

OSCAR testing infrastructure

Benoît des Ligneris
Revolution Linux,
145 rue Sauvé,
Sherbrooke, Québec, Canada, J1L 1L6

Abstract—This document describes the objectives and gives a precise description of an automatic testing infrastructure for OSCAR. The different technologies available, as well the layout of the testing infrastructure will be detailed. The benefits of such a tool for the OSCAR project will also be discussed.

CONTENTS

I	Objective and definition	1
I-A	Rationale	1
I-B	OSCAR workflow modification	1
II	Available technologies	2
II-A	Virtualization of computers .	2
II-A.1	Vmware Workstation	2
II-A.2	plex86	2
II-A.3	Bochs	2
II-B	Linux inside Linux	2
II-B.1	The chroot method	2
II-B.2	The Linux Vservers	2
II-B.3	User Mode Linux	3
II-C	Comparisons of the available methods and choice of the best one	3
III	Proposed implementation	3
III-A	Topology of the virtual computers	3
III-B	Structure of the virtual hard-disk on the master-node . . .	3
III-C	How to test ?	4
IV	Test plan : what to test ?	4
IV-A	OSCAR build	4
IV-B	OSCAR installation	4

IV-B.1	Default installation : core packages only	4
IV-B.2	Installation of each package individually . . .	5
IV-B.3	Installation of package sets . . .	5
IV-C	OSCAR maintenance	5
IV-D	Distribution maintenance . .	5
IV-E	OSCAR update	5

V	Conclusion	5
	References	5

I. OBJECTIVE AND DEFINITION

The OSCAR[1] Automated Suite Installation and Simulation (OASIS) is a tool for OSCAR developers, by OSCAR developers. It has one main goal : to provide a way to automatically test an OSCAR installation process on different distributions and on different architectures.

A. Rationale

As the number of supported combinations {distribution, version, architecture} increases, the release mechanism is increasingly more complex. The testing phase can be very long, for instance, for the IA32 architecture :

- 2 RedHat versions
- 2 Mandrake versions
- 2 Fedora versions

At the time of this writing, there is only one additional architecture (IA64) with two distributions/versions :

- 1 RedHat 7.3

- 1 Debian unstable

In the near future, support for the x86_64 will be required with several distributions (the same as is already supported in other architectures) :

- 3 : RedHat, Mandrake, Debian

With these constraints and without an automated testing tool the *release early/release often* mechanism becomes totally impractical, especially because of some boundary effects : changes in one given combination of {distribution,version,architecture} can affect one or all of the other combinations of {distribution, version, architecture}.

If every developer tested against all supported distributions/versions then development would be extremely slow. Even if such a drastic rule is not applied, if the testing time for a given trio is constant, the testing time for each OSCAR release is proportional to n , the number of trios {distribution,version,architecture} supported : this is a $O(n)$ problem.

Once the OASIS tool is well established, it will run daily on the development version (the head of the CVS repository) on all distributions and architectures supported by OSCAR. These tests will detect defects early in the development process (no later than 24 hours after a faulty commit into the CVS tree). With the OASIS tool, the problem of pre-release testing is now a $O(1)$ problem : all tests are achieved in a 24 hour period. Moreover, it is even more interesting for the development of OSCAR because no developer time is used by OSCAR in order to achieve this result !

B. OSCAR workflow modification

Once the OASIS structure is in place, the daily results will be included in the workflow of the OSCAR development process which can solve some "broken CVS" problems. Ideally, a new role will be created : testing manager, whose job will be to make sure that automatic testing is up to date with the development tree (maintenance of the automatic testing in sync with the new OSCAR feature), and that tests are always successful.

Each time a test does not work, the testing manager will first identify the faulty commit, then ask for corrections, or rollback the commit so that the automatic testing is functional again.

II. AVAILABLE TECHNOLOGIES

In this section, we will discuss the available technologies that can be used for OASIS. It is important to note that several major opensource projects enjoy daily regression testing : kernel, gcc, etc. In this context, developing an automatic regression testing for OSCAR can also be seen as a necessity to increase the quality of OSCAR.

We will not discuss here the tools that can be used to do automatic regression testing, this will be discussed in a the next section (proposed implementation).

