System-Level Virtualization & OSCAR-V

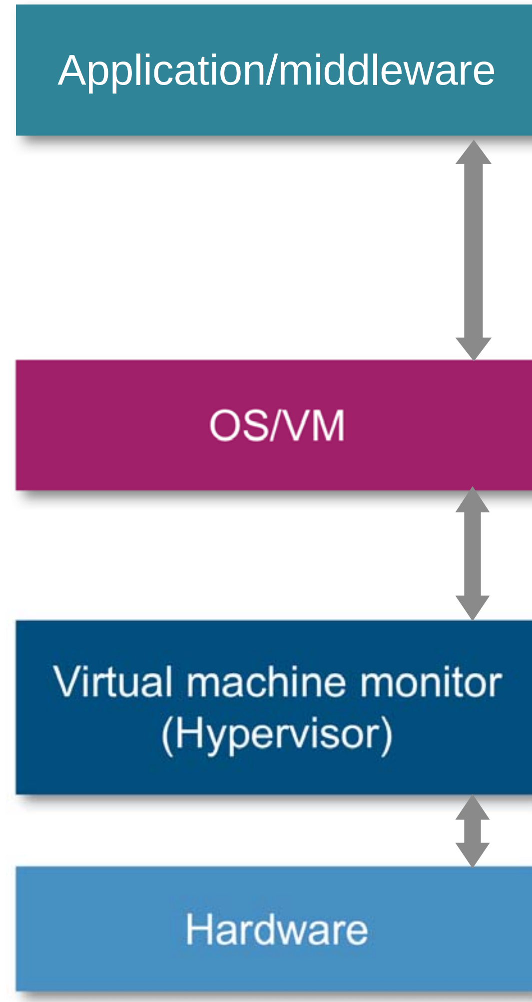
Presented by

Stephen L. Scott
Thomas Naughton
Geoffroy Vallée

Computer Science Research Group
Computer Science and Mathematics Division

Virtualization technologies

- Application/middleware
  - Software component frameworks
    - Harness, Common Component Architecture
  - Parallel programming languages and environments
    - PVM, MPI, Co-Array Fortran
  - Serial programming languages and environments
    - C, POSIX (Processes, IPC, Threads)
- OS/VM
  - VMWare, Virtual PC, Virtual Server, and Qemu
- Hypervisor
  - Xen, Denali
- Hardware
  - OS Drivers, BIOS, Intel VT, AMD-V (Pacifica)



Oak Ridge National Laboratory

Emerging system-level virtualization

- Hypervisors
  - OS-level virtual machines (VMs)
  - Paravirtualization for performance gain
    - Intercept and marshal privileged instructions issued by the guest machines
  - Example: Xen + Linux
- HPC using virtualization
  - Example: Xen + Linux cluster + Infiniband (OSU/IBM)
    - Hypervisor (Host OS) bypass directly to IB

Why hypervisors in HPC?

- Improved utilization
  - Users with differing OS requirements can be easily satisfied, e.g., Linux, Catamount, others in future
  - Enable early access to petascale software environment on existing smaller systems
- Improved manageability
  - OS upgrades can be staged across VMs and thus minimize downtime
  - OS/RTE can be reconfigured and deployed on demand
- Improved reliability
  - Application-level software failures can be isolated to the VMs in which they occur
- Improved workload isolation, consolidation, and migration
  - Seamless transition between application development and deployment using petascale software environment on development systems
  - Proactive fault tolerance (preemptive migration) transparent to OS, runtime, and application

What about performance?

