Operating System Research at ORNL:

System-level Virtualization

Presented by:

Christian Engelmann

engelmannc@ornl.gov

Systems Research Team

Computer Science Research Group
Computer Science and Mathematics Division
Oak Ridge National Laboratory
United States Department of Energy

Largest Multipurpose Science Laboratory within the U.S. Department of Energy

- Privately managed for US DOE
- $1.08 billion budget
- 4000+ employees total
  - 1500 scientists and engineers
- 3,000 research guests annually
- 30,000 visitors each year
- Total land area 58mi^2 (150km^2)

- Nation's largest energy laboratory
- Nation's largest science facility:
  - The $1.4 billion Spallation Neutron Source
- Nation's largest concentration of open source materials research
- Nation's largest open scientific computing facility

ORNL East Campus: Site of World Leading Computing and Computational Sciences

Computational Sciences Building

Research Office Building

Engineering Technology Facility

Systems Research Team

Old Computational Sciences Building (until June 2003)

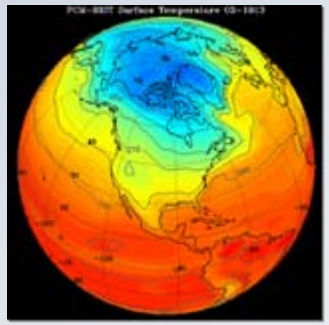
Joint Institute for Computational Sciences

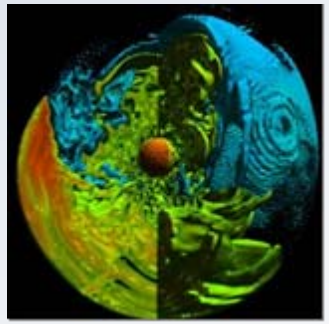
Research Support Center (Cafeteria, Conference, Visitor)

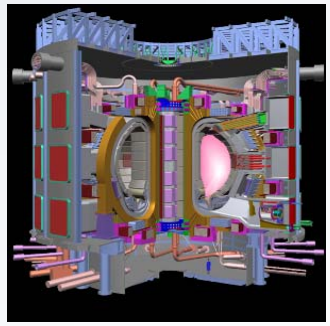
National Center for Computational Sciences

- **40,000 ft^2 (3700 m^2) computer center:**
  - **36-in (~1m) raised floor, 18 ft (5.5 m) deck-to-deck**
  - **12 MW of power with 4,800 t of redundant cooling**
  - **High-ceiling area for visualization lab:**
    - **35 MPixel PowerWall, Access Grid, etc.**
- **2 systems in the Top 500 List of Supercomputer Sites:**
  - **Jaguar: 10. Cray XT3, MPP with 12500 dual-core Processors ⇨ 119 TFlop.**
  - **Phoenix: 32. Cray X1E, Vector with 1014 Processors ⇨ 18 TFlop.**

At Forefront in Scientific Computing and Simulation







- Leading partnership in developing the National Leadership Computing Facility
  - Leadership-class scientific computing capability
  - 100 TFlop/s in 2006/7 (recently installed)
  - 250 TFlop/s in 2007/8 (commitment made)
  - 1 PFlop/s in 2008/9 (proposed)
- Attacking key computational challenges
  - Climate change
  - Nuclear astrophysics
  - Fusion energy
  - Materials sciences
  - Biology
- Providing access to computational resources through high-speed networking (10Gbps)

System Research Team

- SRT team
  - Stephen L. Scott
  - Christian Engelmann
  - Hong Ong
  - Geoffroy Vallee
  - Thomas Naughton
  - Sudharshan Vazhkudai
  - Anand Tikotekar
- Research topics
  - High Availability / Fault Tolerance
  - Operating Systems
  - Distributed Storage
  - Cluster Computing
  - Virtualization Technologies
  - Resource Management
  - Tools

System-level Virtualization - Outline

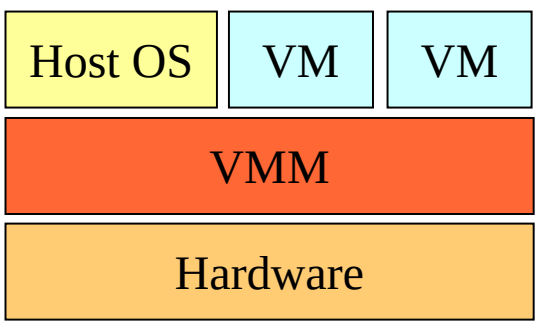
- *Introduction to system-level virtualization*
- Hypervisor for HPC
- High availability
- System management

Why Virtualization?

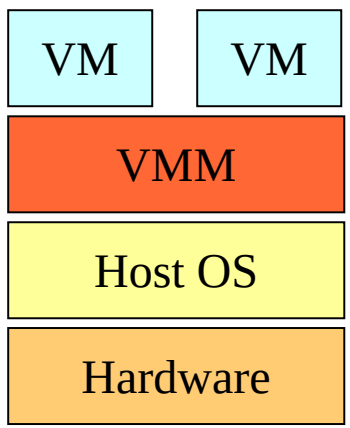
- Workload isolation
  - Improves security and reliability
    - Isolate software stack to own VM
    - Intrusions confined to VM scope
- Workload consolidation
  - Better resource utilization
    - Consolidate work to fewer servers
    - Support incompatible or legacy operating environments
- Workload migration
  - Improves quality of service
    - Workload balance
    - Failure detection/migration

