Automatic Testing Tool for OSCAR Using System-level Virtualization

Geoffroy Vallée¹, Thomas Naughton¹, Wesley Bland¹ ², and Stephen L. Scott¹

¹Oak Ridge National Laboratory
Computer Science and Mathematics Division
Oak Ridge, TN  37831 USA

²Tennessee Technological University
Cookeville, TN  38505, USA
Overview

Synopsis: Enhance OSCAR by providing an automated testing tool for project developers/testers.

• Motivation
  – Why we need testing?
  – Why is it challenging?
• Background/Tools
  – Testing & Release procedures
  – OSCAR “CLI” & OSCAR-V
• Approach
  – Current status & Examples
OSCAR
Open Source Cluster Application Resources

Snapshot of best known methods for building, programming and using clusters.

Consortium of academic, research & industry members.
What does OSCAR do?

• Wizard based cluster software installation
  – Operating system
  – Cluster environment

• Automatically configures cluster components

• Increases consistency among cluster builds

• Reduces time to build / install a cluster

• Reduces need for expertise
Motivation

• Software testing
  – Ensure quality software
  – Identify new bugs or regression from changes

• Test automation
  – Reduce developer time/overhead
  – Reduce human error

• Supported distributions & architectures grow
  – Time consuming
  – Often just want sanity tests for devel tree
OSCAR Testing Challenges

• Distributed system
  – Remote machines & full control

• Installation & configuration
  – Full system access (destructive process)

• Several distributions / architectures
  – Large testing matrix

• Requisite functionality for automation
  – Non-interactive mode
    • OSCAR GUI Wizard
  – Access to distributed platforms
    • Clusters for many distributions/architectures
Automated OSCAR Testing

• Non-interactive mode
  – Recent Command-line Interface (CLI) enhancement
  – Provides basic scripting for all steps*
    * Note: Currently, the optional Configurator step isn’t supported by CLI due to technical issues, which may be resolved in latest OSCAR versions.

• Access to distributed platforms
  – Mature system-level virtualization solutions
  – Recent work in virtualization tools can be leveraged
    • libv3m/v2m for “oscar-v”
Background / Tools
System-level Virtualization

• First research in the domain, Goldberg73
  – type-I & type-II virtualization

• Xen created a new interest
  – performance (para-virtualization)
  – open source / Linux based

• Basic Terminology
  – Host OS: the OS running on physical machine
  – Guest OS: the OS running in a virtual machine

• Different approaches
  – full-virtualization: run an un-modified OS
  – para-virtualization: OS modifications for performance
  – emulation: host OS/guest OS architectures can differ
  – hardware support: Intel-VT, AMD-V
System-level Virtualization Solutions

• What to use in what case?
  – Type-I virtualization: performance
  – Type-II virtualization: development

• Examples used for OSCAR Testing
  – Xen
  – QEMU
  – VMWare*

* Note: not currently used in current automated work discussed in paper, but used in early manual testing.
Brief Comparison

• **Xen**: para-virtualization (type-I)
  – **Pro**: good performances for computation
  – **Con**: overhead for I/O, modifications to Linux kernel, growing complexity (driven by ASP market/different needs), not a full virtualization of the system

• **VMware (Workstation)**: full-virtualization (type-II)
  – **Pro**: mature, reasonable desktop performance
  – **Con**: still difficult to adapt (not open source), not really suitable for HPC

• **QEMU**: full-virtualization (type-II)
  – **Pro**: open source, performance similar to VMWare, supports a lot of architectures
  – **Con**: performance not suitable for HPC
Brief Comments for OSCAR Testing

• Why Xen or QEMU?
  – Modified host/guest OS?
    • Xen (para-virt) needs modified host/guest OS
      – Not as well suited for “laptop testing” or standard OSCAR tests
    • QEMU runs stock host/guest OS
      – Good for “laptop testing!” and unmodified OSCAR tests

  – Boot sequence
    • QEMU – full BIOS emulation & CDrom support
    • Xen (para-virt) – no direct BIOS or CDrom support
    • Xen (full-virt) – full BIOS emulation