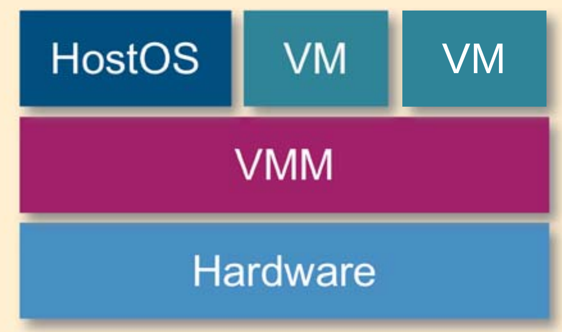
- Today hypervisors cost around 4–8% CPU time
- Improvements in hardware support by AMD and Intel will lessen this impact
- Proactive fault tolerance improves efficiency
  - Nonstop computing through preemptive measures
  - Significant reduction of checkpoint frequency
- Xen-like Catamount effort by Sandia/UNM to use Catamount as a HPC hypervisor

Virtual system environment

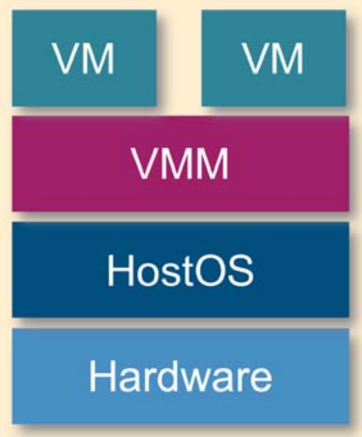
- Powerful abstraction concept that encapsulates OS, application runtime, and application
- Virtual parallel system instance running on a real HPC system using system-level virtualization
- Addressed key issues
  - Usability through virtual system management tools
  - Partitioning and reliability using adaptive runtime
  - Efficiency and reliability via proactive fault tolerance
  - Portability and efficiency through hypervisor + Linux/Catamount

System-level virtualization

- First research in the domain, Goldberg – 73
  - Type-I virtualization
  - Type-II virtualization
- Xen created a new real interest
  - Performance (paravirtualization)
  - Open source
  - Linux based
- Interest for HPC
  - VMM bypass
  - Network communication optimization
  - Etc.



Type I Virtualization



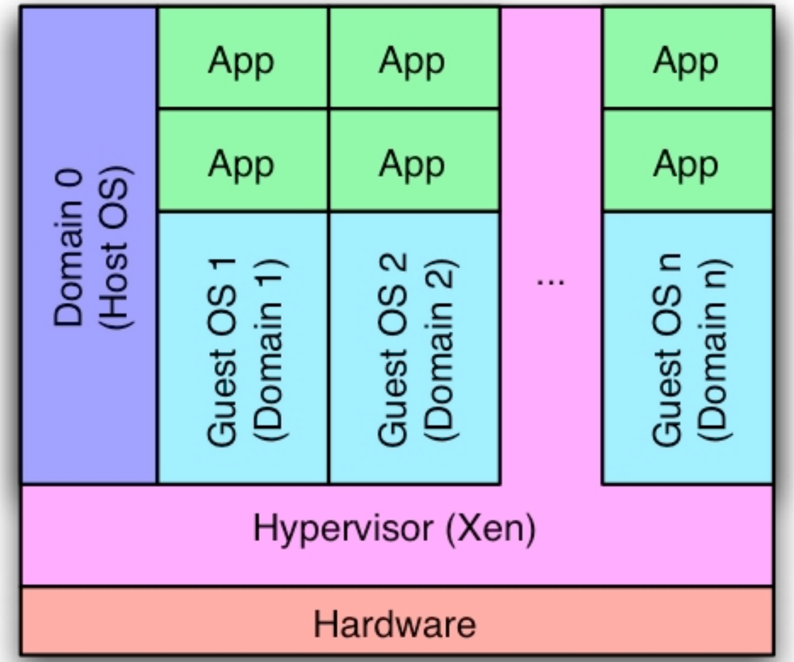
Type II Virtualization

Virtual machines

- Basic terminology
  - Host OS: The OS running on a physical machine
  - Guest OS: The OS running on a virtual machine
- Today different approaches
  - Full virtualization: Run an unmodified OS
  - Paravirtualization: Modification of OS for performance
  - Emulation: Host OS and Guest OS can have different architecture
  - Hardware support: Intel-VT, AMD-V