Classification

- Two kinds of system virtualization
  - Type-I: the virtual machine monitor and the virtual machine run directly on top of the hardware,
  - Type-II: the virtual machine monitor and the virtual machine run on top of the host OS



Type I Virtualization



Type II Virtualization

Virtual Machines

- First studies in the 70's (*e.g.* “Architecture of Virtual Machines”, R.P. Goldberg, 1973)

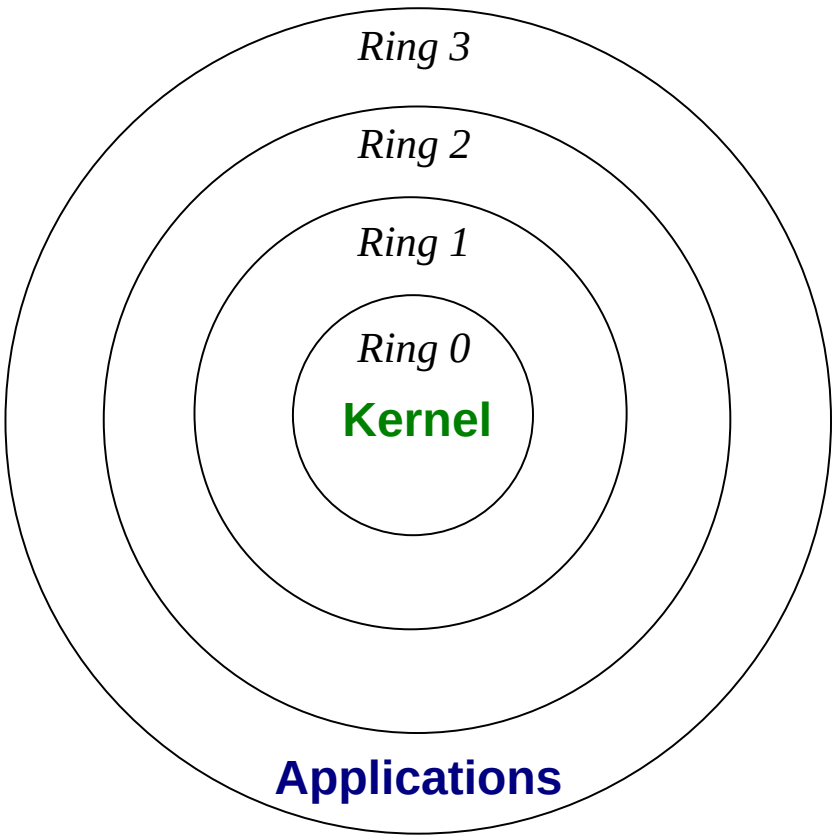
- Common idea
  - the physical machine runs an OS → Host OS
  - the virtual machine runs its own OS → Guest OS

- Today different approaches
  - para-virtualization: modification of the running OS for performance
  - full-virtualization: the running OS is not modified
  - emulation: the host OS has a different architecture than the guest OS
  - hardware support: Intel-VT, AMD-V

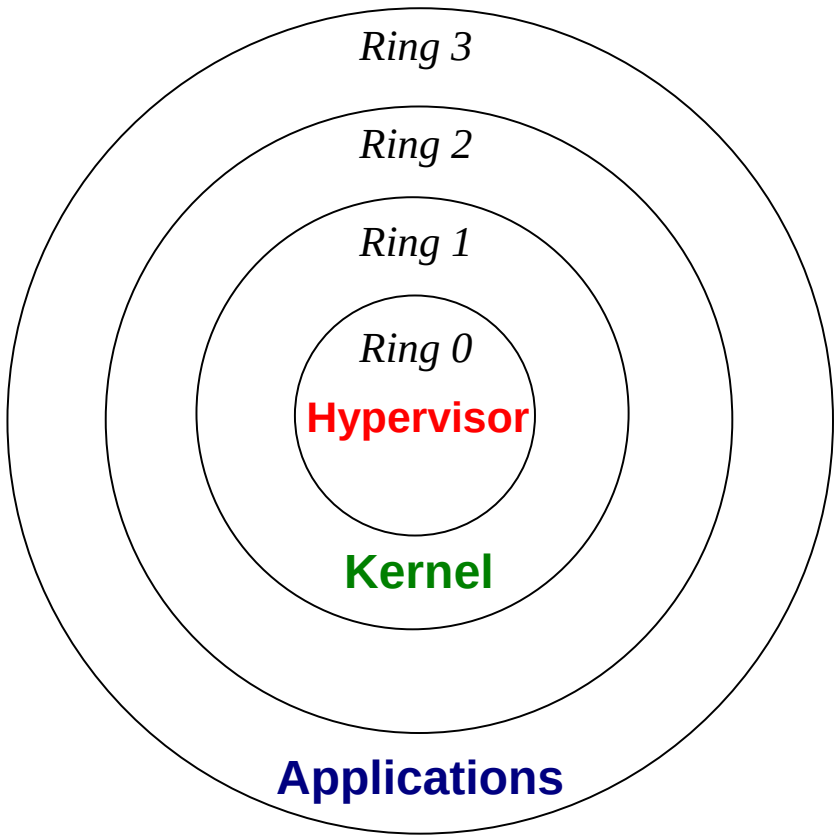
System-level Virtualization Solutions

- Number of solutions, e.g., Xen, QEMU, KVM, VMWare
- *Which one to use?*
  - *Type-I virtualization: performance*
  - *Type-II virtualization: development*

