System-level Virtualization for High-Performance Computing

Stephen L. Scott, Geoffroy R. Vallée, Thomas Naughton, Anand Tikotekar,
Christian Engelmann, Hong H. Ong (Oak Ridge National Laboratory)

Research and development areas

- Efficient hypervisor technology for limiting interferences with scientific applications in high-performance computing systems
- Minimal host operating system for reduced system footprint of system-level virtualization solutions in high-performance computing environments
- System management tools for supporting virtualized and standard HPC systems in disk-full and disk-less scenarios with various virtualization solutions
- Performance characterization of scientific applications running in virtual machines
- Configurable virtual system environments for adaptation of high-performance computing system properties to scientific application needs

Motivation: Portability

- HPC system hardware upgrades or new HPC system installations have become annual or even semi-annual events for many HPC centers
- Similarly, HPC system software upgrades have become monthly or even semi-monthly events
- *There is a constant need to port the same set of scientific applications to new or upgraded systems*
- *Annual or semi-annual HPC system upgrades or new installations incur the highest porting overhead*

Motivation: Configurability

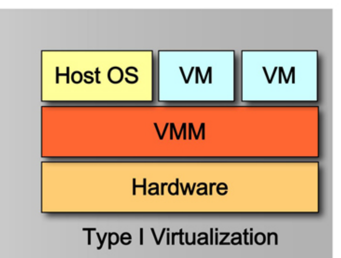
- There is no one-size-fits-all HPC OS solution
- Some HPC applications just need a scalable light-weight OS solution, like Catamount, and MPI
- Other HPC applications need the advanced features provided by a heavy-weight OS, such as Linux
- Vendors and the HPC OS community offer hybrid solutions with limited Linux functionality at scale
- *On-demand OS deployment on HPC systems is needed to fit scientific application needs*

Motivation: Testbeds

- New or enhanced system software solutions need to be tested at scale without corrupting the existing system software deployed on a HPC system
- New or enhanced scientific applications need to be tested at scale without the need of performing a full-scale production-type run
- *Large-scale testbeds are needed for HPC system software and scientific application development*

Virtualized System Environment (VSE)

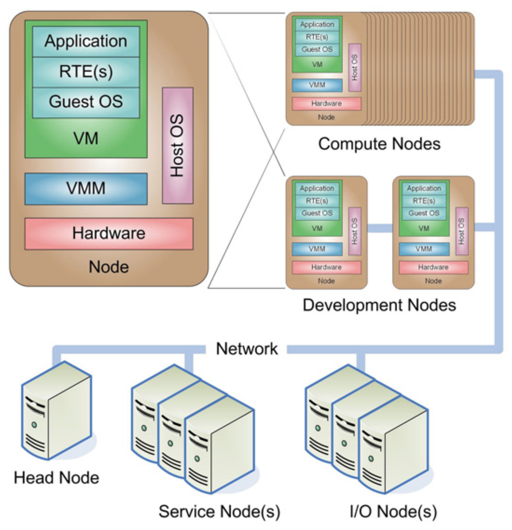
- Hypervisors can provide a configurable 'sandbox' environment for system software and scientific application development and deployment
- System-level virtualization on development systems (desktops and small HPC systems) and production-type systems (large HPC systems) can provide:
  - Simplified application porting through virtualization
  - On-demand OS deployment on virtualized HPC systems
  - On-demand deployment of virtual testbeds isolated from the real systems and from each other via a hypervisor



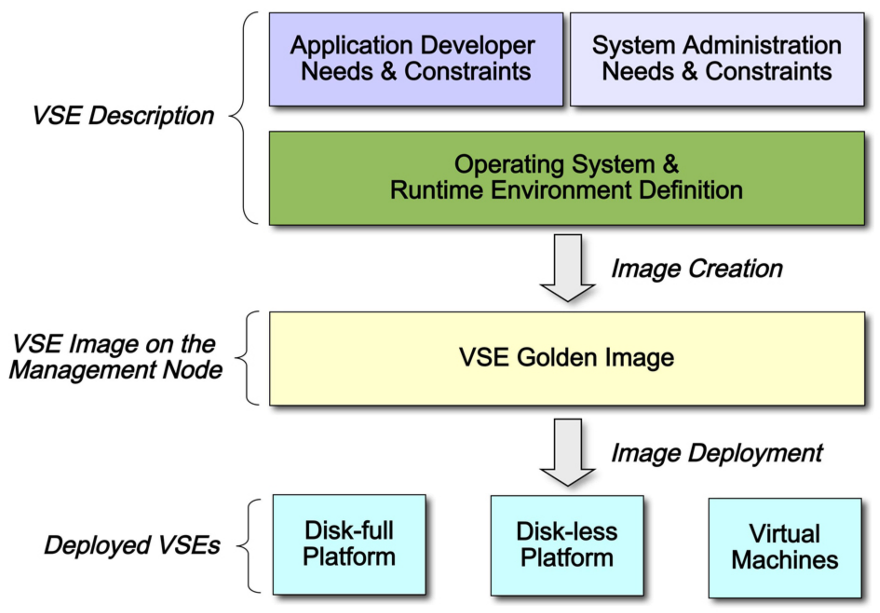
Type I Virtualization

VSE System Architecture

- Hypervisor on development and compute nodes
- Virtual machines run the customized virtualized environment
- Customization is based on:
  - Application needs
  - System capabilities
  - Resource allocation