Type-I: Architecture

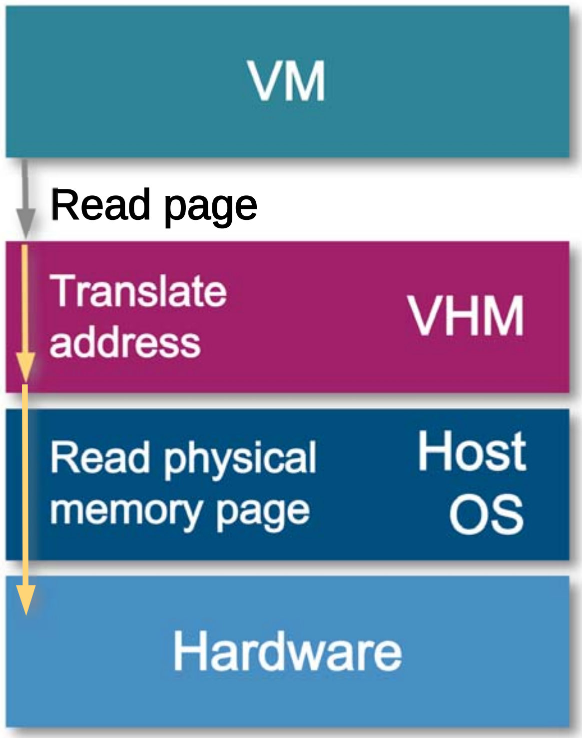
- Device drivers typically not included in the hypervisor
- Couple hypervisor + Host OS
  - Host OS includes drivers (used by hypervisor)
  - VMs access hardware via the Host OS



Source: Barney Maccabe, University of New Mexico

Type-II: Architecture

- Simpler model
  - Host OS and the hypervisor are "stacked"
  - No modifications to OSs
  - Provide a BIOS simulation
- Well suited for architecture emulation
  - Ex., PPC on x86_64
- Less efficient than type-I virtualization
  - Especially to paravirtualization



Why a hypervisor specifically *for* HPC?

- Networking
  - Bridges vs. zero copy (VMM bypass)
  - No RDMA support
- Memory: Important vs. minimal memory footprint
- Processor: Current solutions treat multicores as SMPs
- Tools: No tools available for the management of hundreds of VMs, hypervisors, and Host OSs

Three approaches

Investigate the development of an HPC hypervisor

1. New hypervisor from scratch
2. New hypervisor using the microkernel Catamount
3. New hypervisor modifying and extending Xen

Oak Ridge National Laboratory

1 Hypervisor from scratch

- Develop a new hypervisor using GeekOS
- Current status: A minimal hypervisor has been developed supporting Intel-VT

Pros
- Only necessary features
- Very small system footprint

Cons
- Longer-term effort

2 Hypervisor based on Catamount

- Extend Catamount to
  - Be used as hypervisor
  - As Guest OS
- Current status: Catamount ported to XenoLinux

Pros
- Very small system footprint
- Provide the XT environment within the VMs

Cons
- Still based on the Xen hypervisor

3 Xen-based hypervisor

- Remove unneeded Xen features
- Extend the hypervisor for adaptation (concept of modules)
- Current status
  - Paravirtualization supported
  - Working toward full virtualization (Intel-VT, AMD-V)
  - Adaptation capability
  - Designed FY 2007
  - Implementation FY 2008

Pros

- Quick prototyping
- Compatibility with emerging architectures

Cons

- No optimization (yet)

Reaping the benefit of virtualization: Proactive fault tolerance

- Context
  - Large-scale systems are often subject to failures due to the number of distributed components
  - Checkpoint/restart does not scale very well
- Provide capabilities for proactive fault tolerance
  - Failure prediction
  - Migrate application away from faulty node
    - Without stopping application
    - No application code knowledge (or code modification)

Oak Ridge National Laboratory

Proactive fault tolerance
(System and application resilience)

- Modular framework
  - Support virtualization: Xen, VMM-HPC
  - Designed to support process-level checkpoint/restart and migration
  - Proactive fault tolerance adaptation: Possible to implement new policies using our SDK
- Policy simulator
  - Ease the initial phase of study of new policies
  - Results from simulator match experimental virtualization results

Management of virtualized environments

- Current issues similar to real systems
  - How to deploy a VM?
  - How to configure a VM?
  - How to deploy multiple VMs?
- Reduce complexity (hide technical details); a VM is just
  - An architecture
  - Some NICs
  - Some memory
  - Etc.