Type-I Virtualization - Design



x86 Architecture – Execution Rings



x86 Architecture – “Modified” Execution Rings

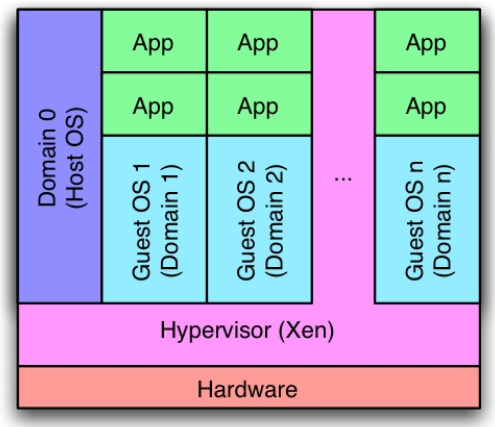
OAK RIDGE National Laboratory

Type-I Virtualization - Hypervisor

- Hypervisor is running in ring 0
- Kernels are running in ring 1
  - impossible to execute protected processor instructions
  - the hypervisor needs to hijack protected processor instructions (paravirtualization: Hypervisor calls, similar to syscalls)
  - overhead for all hypervisor calls
- Applications are still running in ring 3, no modifications

Type-I Virtualization – Device Drivers

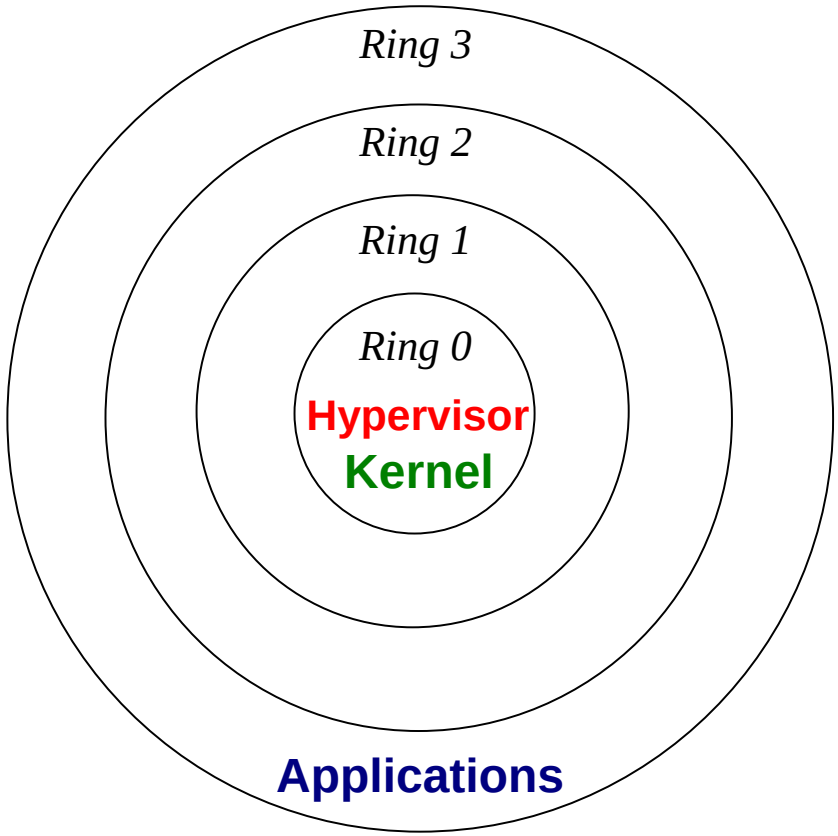
- VMs are running in ring 1, no access to devices
- Most of the time the hypervisor does not include device drivers
- Couple Hypervisor + Host OS
  - host OS includes drivers
  - hardware access from VMs are done through the Host OS



Source: Barney Maccabe, UNM

Type-I Virtualization – Hardware Support

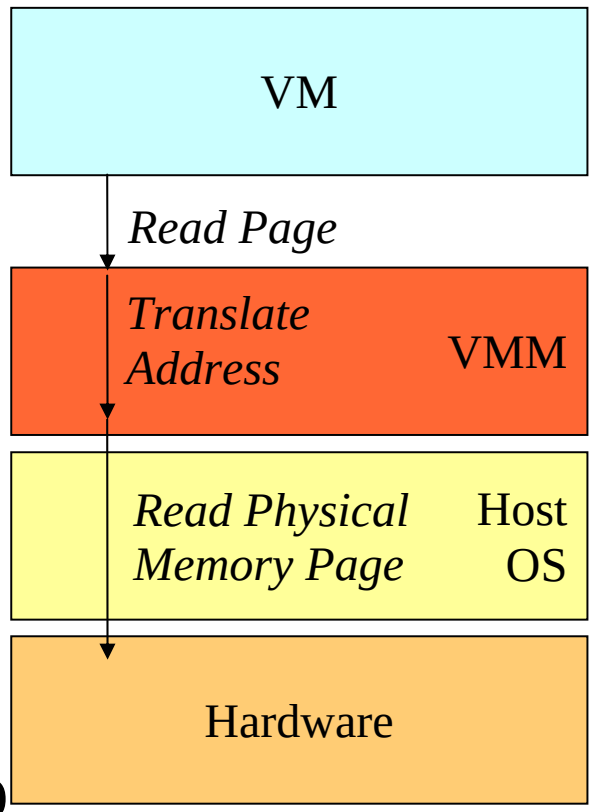
- Create a hardware "virtualized context"
- Transition from VM mode to "Hypervisor mode"
  - save registers
  - context switch (similar to process switch within a traditional OS)
- Current implementations: Intel-VT, AMD-V (not compatible)



