPS rating.

HINT guest vs host Q9300



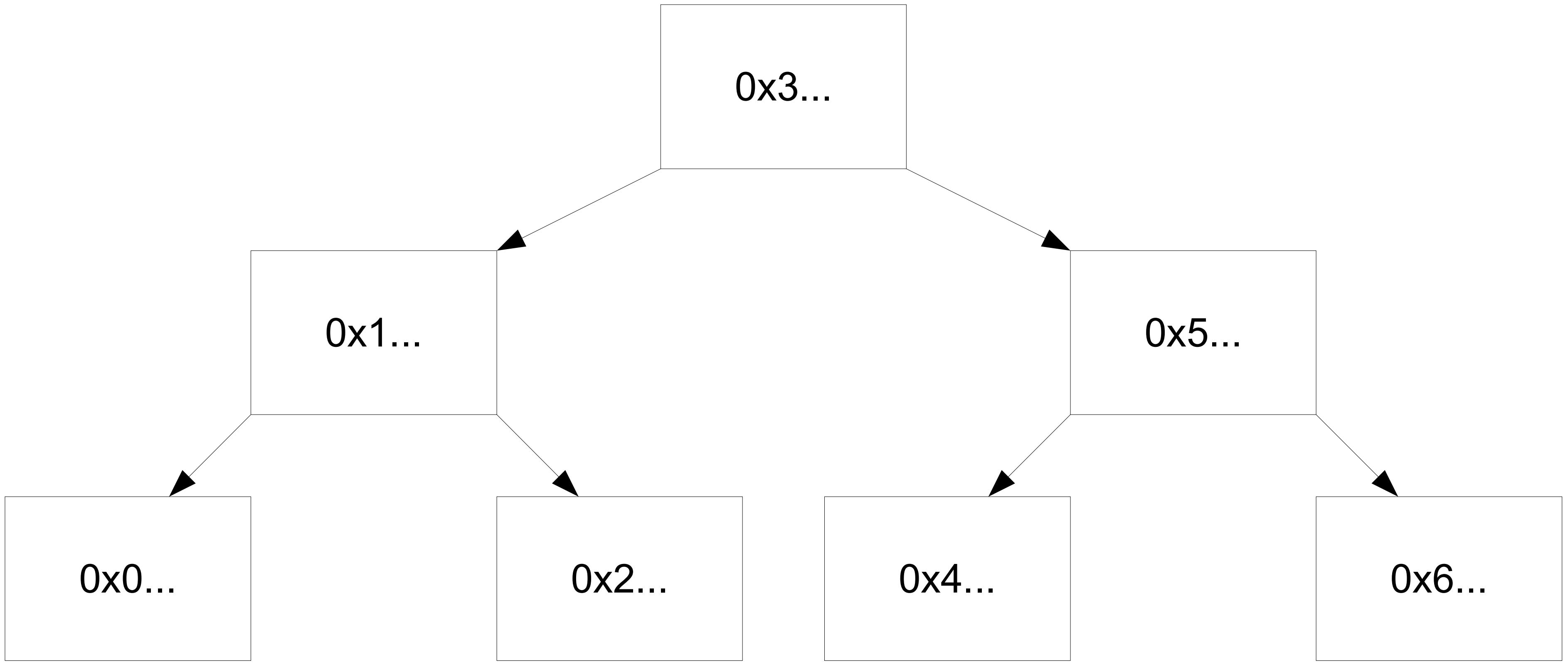
KSM

- KSM is a feature added to the Linux VM to allow merging equal anonymous pages
- Thanks to the MMU notifier, all KVM guest physical memory is identical to regular anonymous memory even when shadow pagetables are mapping it
- KSM works for both regular tasks as for KVM
- KSM is very useful in computation environments (notably the LHC scientific computing)
- The app requires changes to use KSM if it doesn't run under KVM, but **if the app runs inside KVM no change is required**!!