System management issues

Current solutions for virtual environments	Current solutions for standard HPC systems
• Image repository • Solutions developed from scratch	• Mature system management software, e.g., OSCAR, Rocks • System definition • Deployment

- Management of VM images
- Management of Host OS
- Deployment of both Host OSs and VM images
- Distributed environment (clusters of VMs)

OSCAR-V

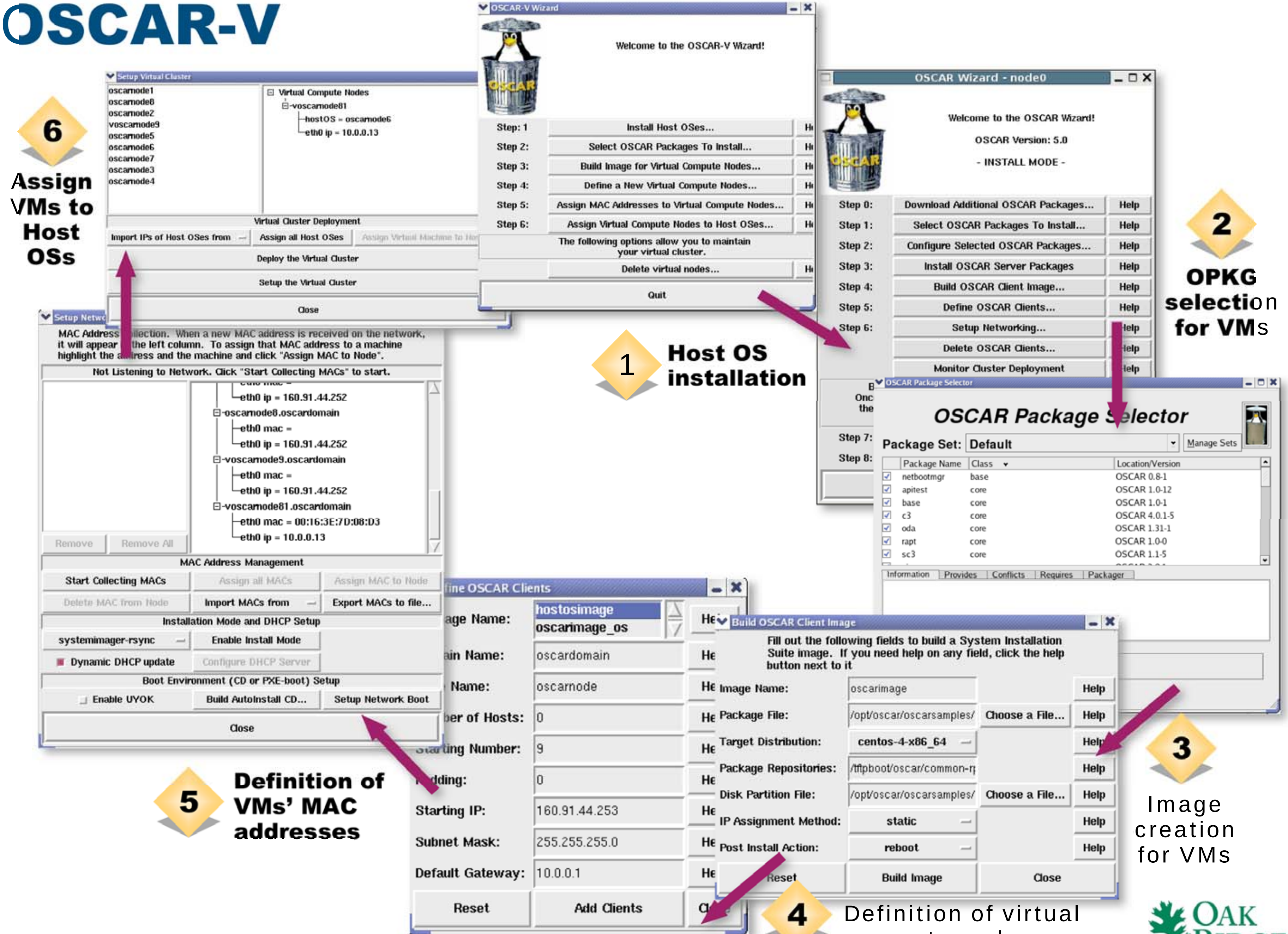
Enhancements to support virtual clusters