A. Virtualization of computers

The vmware[2] workstation program and it's open source equivalent, plex86 [4] or Bochs[3] (which achieve the same goal but not with the same technics) emulate an x86 CPU. Some of these are mature enough to provide a virtualization of resources allowing one computer to host several alien operating systems and to share resources (file system, etc.) between these virtual computers.

1) *Vmware Workstation*: From the VMware[2] web site, *VMware Workstation is a powerful virtual machine software for the desktop. VMware Workstation runs multiple operating systems, including Microsoft Windows, Linux, and Novell NetWare, simultaneously on a single PC in fully networked, portable virtual machines.*

VMware is already in used by most OSCAR developers in order to diminish development time. Real testing occurs only when it is time to do a release.

2) *plex86*: From the plex86[4] website, plex86 is a very lightweight Virtual Machine (VM) for running Linux/x86. It uses the same VMware logic but is restricted only to the Linux OS (native OS as well as guest OS). At this time, it is needed to recompile the kernel on your guest OS and this seems very impractical because OSCAR uses standard distribution kernels.

3) *Bochs*: From the Bochs[3] web site : *Bochs is a highly portable open source IA-32 (x86) PC emulator written in C++, that runs on most popular platforms. It includes emulation of the Intel x86 CPU, common I/O devices, and a custom BIOS. Currently, Bochs can be compiled to emulate a 386, 486, Pentium, Pentium Pro or AMD64 CPU,*

including optional MMX, SSE, SSE2 and 3DNow instructions.

The performance of bochs is does not compare to VMWare or plex86 mainly because it emulates the CPU instead of using the native instruction set of the IA-32 CPUs. There is no locking mechanism for the disks.

B. Linux inside Linux

In this section we will discuss three methods in use in order to have a Linux environment inside a Linux environment.

1) *The chroot method:* As its name suggests, the chroot command allows any command inside a Linux OS to be run within a new root. This is used heavily for security reasons (protect the "true" root of a server) or in order to automatically compute RPM[5] dependencies.

While it can be useful to test certain components of OSCAR it can not be used to simulate a complex clustering environment (network adapters, private network, etc...).

However, at the time of this writing, this is the only program that is truly architecture independent : chroot works on any architecture. I feel, however that the amount of work needed to reproduce an OSCAR installation inside a chroot environment is too lengthy.

2) *The Linux Vservers:* From the Linux-Vserver web site[6] : *Linux-VServer allows you to create virtual private servers and security contexts which operate like a normal Linux server, but allow many independent servers to be run simultaneously in one box at full speed.*

The Linux-Vserver project consists of a patch (on the real OS) and, once done virtual servers can be created. The virtual servers share CPU resources from the real OS and no emulation occurs. This is a very fast system. Several user tools exist to control the vservers (start, stop, etc...).

3) *User Mode Linux:* From the UML-Linux web site[7] : *User-Mode Linux is a safe, secure way of running Linux versions and Linux processes. Run buggy software, experiment with new Linux kernels or distributions, and poke around in the internals of Linux, all without risking your main Linux setup.*

User-Mode Linux gives you a virtual machine that may have more hardware and software virtual

resources than your actual, physical computer.

This is a similar project to the Linux Vservers project. It provides the same level of functionalities but the emulation is very slow at this time.

C. Comparisons of the available methods and choice of the best one

Because VMWare testing is already widely in use by OSCAR developers this will be the best choice at this time. By choosing VMWare, the automated testing can be seen as a complete standardization of the testing method : the OSCAR community will be able to use the testing case manually (if needed) inside their own VMWare.

However, it is important to note that the functionality of the VMWare testing is completely similar to the one from bochs, VServers or UML-Linux. As such, and for licensing purposes, it would be very interesting to use one of these technologies, especially if the number of VMWare licences is limited. Moreover, it will allow the development of OASIS to be done by anybody, and not only by VMWare licenced holders.

As noted above, the only architecture independent technical solution is chroot.

III. PROPOSED IMPLEMENTATION

The proposed implantation is independent of the tool used to provide the emulation. For practical reasons, and because the user has extended VMWare knowledge, we will use VMWare terminology. However, the same outcome can achieved with another tool (the author is currently experimenting with the Linux-vserver project with very promising results).

One of the main drawbacks of the proposed solutions is the lack of architecture support : only the IA32 architecture is virtualized (or emulated). However, it is expected that the new 64 bit architecture that is of major importance for OSCAR (IA64 and x86_64) will be supported as it become more popular.