  – Performance
    • Type-I (Xen) typically better performance than Type-II (QEMU)
VM Profile Management

• High-level abstraction
• 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
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>
```
V2M – Virtual Machine Management

**High-Level Interface**

(vm_create, create_image_from_cdrom, create_image_with_oscar, vm_migrate, vm_pause, vm_unpause)

**Virtualization Abstraction**

Qemu  
Xen  
VMWare  
...

Applications based on **libv3m**

V3M Front-end

V3M Back-ends

V2M  
(KVMs (GUI for Linux - KDE/Qt))

(Virtual Machine Management Command Line Interface)

Oak Ridge National Laboratory — U.S. Department of Energy
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
  - only change the “type”
OSCAR CLI

- OSCAR Command-Line Interface
- Text only menu-based interface to OSCAR
- Two modes of use
  - Interactive
  - Non-interactive
- Basic interactive invocation
  - requires single option on command line
    
    root# ./install_cluster --cli eth0

- For further details see OSCAR’07 talk, or OSCAR developer Wiki 😊
# CLI Usage

Usage: `install_cluster [OPTION] adapter`

Starts the OSCAR install process.

By default, `install_cluster` uses the Graphical mode.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--cli</td>
<td>Runs the program in command line mode.</td>
</tr>
<tr>
<td>--opkgselector file</td>
<td>Passes the file into the selector stage of the install. That stage will not ask for user input.</td>
</tr>
<tr>
<td>--buildimage file</td>
<td>Passes the file into the build stage of the install. That stage will not ask for user input.</td>
</tr>
<tr>
<td>--defineclients file</td>
<td>Passes the file into the define clients stage of the install. That stage will not ask for user input.</td>
</tr>
<tr>
<td>--networkclients file</td>
<td>Passes the file into the setup network stage of the install. That stage will not ask for user input.</td>
</tr>
<tr>
<td>--bootscript file</td>
<td>Passes the file to confirm the client nodes have booted with their new images into the main cli.</td>
</tr>
<tr>
<td>--help</td>
<td>Display this help and exit.</td>
</tr>
</tbody>
</table>
Node boot/build mechanism
(--bootscript file)

• This provides a generic hook for controlling the transition between node build and the final step (post_install) of OSCAR.

• After network setup completes, where typically you would manually boot nodes and wait until they complete before proceeding.

• Can be as simple or intelligent as you can script, just return zero (0) on success, or non-zero for error, and then will proceed accordingly.
Approach
Approach for OSCAR Testing

• Leverage virtualization
  – Xen and/or QEMU
  – V2M/libv3m

• Use non-interactive OSCAR interface
  – Command-line interface

• Fully automate / script
  – Download release (Subversion or Tarballs)
  – Setup environment
  – Launch/run install
Design

• Consistency
  – Limit/eliminate modifications to OSCAR
  – Test system similar to typical of OSCAR end-users
  – Duplicate/re-run tests – simple to setup test platform

• Virtualization
  – Unmodified host OS, just add “testing driver” script
  – Virtualize entire cluster for install/tests
    • head & compute nodes
  – Setup/manage virtual machines from basic “testing template”
Testing Driver

Basic algorithm for testing driver

1. Create base VM image(s) from “image template”
2. Boot virtual headnode
3. Checkout OSCAR in virtual headnode
4. Start OSCAR in virtual headnode
5. Boot virtual compute node(s)
6. Wait for node installation completion
7. Finalize OSCAR installation
8. Extract logs / output
9. Send logs/output to developers
VM Management via V2M

- The “testing driver” creates/uses VMs so must generate V2M profiles

Example V2M Profile for virtual headnode