- OSCAR-core modifications
- Create OSCAR Packages for virtualization solutions
- Integrate scripts for automatic installation and configuration

Abstracts differences in virtualization solutions

- Must provide abstraction layer and tools—*libv3m/v2m*
- Enable easy switch between virtualization solutions
- High-level definition and management of VMs: Mem/cpu/etc., start/stop/pause

OSCAR-V



1 Host OS installation
2 OPKG selection for VMs
3 Image creation for VMs
4 Definition of virtual compute nodes
5 Definition of VMs' MAC addresses
6 Assign VMs to Host OSs
OSCAR-V Wizard
Welcome to the OSCAR-V Wizard!
Step: 1 Install Host OSes...
Step 2: Select OSCAR Packages To Install...
Step 3: Build Image for Virtual Compute Nodes...
Step 4: Define a New Virtual Compute Nodes...
Step 5: Assign MAC Addresses to Virtual Compute Nodes...
Step 6: Assign Virtual Compute Nodes to Host OSes...
The following options allow you to maintain your virtual cluster.
Delete virtual nodes...
Quit
OSCAR Wizard - node0
Welcome to the OSCAR Wizard!
OSCAR Version: 5.0
- INSTALL MODE -
Step 0: Download Additional OSCAR Packages...
Step 1: Select OSCAR Packages To Install...
Step 2: Configure Selected OSCAR Packages...
Step 3: Install OSCAR Server Packages
Step 4: Build OSCAR Client Image...
Step 5: Define OSCAR Clients...
Step 6: Setup Networking...
Delete OSCAR Clients...
Monitor Cluster Deployment
OSCAR Package Selector
Package Set: Default
Build OSCAR Client Image
Image Name: oscarimage
Target Distribution: centos-4-x86_64
IP Assignment Method: static
Post Install Action: reboot
Build Image
Close
Starting IP: 160.91.44.253
Subnet Mask: 255.255.255.0
Default Gateway: 10.0.0.1
Reset
Add Clients
Setup Virtual Cluster
Virtual Compute Nodes
Virtual Cluster Deployment
Deploy the Virtual Cluster
Setup the Virtual Cluster
MAC Address Management
Start Collecting MACs
Installation Mode and DHCP Setup
Enable Install Mode
Boot Environment (CD or PXE-boot) Setup
Setup Network Boot