OAK RIDGE National Laboratory

Type-II Virtualization - Design

- Simpler model: the Host OS and the Hypervisor are “stacked”
- No modifications of OSes
- Allow a BIOS simulation
- Easy to extend for emulation (e.g. PPC on x86_64)
- Less efficient than type-I virtualization (especially comparing to para-virtualization)



Example of Type-I Virtualization: Xen

- **Xen: *para-virtualization (type-I)***
  - *Adv.*: good performance for computation
  - *Problems*: overhead for I/Os, modification of the Linux kernel, start to be complex (driven by ASP market, different needs than ours), not a full virtualization of the system

Example of Type-II Virtualization

- **VMWare: *full-virtualization (type-II)***
  - *Adv.: mature*
  - *Problems: still difficult to adapt (not open source), not really suitable for HPC*
- ***QEMU: full-virtualization (type-II)***
  - *Adv.: open source, performance similar to VMWare, support a lot of architectures*
  - *Problems: performance not suitable for HPC*
- ***KVM: full-virtualization (type-II)***
  - *Adv.: open source, maintained by the Linux community*
  - *Problems: Linux as Hypervisor*

System-level Virtualization - Outline

- *Introduction to system-level virtualization*
- *Hypervisor for HPC*
- High availability
- *System management*

Context

- Large scale systems
  - XT3/XT4 Cray machine
  - ORNL Institutional Cluster
- Different users needs
  - Development environments may differ: from desktops to small clusters
- Variety of user applications

Challenges

- Difficult to provide a single execution environment
  - that fits users/applications needs
  - that is similar to all development environments
- Several challenges
  - plug-and-play computing
  - execution environment customization

A possible approach: System-level Virtualization

Xen for HPC?

- Xen has proven to be interesting for HPC
- But Xen also has drawbacks
  - it has become too large
  - supports many architectures (much legacy stuff, we are only interested in x86_64)
  - some functionalities interesting for HPC are periodically broken
  - monolithic
  - static

VMM for HPC

- LDRD Project at ORNL: ORNL, Sandia, UNM, Northwestern University
- Sandia/UNM: development of a VMM based on Catamount
- Northwestern: development of a dynamic hypervisor framework
- ORNL: development of a VMM based on Xen
  - Fork of Xen 2 (smaller), “take the best of Xen”
  - port on x86_64
  - write documentation (education)
  - focus on only functionalities for HPC

VMM for HPC

- Goals
  - be able to modify dynamically Hypervisor's behavior (T. Naughton, G. Vallee)
  - have a small hypersivor, i.e., w/ a small footprint (G. Vallee, T. Naughton)
  - efficient I/Os
    - VMM by-pass
    - API and integration of IOMMU
  - have documentation!

In collaboration with Northwestern University (Prof. Peter. Dinda)

System-level Virtualization - Outline

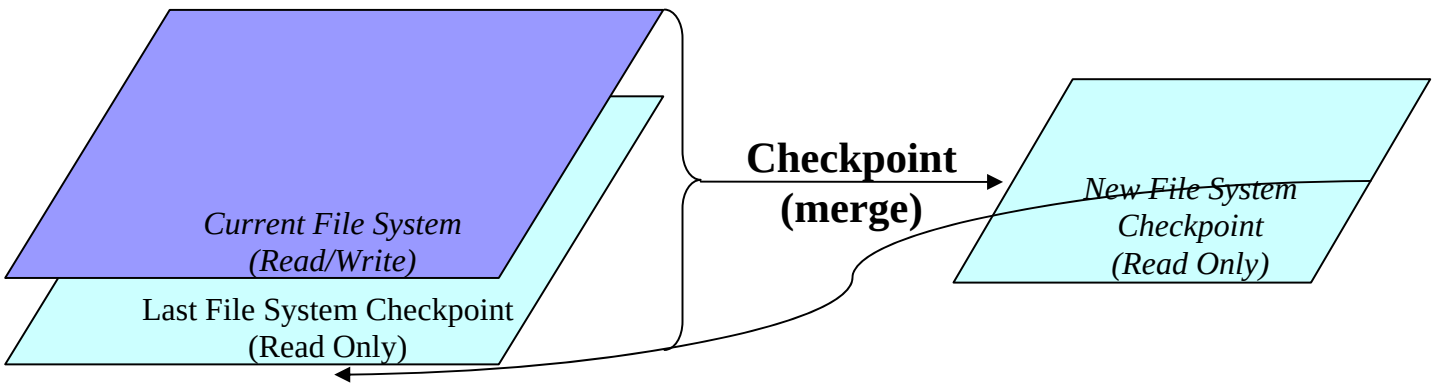
- *Introduction to system-level virtualization*
- Hypervisor for HPC
- *High availability*
- *System management*

System-level Virtualization and High Availability

- Current studies based on Xen
- Virtualization provides interesting mechanisms for HA: VM live migration, VM checkpoint/restart
  - Live migration is efficient
  - Checkpoint/restart is not yet complete (better with Xen 3): no checkpoint of the file system

VM Checkpoint / Restart

- Already possible to checkpoint the memory (memory dump in a file)
- File system checkpoint/restart



In collaboration with LATech (Prof. Box Leangsuksun)