```xml
<?xml version="1.0"?>
<!DOCTYPE profile PUBLIC "" "v3m_profile.dtd">
<profile>
  <name>oscar-headnode</name>
  <type>Qemu</type>
  <memory>256</memory>
  <image>
    /tmp/oscar-testing/oscar-headnode.img
  </image>
  <nic1>
    <type>TUN/TAP</type>
    <mac>00:01:02:03:04:05</mac>
  </nic1>
  <nic2>
    <type>VLAN</type>
    <mac>00:01:02:03:04:06</mac>
  </nic2>
</profile>
```
VM Management via V2M

- The “testing driver” creates/uses VMs so must generate V2M profiles

Example V2M Profile for virtual compute node

```xml
<?xml version="1.0"?>
<!DOCTYPE profile PUBLIC "" "v3m_profile.dtd">
<profile>
  <name>oscarnode1</name>
  <type>Qemu</type>
  <memory>128</memory>
  <image>
    /tmp/oscar-testing/oscarnode1.img
  </image>
  <nic1>
    <type>VLAN</type>
    <mac>00:01:02:03:06:06</mac>
  </nic1>
</profile>
```
Virtual Cluster

• Execution within virtual headnode
  – Two phases for virtual cluster
    • Deployment of compute nodes (boot/build)
    • Final configuration
  – Phase transition (wait barrier)
    • Monitor VM status
    • Use OSCAR CLI “hook”

• Compute Node Installation
  – Must “simulate” network boot for install
  – V2M boot with OSCAR regerated CD-rom (ISO) image
Other Testing Details

• Test driver uses (previously saved) CLI logs
  – Saved output from interactive CLI runs
  – Re-use the same CLI inputs or modify as needed

• Output is saved from the automated test
  – Standard OSCAR install log (oscarinstall.log)
  – Output from “test driver” itself (invocations, etc.)
Implementation / Current Status

• Test driver in Perl
  – Not yet integrated into OSCAR core code base

• Pre-requirements
  – System-level virtualization installed (Xen or QEMU)
  – V2M/libv3m tool installed
  – Assumes virtual headnode image setup with basic facilities
    • SSH installed (allowing connection as root with password)
    • Subversion client installed (to download OSCAR source)
    • Must manually setup ‘/tftpboot’ on host OS & in virtual headnode
      – Local repositories for OSCAR install
Implementation / Current Status (2)

• Network settings (assumptions of current prototype)
  – Virtual headnode with 2 NICs
    • Access to private VLAN
    • Access to public LAN (to include host OS)
  – If using online repository, host OS must provide NAT

• Node installation monitoring
  – Use CLI “hook” to monitor node installation
  – Use data from SystemImager rsyncd logs
    • /var/log/systemimager/rsyncd

• Limitations
  – Prototype uses V2M & has only been validated using x86/QEMU for the generated configuration files, etc.
  – Current tests have been limited to OSCAR on Debian
Related Work

• Camargos:oscar05 vservers
  – Earlier approach
  – Virtualization technique creates a resource “jail”
  – Several limitations:
    • Only applicable to VServers (no abstraction)
    • NFS not usable between virtual headnode/compute node
    • Can not “simulate” network boot (eliminate phases of OSCAR)
    • No OSCAR CLI available, no automated OSCAR testing

• Vallee:xhpc06 xen-oscar
  – Initial integration of OSCAR with (only) Xen
  – Works with standard OSCAR GUI
  – Only Xen & no drivers for automated testing
Conclusion

• OSCAR testing
  – Ever growing problem, very time consuming
  – Large testing matrix (distros/archs)

• Automated testing tool fosters
  – Improved development by checking code quality via testing
  – Speeds the release cycle by reducing basic testing time

• Current prototype
  – Uses non-interactive OSCAR CLI
  – Uses V2M/libv3m tools to abstract/manage virtualization
  – Used by OSCARonDebian effort
  – Validated using x86/QEMU
Questions?

OSCAR
http://oscar.openclustergroup.org/

OSCAR-V
http://www.csm.ornl.gov/srt/oscarv.html

V2M/libv3m
http://www.csm.ornl.gov/srt/v2m.html

ORNL's work was supported by the U.S. Department of Energy,
under Contract DE-AC05-00OR22725.