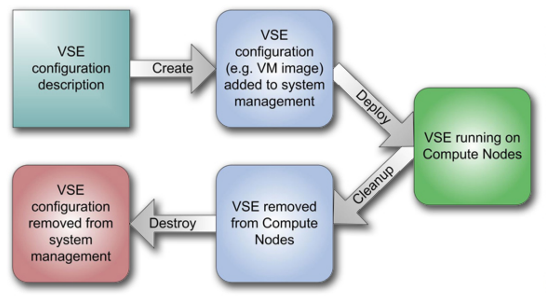


VSE Management



VSE Life Cycle

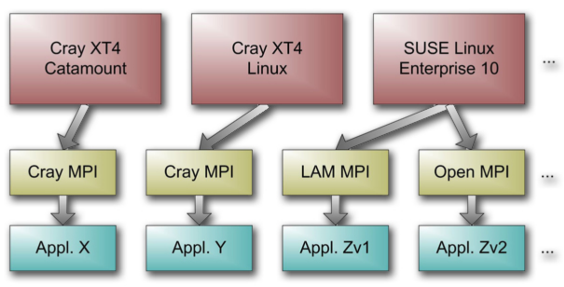
- System management tools allow for virtual system environment configuration:
  - Description
  - Creation
  - Deployment
  - Cleanup
  - Destruction
- Adaptation of existing VM management tools to system resource management and software development tools.



VSE Configuration Management

- Hierarchical configuration scheme enables users to:
  - Override
  - Remove
  - Add

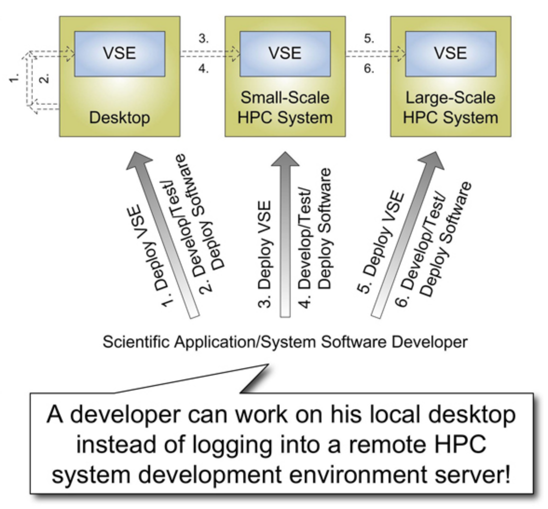
  configuration options.
- Vendor and/or system operator configuration descriptions can be used as base configuration



VSE Use Case Scenarios

- Application and system software developers can deploy virtualized system environments based on their actual needs to:
  - Desktops
  - Small-scale HPC systems
  - Large-scale HPC systems

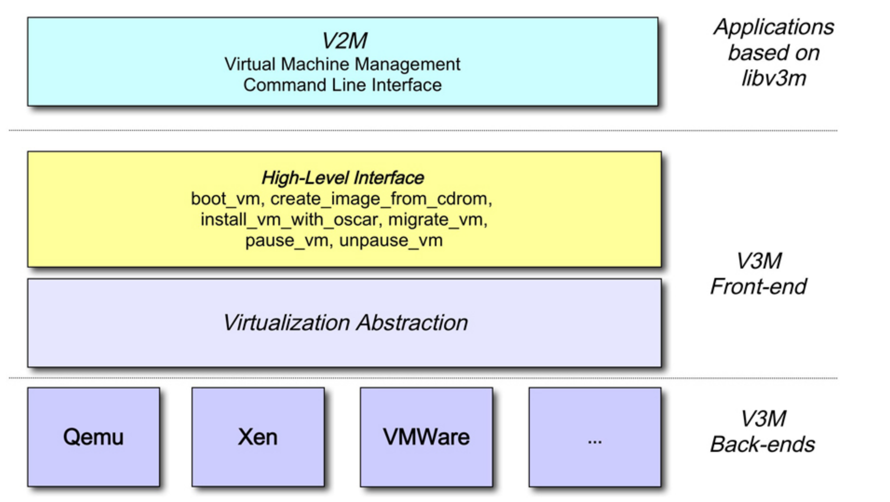
  for software development and deployment activities.