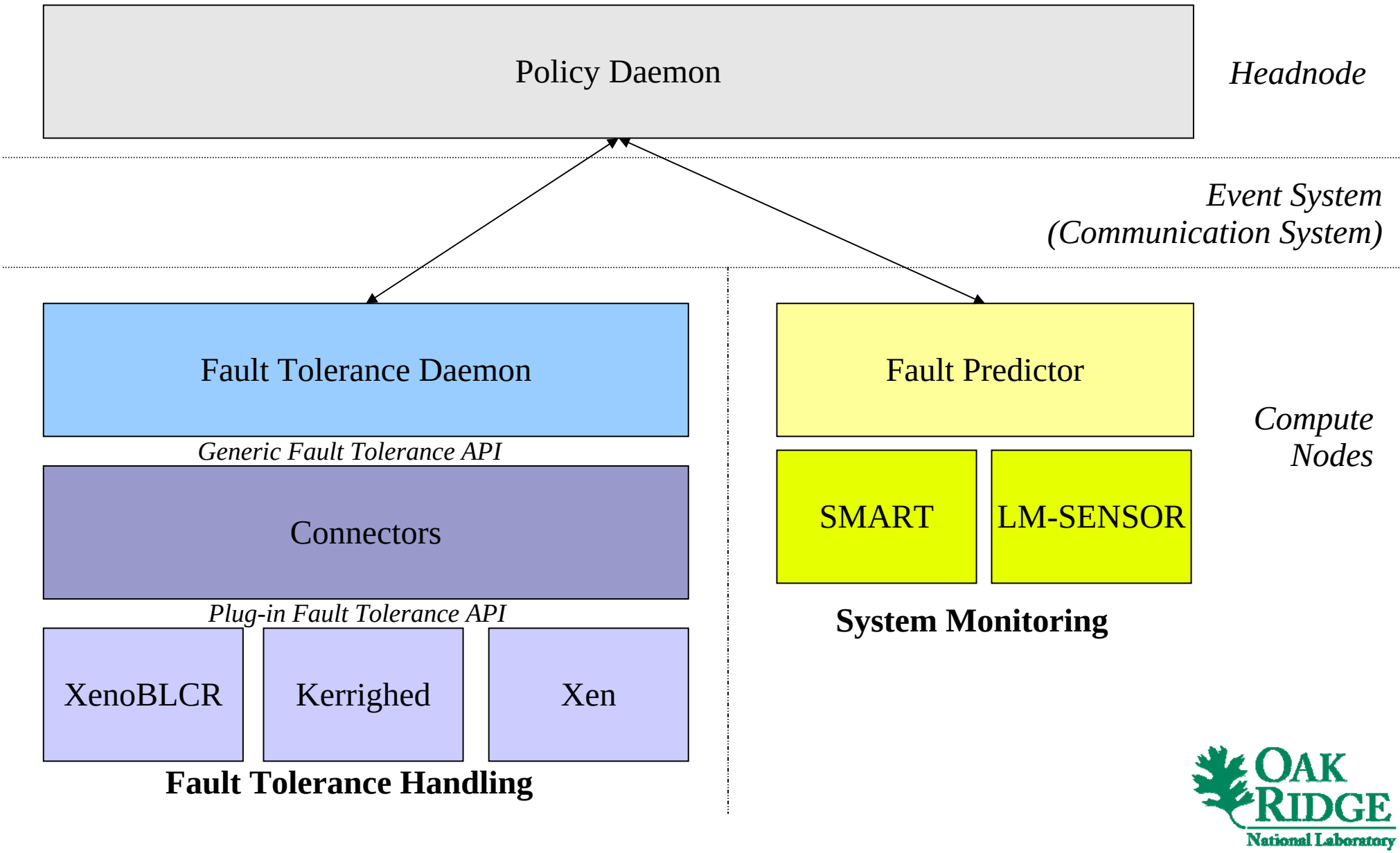
HA Framework Based on Virtualization (1)

- *Goal*: take benefit of system abstraction provided by system virtualization
- *Pro-active FT*: migrate, pause VMs before failure occurs when something wrong is detected
  - implementation of a framework to provide various policies regarding when and how to checkpoint/migrate