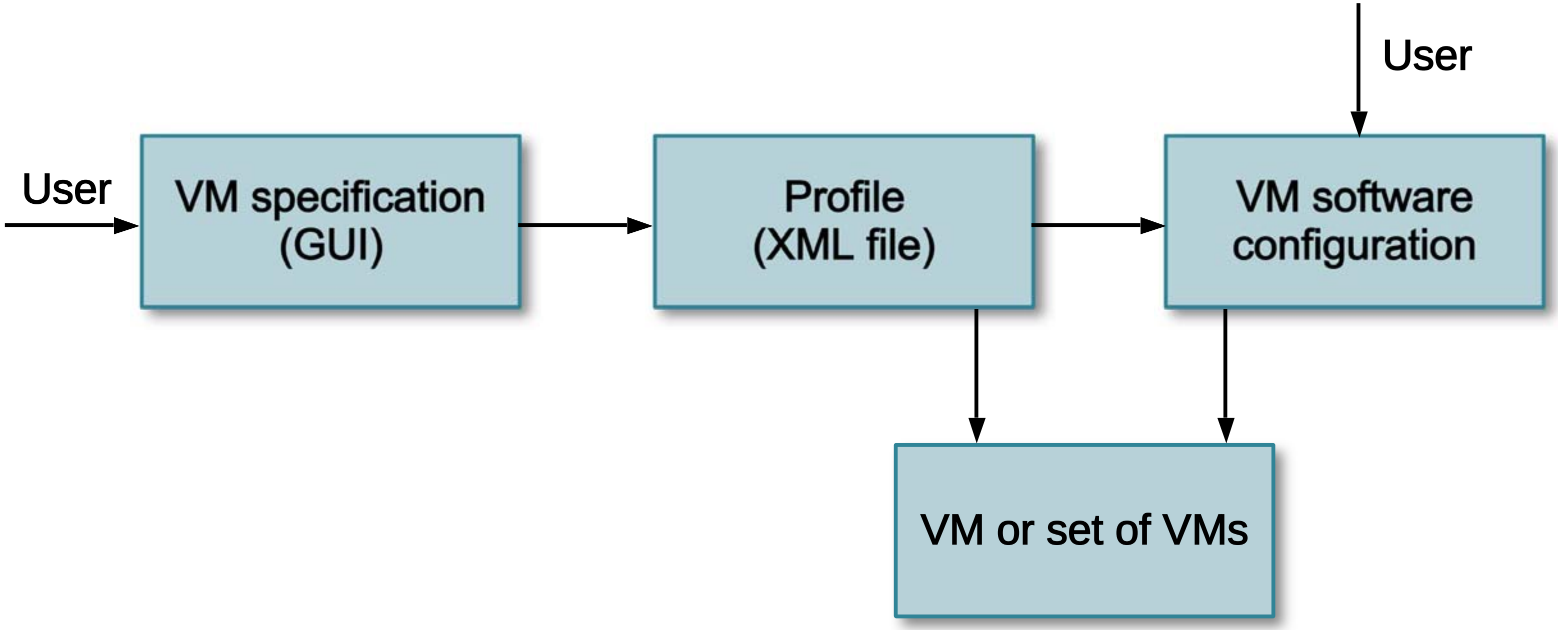
OSCAR-V: Description of steps

Initial setup

1. Install supported distro head node (host)
2. Download/set up OSCAR and OSCAR-V
    - OSCAR: Untar oscar-common, oscar-base, etc., and copy distro RPMs
    - OSCAR-V: Untar; run “make install”
3. Start Install Wizard
    - run “./oscarv $network_interface” and follow setups

OSCAR-V: VM profile management

- Concept of profiles
  - VMs: A profile is memory, disk, OS, NICs, network config.
  - Virtual distributed system: A profile is set of VM profiles