A. Topology of the virtual computers

The architecture of the virtual network of computers used by OASIS is presented in Figure 1. The real computer hosting all the others is called "Linux Box". Inside this box, three virtual computers called "Master", "Node01" and "Node02" are running

using, for instance, VMWare. The three virtual hosts are linked by a private virtual network. The virtual computer called "Master" has access to the internet using the internet connection of the Linux box.

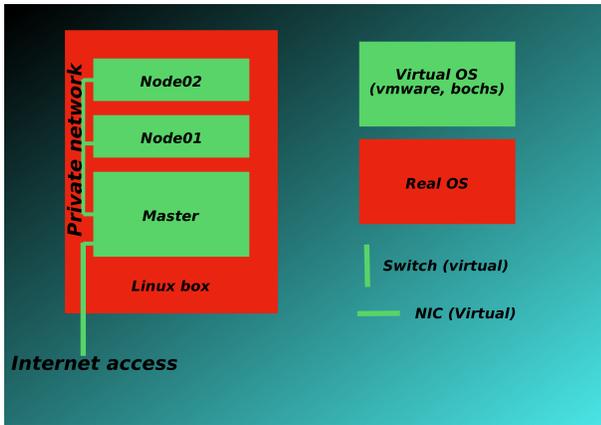


Fig. 1

TOPOLOGY OF THE VIRTUAL COMPUTERS

This architecture is the smallest one that will allow us to test several key functions of OSCAR as well as the installation itself, which requires only one Master server, and one node. With the proposed setting, we can test the following functions :

- add/remove nodes
- heterogeneous cluster : two different images on node01 and node02
- heterogeneous hardware : for instance IDE disk on node01, SCSI disk on node 02

All this will be detailed in the sectionIV.

B. Structure of the virtual hard-disk on the master-node

The hard-disk structure of the virtual nodes (Node01 and Node02) will not be presented here : nodes are completely reinstalled during a regular OSCAR installation including formatting of the hard disk.

For the server however, it is important to always start from a well known starting point in order to ensure good reproducibility. VMWare allows us to lock a virtual disk : this achieves exactly what is desired. All changes to this virtual hard disk will be lost when the virtual computer is rebooted or stopped.

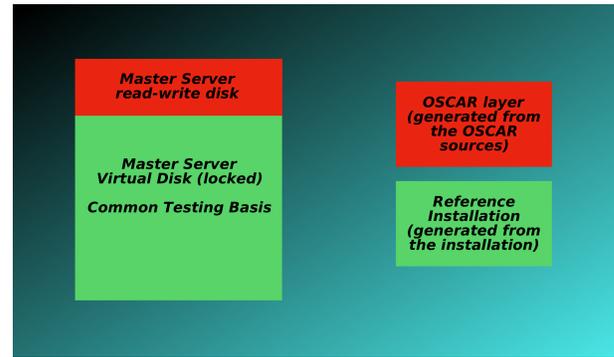


Fig. 2

SETTING OF THE VIRTUAL DISK (ON THE OSCAR VIRTUAL SERVER) AS USED BY OASIS

The basic disk image consists of one "regular" installation of the distribution/version to be tested as well as the necessary RPMs placed inside the /tftpboot/rpm directory. This is done only once, and once this is done, the content of the disk is locked and shared so that any OSCAR developer can obtain the basic disk image.

The /opt/oscar directory will be copied from a fresh checkout done at a given time (let's assume midnight EST everyday) directly from the Linux box disk.

C. How to test ?

At the time of this writing, OSCAR is completely dependant on its Graphical User Interface (GUI) in order to work properly. It is certainly easier to test a Command Line Interface (CLI) than a GUI because of the increased capacity to script the actions done with a CLI.

OSCAR is using the X-windowing system and several tools exist to generate arbitrary X events to any given application [8], [9] so that there is no real technical showstopper.

However, the separation of functionality and interface is now well established in OSCAR, which means that the development of a CLI is a feature that can be very useful in order to simplify the automatic testing [10].

IV. TEST PLAN : WHAT TO TEST ?

In this section we will list the functionality that has to be tested. Of course, this is an iterative pro-

cess that will be refined while others will be added but this subset is interesting because it provides minimal testing for an OSCAR release.