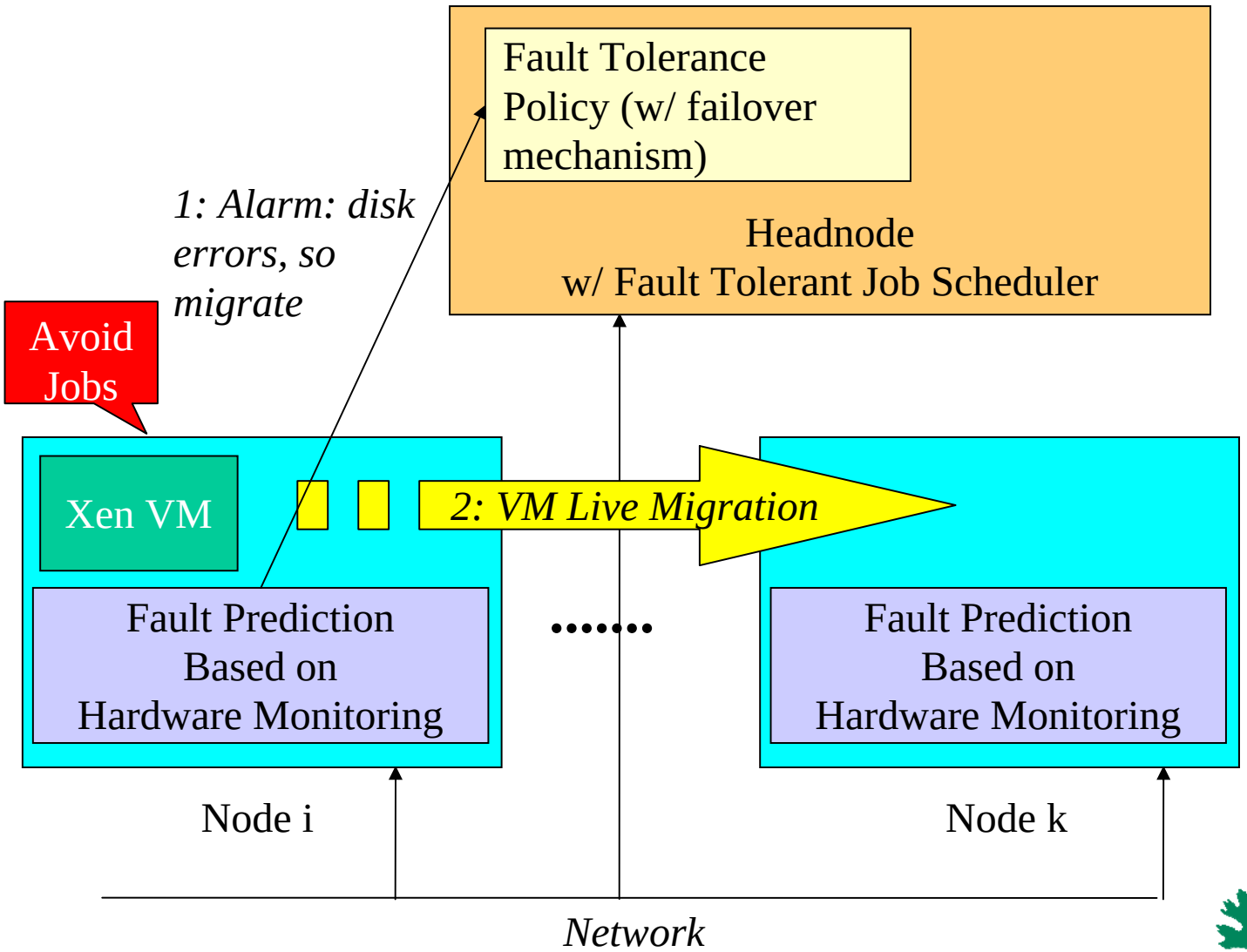
HA Framework Based on Virtualization (2)

- Reactive FT
  - checkpoint/restart of applications with BLCR
  - BLCR port to Xenolinux
  - active/standby duplication for service on headnode
- Combining pro-active and reactive FT
  - pro-active FT avoids issues when problems are detected before failures
  - reactive restart applications after a failure and stop pro-active FT mechanism