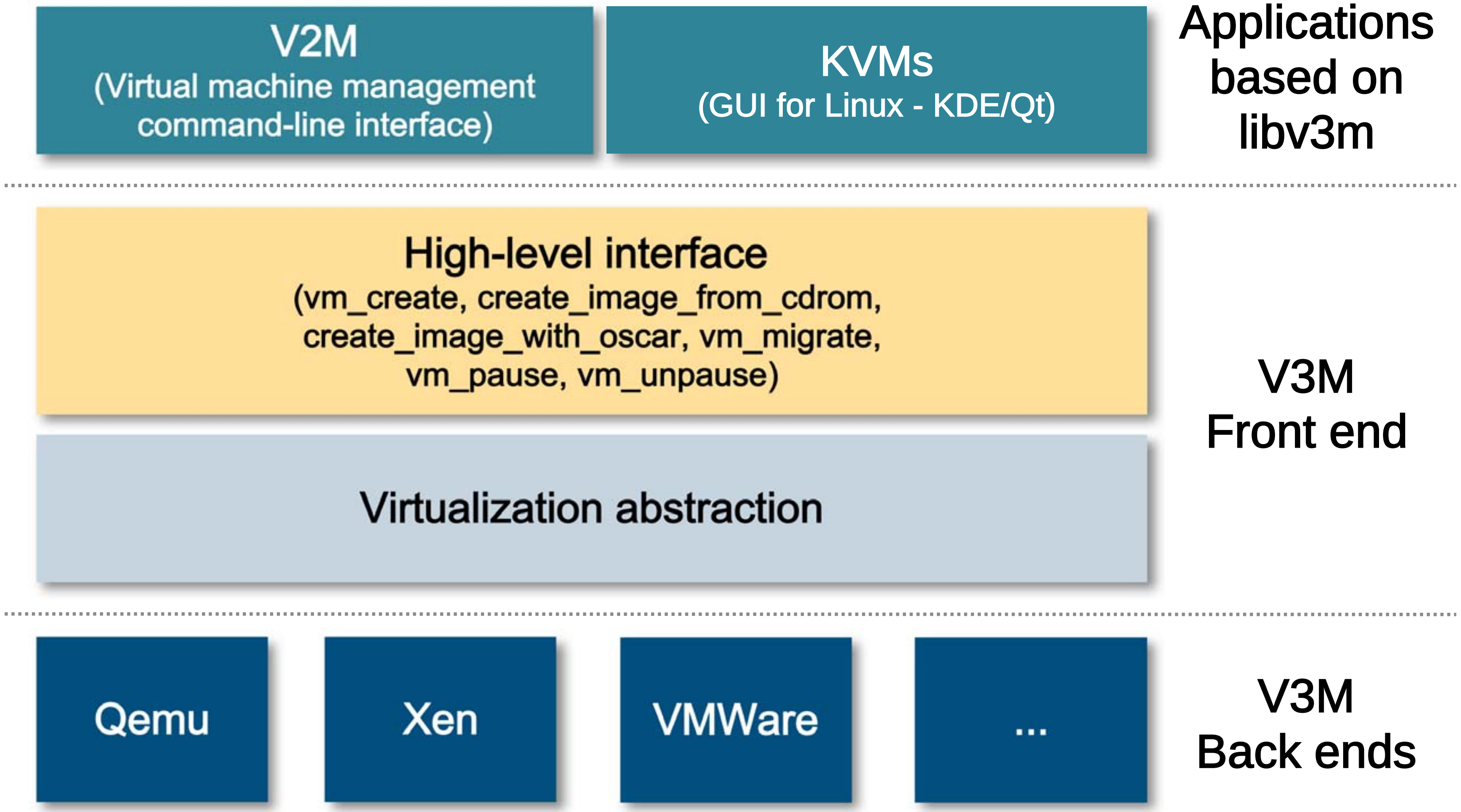


OSCAR-V: Virtual machine abstraction

Provide a simple, human-readable VM specification

```xml
<?xml version="1.0"?>
<!DOCTYPE profile PUBLIC "" "xen_vm.dtd">
<profile>
	<name>test</name>
	<image size="500">/home/gvallee/vms/test.img</image>
	<nic1>
		<mac>00:02:03:04:05:06</mac>
	</nic1>
</profile>
```

OSCAR-V: V2M – virtual machine management



OSCAR-V: V3M – functionality

- Check the system (files/tools)
- Check the profile (validation)
- Create configuration scripts for VM management
- Provide simple interface for VM management
  - Boot, image management, status
- Switch to a new virtualization solution
  - Change only the “type”