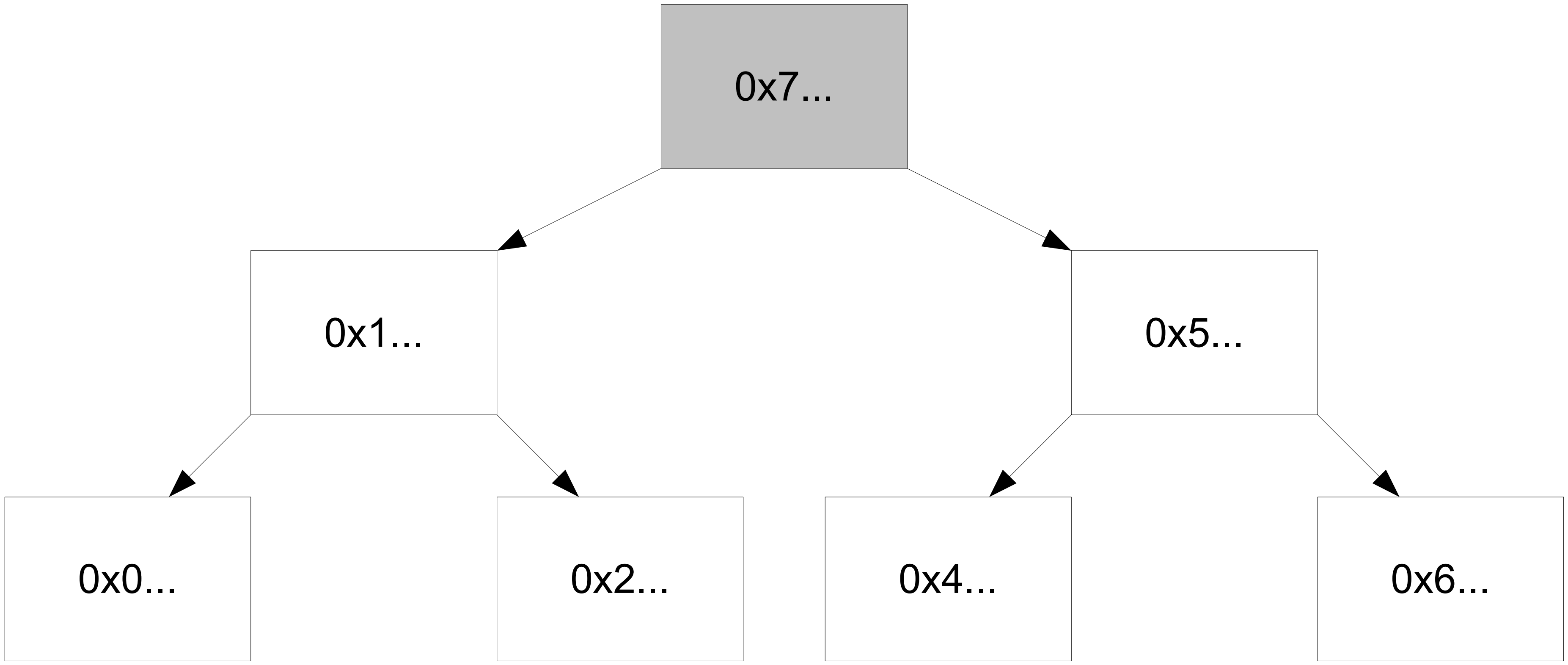
KSM tree

- KSM sorts the pages by raw page content with memcmp() in two trees:
  - Stable tree
    - Only contains KSM pages shared, in turn wrprotected (page content and hence index doesn't change under the tree)
  - Unstable tree
    - Contains all tracked anonymous pages merge candidates, they can be mapped writable in the process
    - Page content and hence rbtree-node index can change under the tree