OSCAR-V: System Management with Virtualization Support

- Extension of Open Source Cluster Application Resources (OSCAR) Linux cluster installation and management suite
- Includes system-level virtualization support:
  - Capability to switch between virtual and standard cluster computing environments
- Abstracts underlying virtualization solution:
  - Generic virtual machine management (V2M) layer
  - Capability to switch between different virtualization solution
- VSE configuration consists of a set of OSCAR packages
- Support for various Linux distributions: SUSE, RedHat, Debian, ...

V2M Architecture

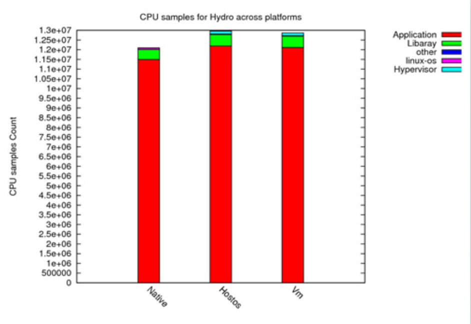


Performance Characterization & Analysis

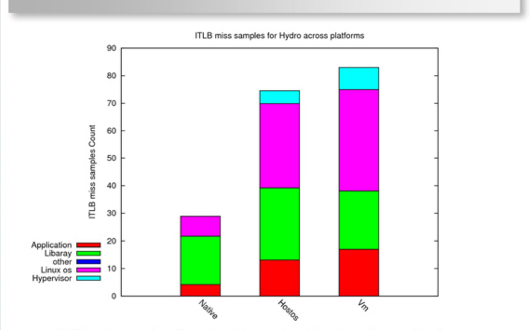
- *Goal*: Understanding the impact of system-level virtualization on scientific applications in detail
- *Experiment*: Hyperspectral Radiative Transfer Code
  - 2GHz Pentium IV, 768 MB of memory, Xen 3.0.4
  - Comparison of native, virtual machine, and host OS for:
    - CPU consumption
    - ITLB misses
    - DTLB misses

Breakdown for CPU time

- User code (Application) dominates
- More time for Hypervisor than Guest/Host kernel
- CPU time – Native vs. Virtual
  - User code: Slightly faster on Native
  - System code: Twice as fast on Native
  - System code: Variability higher on Native



* VM values only reflect domU samples (dom0 portion omitted)



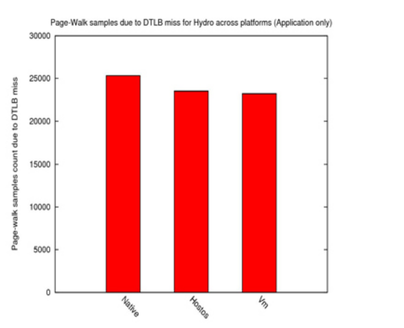
* VM values only reflect domU samples (dom0 portion omitted)

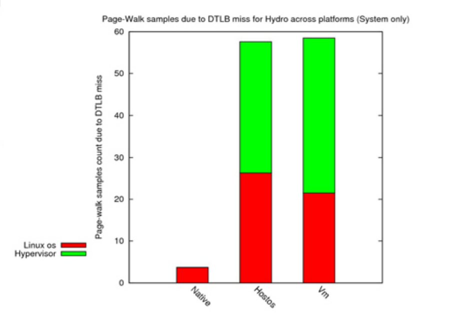
Breakdown for ITLB samples

- Fewer misses for Hypervisor than Guest/ Host kernel
- ITLB misses – Native vs. Virtual
  - User code: More on Native
  - System code: More on VM
  - Noted high variability on all platforms

Breakdown for Page table walks for DTLB miss samples

- User code: Native vs. Virtual
  - Page-walks caused by DTLB misses is higher on Native
- System code: Native vs Virtual
  - Page-walks caused by DTLB misses is much less compared to User code
  - Native is 14X less than virtual
  - Observed higher standard deviation on Native for system code





* VM values only reflect domU samples (dom0 portion omitted)

Ongoing Studies

- Hypervisor for high-performance computing
  - Low-profile virtual machine monitors (VMMs)
  - Modular VMMs for adaptation
  - Efficient I/O using VMM-bypass:
    - Isolation vs. performance
    - RDMA support
  - Optimizations for modern hardware features, such as IOMMU, Intel-VT, and AMD-V
- Tiny domains
  - Decrease the size of the host OS and VMs
  - Minimize overall system footprint

Office of Science
U.S. DEPARTMENT OF ENERGY

Point of Contact: Stephen Scott (scottsl@ornl.gov)

Oak Ridge National Laboratory