Automatic Testing Tool for OSCAR Using System-level Virtualization

Geoffroy Vallée[1], Thomas Naughton[1], Wesley Bland[1,2], and Stephen L. Scott[1] *

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

[2]Tennessee Technological University
Cookeville, TN 38505, USA

Abstract

To ensure quality, software development has to include testing mechanisms. OSCAR today supports several Linux distributions and several architectures. In such a context, the release cycle suffers of a important overhead created by the testing and stabilization phase. To address this issue, an approach is to implement a tool for automatic testing.

This paper presents such a tool which is based on the OSCAR command line interface. This tool, based on system-level virtualization techniques, creates a virtual cluster to perform the test. This approach has the benefit of not corrupting the system of the physical machine and guarantee that the environment used for testing has not been corrupted before testing.

1 Introduction

Software testing is mandatory for the release of quality software but it is also a difficult tasks especially for softwares targeting distributed systems. For instance, software such as OSCAR, a suite for the installation and the management of clusters which supports several Linux distributions and several architectures, are difficult to test because of its distributed nature but also because of the time needed for each supported Linux distribution/architecture.

Automatic testing is a common usage of software testing. However, automatic testing for software such as OSCAR implies implicit requirements: (i) be able to use the software in non-interactive mode, and (ii) have access to a distributed platform.

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

It is today possible to use OSCAR in non-interactive mode thanks to the command line interface. Moreover, the current OSCAR CLI is menu based which allows one to create a set of tests that are comparable to a usage via the Graphical User Interface (GUI), the initial user interface widely used by users. The OSCAR CLI also allows one to specify OSCAR inputs via files, enabling duplication of the cluster installation test.

The use of a real for automatic testing is most of the time not possible because of the cost that it implies (access to a cluster is most of the time charged to users and therefore reserved for application execution). Furthermore, system-level virtualization are today mature solutions and allow one to create a virtual cluster.

This document presents a tool for OSCAR automatic testing using system-level virtualization techniques. A *driver*, based on the OSCAR CLI, has been developed to drive the test without to have to modify OSCAR or the system used for testing.

The remainder of the paper includes, an introduction to system-level virtualization (Section 2) and design (Section 4) for the OSCAR CLI. Followed by a description of the current CLI implementation (Section 5) along with some comments for future work and concluding remarks (Section 6).

2 Introduction to System-level Virtualization

System-level virtualization is a pretty old topic of research in operating system. In 1973, Goldberg [4] defined the first classification of virtualization techniques (see Figure 1).

Type-I virtualization allows the execution of *virtual machines* (VM) directly on top of the hardware. This is done modifying the typical execution pattern of the operating sys-

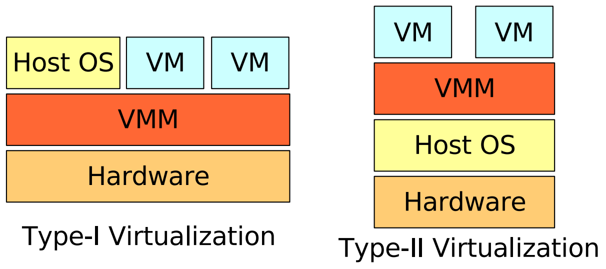

Figure 1. Classification of Virtualization Techniques

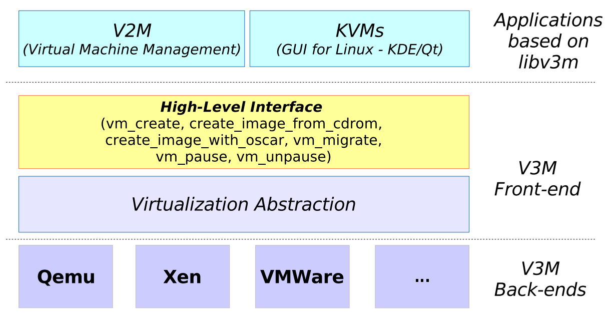
V2M
(Virtual Machine Management)
KVMs
(GUI for Linux - KDE/Qt)
Applications based on libv3m
High-Level Interface
(vm_create, create_image_from_cdrom, create_image_with_oscar, vm_migrate, vm_pause, vm_unpause)
Virtualization Abstraction
V3M Front-end
Qemu
Xen
VMWare
...
V3M Back-ends

Figure 2. V2M Architecture

tems on the processor. For instance, on the x86 architecture, VMs are running in the execution ring 1 instead of 0 [1]. But since the operating system of virtual machines are not running in a "privileged mode", an entity has to translate privileged processor instruction issued by the VMs' operating system. This entity is called *hypervisor*, or *Virtual Machine Monitor (VMM)*. The VMM is also in charge of the management of virtual machines and the scheduling of the existing VMs. The VMM does typically include device drivers; therefore, for hardware access, the VMM is coupled to an *host OS* which provides the driver and also provide an interface with the hypervisor for users.

Type-II virtualization does not modify the execution pattern of the hosted operating system: the virtual machine is typically running inside a process of an existing operating system, named the host OS. The host OS is in charge of the translation of all hardware access issued by virtual machines (memory access, disk access, network access).