KSM stable tree



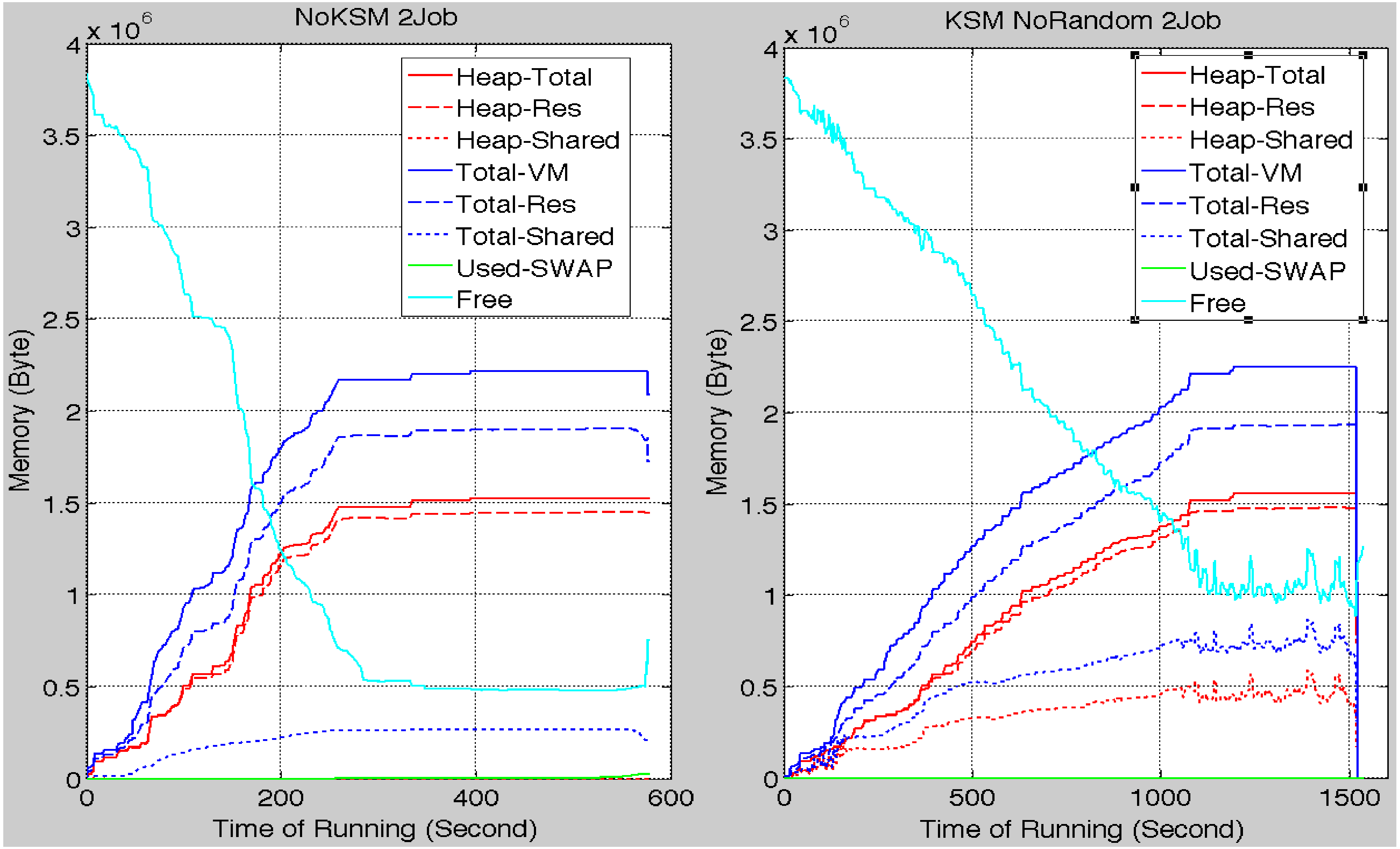
KSM unstable tree (gone unstable...)