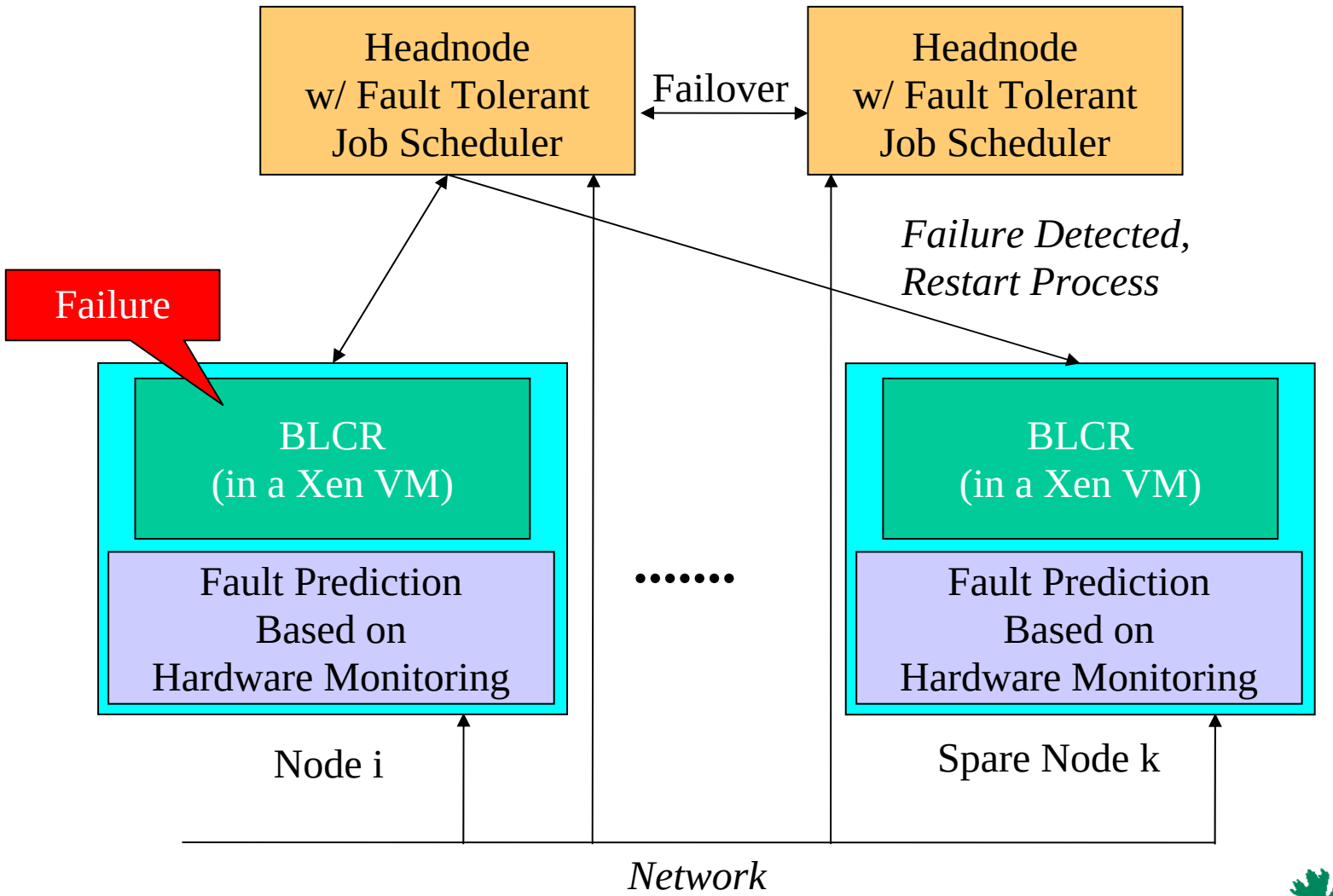
Framework Architecture



Pro-active Fault Tolerance



Reactive Fault Tolerance



System-level Virtualization - Outline

- *Introduction to system-level virtualization*
- Hypervisor for HPC
- High availability
- *System management*

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?
- Users do not want to deal with technical details, a VM is just
  - an architecture
  - some NICs
  - some memory
  - etc.

Abstraction of System Virtualization Solutions - Introduction

- Users
  - do not want to deal with technical details
  - wants to specify their need in term of VMs with an high-level description (memory, disk, etc.)
- Virtualization solutions have different benefits
  - Xen: performance
  - QEMU: full system emulation, eases developments

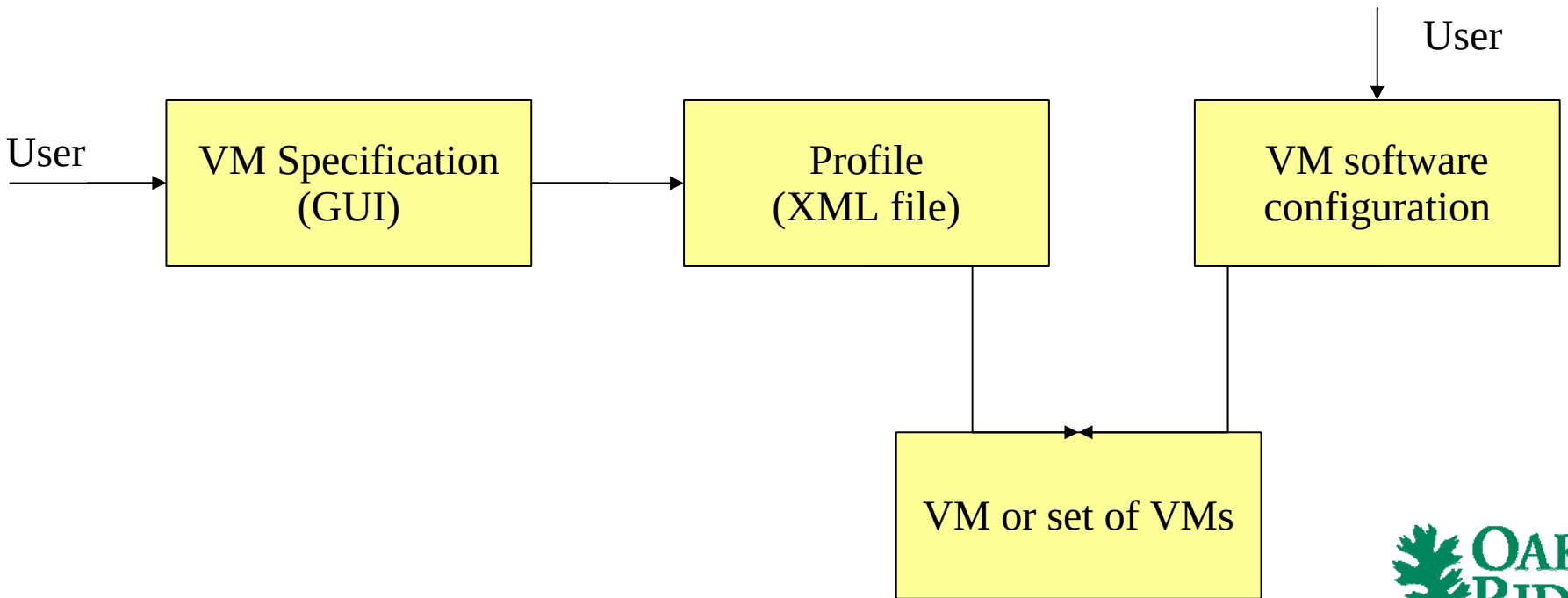
V3M – VM Profile

- Simple, high-description of VMs
- Example

```
<?xml version="1.0"?>
<!DOCTYPE profile PUBLIC "" "v3m_profile.dtd">
<profile>
        <name>test</name>
        <type>Xen</type>
        <image size="50">/home/gvallee/temp/v2m/test_xen.img</image>
        <nic1>
                <type>TUN/TAP</type>
                <mac>00:02:03:04:05:06</mac>
        </nic1>
</profile>
```