Emulation is also a common solution. In that case, a full architecture is emulated on top of the hardware, most of time, using an approach similar of the type-II virtualization (the virtual machine is running within a process of the host OS). Therefore, simulation is slower than virtualization techniques described previously especially when the virtual machines are running directly on top of the hardware.

The goal of this document of not to provide a survey on virtualization and emulation techniques therefore we use in this document the generic term of virtualization, without distinction between virtualization (as defined by Goldberg) or emulation.

Therefore, even if these solutions have the same goal, provide a virtual machine, implementation and configuration differ for each of them. A tool has been developed at ORNL, called Virtual Machine Management (V2M), in order to provide an abstraction of these virtual machine. Thanks to this abstraction, users describe a VM using a high-level XML language; V2M dealing for the configuration and the management of the virtual machine based on the selected virtualization solution (see Figure 2). It is then very simple to switch from a virtualization solution to another, only the XML description has to be update, V2M modifying the configuration in order to effectively do the transition between the two different virtualization solutions.

3 Related Work

[3] presents an automatic testing tool based on VServer [8]. VServer is a Linux kernel extension that creates a *jail* in order to securely partition resources of the host OS. VServer is widely used for isolation of web servers on the same physical machine. This automatic testing tool has several limitations: (i) only VServer can be used, there is no abstraction of the virtualization solution, (ii) NFS cannot be used between the virtual headnode and the virtual compute nodes, (iii) it is not possible to "simulate" a network boot, this phase of the OSCAR installation cannot be fully tested, (iv) at the time this tool has been implemented, OSCAR did not provide a command line interface, therefore the test is in interactive mode and therefore does not provide a fully automated solution.

[7] presents the integration of Xen into OSCAR in order to support the creation of virtual clusters based on Xen. This solution allows users to "emulate" a cluster using Xen virtual machines, supporting network boot and a full installation via the OSCAR GUI. However, this solution does not support other virtualization solution and no drivers are available to perform the tests.

4 Design

In order to guarantee the validity of the automatic testing, the testing tool has to guarantee that: (i) OSCAR is not modified, and (ii) the system on within is tested OSCAR is similar to the typical system used by users want they use OSCAR. In order to be able to duplicate and rerun tests, we also need to be able to setup easily a new testing platform.

Virtualization is therefore a good solution for automatic testing: (i) the host OS is not modified, a *driver* only need to be implemented for the automatic management of a virtual cluster, used for the tests; (ii) within the virtual headnode, OSCAR can be executed without any modifications, a virtual machine provides exactly the same system interface than on a physical machine; and (iii) virtual machines are based on a image it is therefore simple to create an *image template*, basic system that is used for the creation of the virtual cluster, duplicated for every testing, guaranteeing a not corrupted system.

4.1 Testing Driver

The testing driver will drive the testing from the host OS. The driver executes the algorithm listed in Listing 1.

Listing 1. Driver Algorithm

```
copy_vm_image (image_template);
boot_virtual_headnode ();
checkout_oscar_within_virtual_headnode();
start_oscar_within_virtual_headnode ();
bootup_compute_nodes();
wait_for_node_installation_completion();
finish_oscar_installation();
grab_oscar_logs();
send_logs_to_developers();
```

4.1.1 Management of the Virtual Machines

The testing driver has to create a virtual cluster. For that, the driver generates automatically V2M profiles for virtual machines. These profiles are XML files that describe a virtual machine. Listing 2 presents a profile example for a virtual headnode with two networks cards (one connected with the host OS (TUN/TAP) and another for the creation of a virtual network used by the virtual cluster), having 256MB of memory, based on the VM image `/tmp/oscar-testing/oscar-headnode.img`, using QEMU.

Listing 2. Example V2M Profile

```
<?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>
```

Virtual Headnode Management The virtual headnode has to be based on a Linux distribution supported by OSCAR and have to be similar to the typical Linux operating system used by users before the installation and the use of OSCAR.

For that, a *template image* has to be created. This template will be copied for the automatic tests, the template is therefore not corrupted by a previous OSCAR execution and may be reused. This template provides all software needed for the use of OSCAR, as well as the binary package repositories needed by OSCAR (cf. OSCAR documentations for more details [6]).

When the virtual headnode is created, it is very simple to drive the tests from the host OS using *Secure Shell* (ssh).

4.1.2 OSCAR Execution Within the Virtual Headnode

There are two broad phases related to the virtual cluster: (i) deployment of the compute nodes, and (ii) final configuration of the cluster. Before proceeding to the cluster configuration stage, all the nodes have to be installed and booted in order to accept the final configuration commands from the headnode. The separation of these two steps lead to the need to monitor the cluster state when using OSCAR is non-interactive mode. Typically, one has to poll monitoring information from OSCAR, allowing one to know when the cluster is ready to be configured.

The OSCAR CLI provides a hook for the customization of this monitoring step. This hook is used for automatic testing, polling monitoring information: the script polls monitoring information until all the compute nodes are installed and rebooted. When compute nodes are back online, the script exists and the testing driver continues its execution.