A. OSCAR build

The first and easiest function to test is actually the OSCAR build process because it is CLI based. The goal of the build process is to generate several kinds of tarballs (including documentation) that can then be distributed.

The test will simply build a tarball and look for errors. The error log as well as the complete build log will be available on a public webpage.

The objective of this test is to make sure that the build program is in sync with the development version of OSCAR.

B. OSCAR installation

This is the most important task, and the one that OSCAR was originally designed for, as well as the most complex one. The starting point here is the locked image of the virtual master node. A bit of human interaction is needed for the image creation as described in section III-B.

While no CLI is available, the X events will be recorded and then replayed accordingly. The error and regular flow will be stored for future reference. The time needed to perform the installation will also be recorded. Both of recordings will be available on a public web site. Each time the GUI changes, adaptations to OASIS will be necessary in order to reflect these changes.

1) *Default installation : core packages only:*

While this is not an atomic step (several steps make the installation of the cluster possible) this is a very interesting test for OSCAR developers as this tests the core packages of OSCAR for installation.

These packages are a prerequisite for any additional tools to work. Any failure at this test should be corrected as soon as possible in order to maintain a functional development tree.

2) *Installation of each package individually:* If the default installation is successful, then all the packages will be tested individually (including their dependencies). This will be possible only if time permits, or if there is enough dedicated testing computers.

3) *Installation of package sets:* While unit testing is very important, it does not correspond in practice to a real use case. In general, users will install package sets. This can be simulated also : several use cases (MPI, PVM, scheduler, etc...) will be created in order to reflect this.

OASIS will test these use cases in order to ensure that the packages are working well together.

C. OSCAR maintenance

The maintenance operations are the following : add/remove node, and add/remove packages. Those scenarios will be tested in this regard.

D. Distribution maintenance

The distribution vendor regularly publishes security updates, especially for servers. Because OSCAR uses a lot of servers, the security update of one distribution vendor can affect the global OSCAR functionality.

In order to test this, the distribution updates will be installed (from a local copy on the testing computer) and then all OSCAR installations and maintenance will be tested.

This kind of test can be generalize to a non-development version of OSCAR : this will allow the OSCAR community to publish bugfixes necessary to operate older clusters.

E. OSCAR update

While there is no official way for updating OSCAR at this time, the fact that all the relevant information is in ODA (the OSCAR Database), it should not be difficult to update the newest OSCAR version.

OASIS can be used to test these major updates.

V. CONCLUSION

In this paper, we briefly presented the architecture of a completely automated OSCAR testing suite call OASIS. We analyzed the existing tools that can be used to test OSCAR and made some recommendations regarding these technologies. Then we detailed the basic structure for the OASIS tool as well as the requirements on the hardware, and software side.

We have established that the testing time for each OSCAR release is proportional to the number of combinations of supported {distribution, version,

architecture} : this is a $O(n)$ problem. The development of OASIS will transform this into a $O(1)$ problem (24 hours).

OASIS will ensure better reproducibility of an OSCAR installation and development as the initial state of the OSCAR installation will be well known and shared among the developers.

REFERENCES

- [1] OSCAR : Open Source Cluster Application Ressources,
<http://oscar.openclustergroup.org/> 30th of april 2004
- [2] Vmware,
http://www.vmware.com/products/desktop/ws_faqs.html 30th of april 2004
- [3] Bochs,
<http://bochs.sourceforge.net/> 30th of april 2004
- [4] plex86 ,
<http://www.plex86.org/> 30th of april 2004
- [5] Stephan little build dameon,
<http://qa.mandrakesoft.com/twiki/bin/view/Main/S1Bd> 30th of april 2004
- [6] The Linux-VServer project,
<http://www.linux-vserver.org/> 30th of april 2004
- [7] The User-mode Linux Kernel Home Page,
<http://user-mode-linux.sourceforge.net/> 30th of april 2004
- [8] The Android GUI Testing Tool,
<http://www.wildopensource.com/larry-projects/android.html> 30th of april 2004
- [9] Xnee,
<http://www.gnu.org/software/xnee/> 30th of april 2004
- [10] OSCAR CLI : a Command Line Interface for OSCAR, Benoît des Ligneris and Fernando Laudaes, to be published.