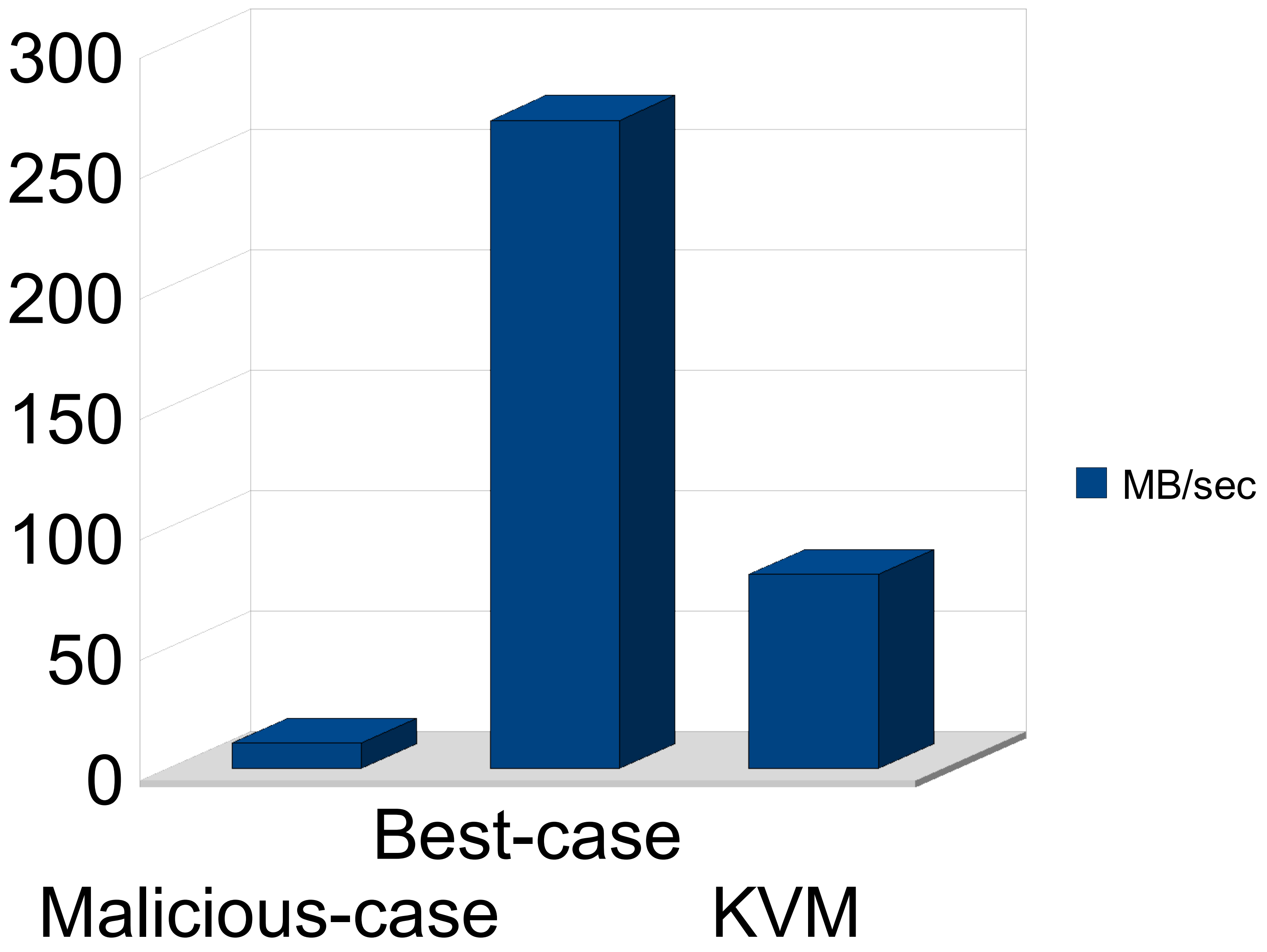
KSM unstable tree impact

- To deal with the instability of unstable tree:
  - We take advantage of the rbtree that rebalances without looking at the tree index to guarantees us the max level of the trees will remain <=2*log(N)
  - We only merge pages after a full memcmp with page wrprotection so merge is always 100% safe (the only risk is to delay the merging)
  - We add pages to the unstable tree only if their content didn't change for one full pass over all candidate pages
  - We rebuild the unstable tree from scratch at each pass

KSM @ LHC frees 500M of system RAM



KSM tree algorithm performance



KSM references

- Andrea Arcangeli: pending KSM paper @ http://linuxsymposium.org/ July 2009 Montreal
- Vincenzo Innocente: Summary of the evaluation of KSM for sharing memory in a multiprocess environment @ https://twiki.cern.ch/twiki/bin/view/LCG/EvaluationKSM0409

RHEV-M is ideal for clouds

- RHEV-M/H management/hypervisor
- Very efficient MMU (ideal for computations)
- Libvirt/qemu very flexibile and user friendly and trivial to script
- Just using tap-fd like VDE it's even possible to create a p2p encrypted mac-enforced (or routed) secure virtual ethernet (p2p cloud computing)
- Applications unpacked at boot on top of qcow2 base image, or distributed as child images
- KVM takes advantage of the optimal Linux VM to provide swapping and transparent KSM

Ideal virtualization @ CERN

- Deploy RHEL5.4/RHEV to the grid
    - This will make KVM, KSM, libvirt available to all compute nodes in the grid
- Allow scientist to deploy qcow2 images to the grid, all computations will run inside KVM
    - Manage, migrate the VM with RHEV-M (auto shutdown, rhev-h upgrades etc...)
  - Not all grid department forced to use RHEV-M
      - They can use libvirt (then Xen becomes almost transparent)
      - They can manage KVM themselves

Many cores one KVM

- Assume hardware has 16 physical CPU
  - Run the deployed qcow2 with -cpu 16 (16 vcpus)
- Assume your application can only be threaded 4-way
  - Run 4 copies of your threaded application with 4 threads each inside the same guest image
  - Use taskset -c 0-3, taskset -c 4-7 and/or cgroups..... in the guest

Many cores multiple KVM

- Alternative if it makes life easier
  - Run 4 KVM
  - Option: use taskset -c 0-3 and/or cgroups … in the linux host

Q/A

- You're very welcome!