4.1.3 Compute Nodes Installation

In order to "simulate" compute nodes installation, we need to boot up a virtual compute node and simulate a network installation. For that, we use the capability of V2M to perform a virtual boot based on a bootable CDROM image. In fact, we use the bootable CDROM that OSCAR generates for systems that do not support PXE. Listing 3 presents a profile example for such a compute node.

Listing 3. Example Node Profile

```
<?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>
```

4.1.4 Testing Parameters

The driver behavior is defined by input files used by the CLI: the CLI is menu based and provide the capability to define OSCAR parameters within files that are then used by OSCAR [2]

A set of default values have been defined and are used by the driver. However, it is possible to change these inputs and therefore modify the tests.

The driver also create automatically configuration files used to run the tests. These files are defined at the beginning of the driver code, easing their modifications, but users should not need to modify them since they are LSB [5] compliant and therefore useable on all Linux distributions.

5 Implementation

Currently the testing driver is implemented in Perl, like most of the OSCAR code. The current prototype has not been integrated into the OSCAR code.

5.1 Pre-requirements

The first requirement is to have a system-level virtualization solution installed, as well as V2M; V2M currently supporting QEMU and Xen. The selected virtualization solution and V2M has to be setup and should allow the creation of a virtual machine with one connection with the host OS (also called *TUN/TAP*).

The current prototype also assumes that the image used for the headnode is setup with few capabilities. The virtual machine have to have the *secure shell* (ssh) tool installed and allow connection as root with password, as well as subversion (used to get the OSCAR source). This is mandatory in order to be able to drive the test from the host OS in a non-interactive mode.

The current OSCAR implementation requires that users setup `/tftpboot` on the headnode. This directory is setup to provide a local binary packages repository for the targeted Linux distribution as well as local repository for OSCAR binary packages. The user also have to setup such repositories within the virtual headnode, the driver does not automatically set it up since it is not part of the standard OSCAR usage.

The current prototype also requires a specific network configuration. First of all, the virtual headnode have to have two network interface setup: one in order to access the private network (virtual LAN) and one to access the public network (which includes communication with the host OS). Moreover, if the virtual headnode needs an internet access (for instance if the virtual headnode is setup to use an online binary package repository), the host OS has to provide a network address translation translation to route packets from/to the virtual cluster.

5.2 Virtual Cluster Installation Monitoring

In order to know if compute nodes are installed and then set them up, the virtual cluster status has to be monitored. This monitoring mechanism is based on the OSCAR CLI hook presented in Section 4.1.3.

The script polls monitoring information from OSCAR, especially logs from `rsyncd` and `systemimager` (system logs available in `/var/logs/systemimager/`).

5.3 Current Limitations

The current prototype is based on V2M which is designed to abstract the system-level virtualization technology used. Therefore, the virtual cluster is defined via XML files, and depending on the virtualization solution the user wants to use, V2M generates the associated configuration file and then allows the management of the virtual cluster. However, the current prototype has been validated only using QEMU virtual machines, emulating the x86 architecture.

The current prototype has been tested using OSCAR on Debian, further tests have to be done using others Linux distributions.

6 Conclusion and Future Work

This document presents a tool for automatic testing of the OSCAR suite. This tool allows developer to: (i) improve developments, checking the code quality by testing, (ii) speed up the release cycle, decreasing the testing/stabilization phase.

The testing tool is based on the OSCAR CLI and take benefit of the menu based approach: a preset configuration has been created for testing and is used to automatically run tests. Moreover, the menu based approach allows developers to easily modify inputs, creating new tests.

The automatic testing tool is today used daily in the project OSCARonDebian.

The current implementation of the testing tool is based on V2M an abstraction tool for system-level virtualization which is designed to ease the transition from a virtualization to another. However, the current implementation has been validated only using QEMU, but future tests are planed using Xen.

References

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating System s Principles (SOSP19)*, pages 164–177. ACM Press, 2003.

[2] W. Bland, T. Naughton, G. Vallée, and S. L. Scott. Design and Implementation of a Menu Based OSCAR Command Line Interface. In *Submitted to the 5th OSCAR Symposium, held with 21th International Symposium on High Performance Computing Systems and Applications (HPCS)*, 2007.

[3] F. L. Camargos and B. des Ligneris. Automated OSCAR Testing with Linux-VServers. In *Proceedings of the 19th International Symposium on High Performance Computing Systems and Applications (HPCS)*, pages 347–352, Guelph, Ontario, Canada, May 15-18 2005. IEEE Computer Society. Session track: 3rd Annual OSCAR Symposium.

[4] R. P. Goldberg. Architecture of virtual machines. In *Proceedings of the workshop on virtual computer systems*, pages 74–112, Cambridge, Massachusetts, United States, 1973. ACM Press.

[5] Linux Standard Base (LSB). http://www.linux-foundation.org/en/LSB.

[6] OSCAR Documentation. http://svn.oscar.openclustergroup.org/trac/oscar/.

[7] G. Vallée and S. L. Scott. Xen-oscar for cluster virtualization. In *Proceedings of ISPA Workshops: Workshop on Xen in HPC Cluster and Grid Computing Environments (XHPC)*, pages 487–498, 2006.

[8] Linux-vserver.org. http://linux-vserver.org/.