OSCAR-V: V3M – supported features summary

Supported features	Xen (paravirtualization)	Xen (full virtualization)	Qemu	VM ware
VM instantiation	Yes	Yes	Yes	Yes
VM image creation	Yes	Yes	Yes	No
Installation via CD-ROM	N/A	Yes	Yes	No
Installation via OSCAR	Yes	Yes	Yes	No
VM migration	Yes	Experimental	No	No
VM pause/unpause	Yes	Experimental	Experimental	Experimental
Virtual disk	Yes	Yes	Yes	Yes

OSCAR-V: Modifications for OSCAR-V

- SystemConfigurator modification
  - Used after the image copy on the remote node to do local configuration (IP, hostname, etc.)
  - Goal: Support Xen specific GRUB entries

    Title Xen system
    Root (hd0,0)
    Kernel /boot/xen.gz dom0_mem=131072
    Module /boot/vmlinuz-2.6.12-dom0 root=/dev/sda1 ro
    Module /boot/initrd.img-2.6.12
  - Add a new option to specify “module” options
  - Integrated in SystemConfigurator trunk
- kernel_picker modification
  - Allow one to include a specific kernel within an image
  - Set up a specific SystemConfigurator configuration file
  - Add a new option to specify “module” options

OSCAR-V: Image management

Host OS

- OSCAR Packages (OPKG) are available
  - Xen case: Xen hypervisor, Xen kernels (dom0, domU), Xen tools
- Use the unmodified OPKG/OPD mechanism
  - Automatically add software components
  - Automatically set up the virtualization solution
- Current limitation
  - Only REHL, CentOS, Fedora Core are currently supported

Virtual machines

- One OSCAR Package is available
  - Automatically includes the kernel (optional)
  - Automatically sets up the environment
- OSCAR can be used to define VMs
  - Set up the number of VMs
  - MAC addresses
  - IPs

Virtual machines may be deployed

OSCAR-V: Summary

- Capability to create image for Host OSs
  - Minimal image
  - Take benefit of OSCAR features for the deployment
  - Automatic configuration of system-level virtualization solutions
  - Complete networking tools for virtualization solutions
- Capability to create images for VMs
  - May be based on any OSCAR-supported distributions: Mandriva, SuSE, Debian, FC, Red Hat EL, etc.
  - Leverage the default OSCAR configuration for compute nodes

OSCAR-V: Current status

- Stabilization for a public release
  - OSCAR-V 1.0
    - Support of Xen full virtualization and paravirtualization
      - CentOS 4.4 x86_64 and x86
  - OSCAR modifications for OSCAR-V (still ongoing)
  - Road map
    - OSCAR-V 2.0: Add support of QEMU and KVM, CentOS 5 (x86_64, x86)
    - Google Summer of Code
      - VM monitoring via V2M
      - Support of other Linux distributions and architecture
      - Stabilize VM migration support (first prototype unsuitable for a public release)
- Resources
  - V2M/libv3m: http://www.csm.ornl.gov/srt/v2m.html
  - OSCAR-V: http://www.csm.ornl.gov/srt/oscarv.html
  - OSCAR: http://oscar.openclustergroup.org

OSCAR-V Collaborations: VM deployment on demand

- OSCAR-V does not allow for the automatic deployment of VMs at job submission time
- Integration of Dynamic Virtual Clusters (DVCs)
  - Moab extensions for VM deployment during job submission
  - Use OSCAR images; deployment-based on DVC
  - Collaboration with ASU (Dan Stanzione)

Oak Ridge National Laboratory

Contacts regarding System-Level Virtualization & OSCAR-V

Stephen L. Scott

Computer Science Research Group
Computer Science and Mathematics Division
(865) 574-3144
scottsl@ornl.gov

Thomas Naughton

Computer Science Research Group
Computer Science and Mathematics Division
(865) 576-4184
naughtont@ornl.gov

Geoffroy Vallée

Computer Science Research Group
Computer Science and Mathematics Division
(865) 574-3152
valleegr@ornl.gov

Oak Ridge National Laboratory