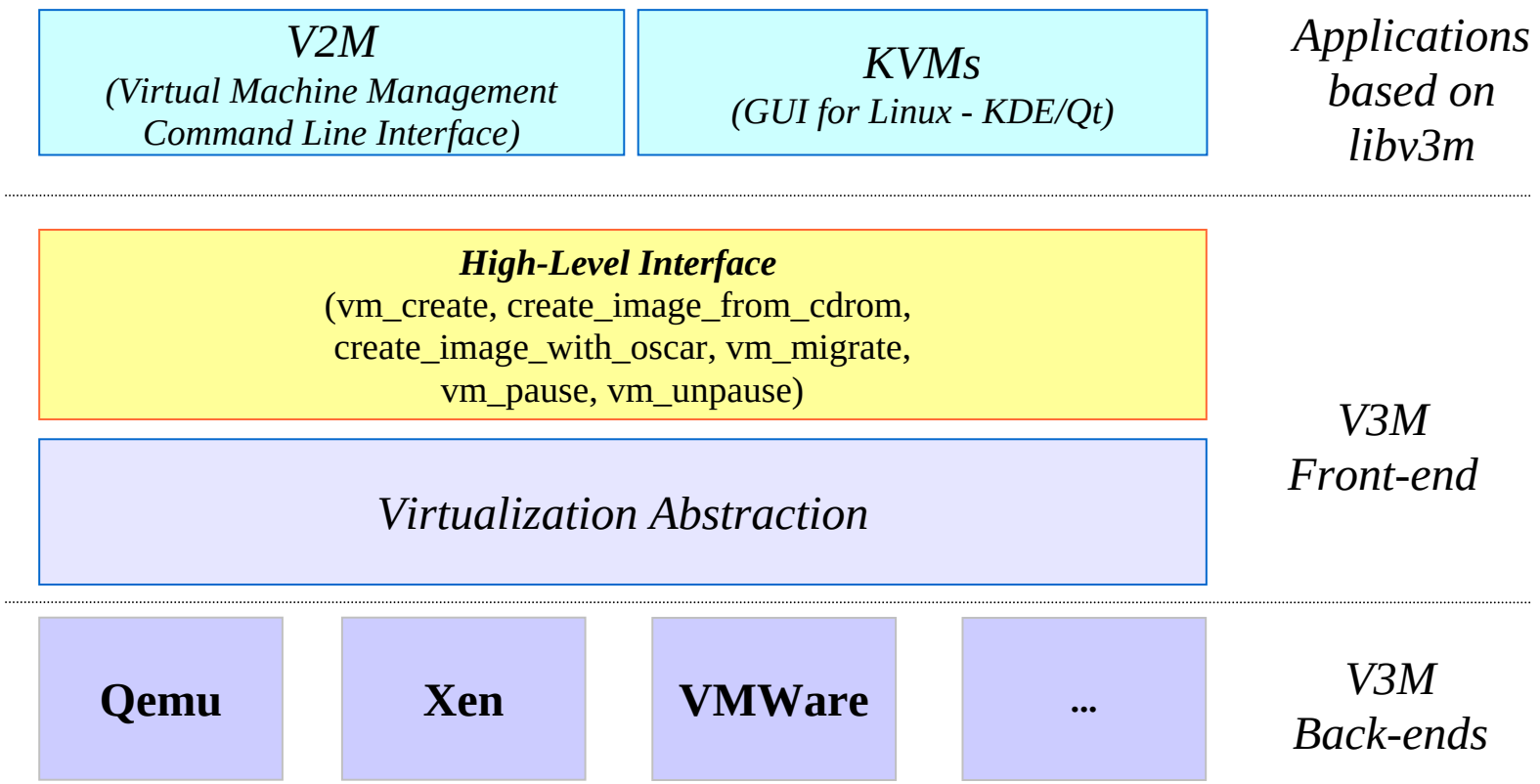
Profile Management

- Concept of profiles
  - for VMs, a profile is : memory, disk, OS, NICs, network configuration
  - for virtual distributed system, a profile is: a set of profiles of virtual machines



Abstraction of System Virtualization Solutions

- V3M (Virtual Machine Management and Monitoring)
- Allows users to specify VMs through a simple XML file (profile)



V3M – Management Capabilities

- Check the system
- Check the profile
- Create configuration scripts for VM management
- Provide simple interface for VM management: boot, image management, status
- Switch to a new virtualization solution: only change the type

OSCAR-V

OAK RIDGE
National Laboratory

Virtual Cluster Management

- Goals
  - Host OS management
  - Definition of images for VMs (which may deeply differ)
  - Deployment of VMs
  - Hide technical details associated to each virtualization solution
- OSCAR-V
  - OSCAR extension for the management of VMs
  - Integrates V3M and V2M

OSCAR-V

- OSCAR packages for Xen, Qemu
  - capability to create an image for Host OSes
    - minimal image
    - 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 Linux distributions supported by OSCAR: Mandriva, Suse, Debian, FC, RHEL, etc.
    - benefit of the default OSCAR configuration for compute nodes

VM Deployment on Demand

- OSCAR-V does not allow an automatic deployment of VMs at job submission time
- Integration of Dynamic Virtual Clusters
  - Moab extension for the deployment of VMs during job submission
  - Use OSCAR images, deployment based on DVC
  - Collaboration with ASU (Dan Santizone)

Operating System Research at ORNL:

System-level Virtualization

Presented by:

Christian Engelmann

engelmannc@ornl.gov

Systems Research Team

Computer Science Research Group
Computer Science and Mathematics Division
Oak Ridge National Laboratory
United States Department of Energy