Design and Implementation of a Menu Based OSCAR Command Line Interface

*Wesley Bland*\(^1\)\(^2\), *Thomas Naughton*\(^1\), *Geoffroy Vallée*\(^1\), 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*
Introduction

- OSCAR Overview
- Background
- Motivation
- Design / Implementation
- Usage
- Future Work
- Conclusion
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
OSCAR Overview

• Framework for cluster management
  – simplifies installation, configuration and operation
  – reduces time/learning curve for cluster build
    • requires: pre-installed headnode w. supported Linux distribution
    • thereafter: wizard guides user thru setup/install of entire cluster

• Package-based framework
  – Content: Software + Configuration, Tests, Docs
  – Types:
    • Core: SIS, C3, Switcher, ODA, OPD, APItest, Support Libs
    • Non-core: selected & third-party (PVM, LAM/MPI, Toque/Maui,...)
  – Access: repositories accessible via OPD/OPDer
OSCAR Design Goals

• **Reduce overhead for cluster management**
  – Keep the interface simple
  – Provide basic operations of cluster software & node administration
  – Enable others to re-use and extend system – deployment tool

• **Leverage “best practices” whenever possible**
  – Native package systems
  – Existing distributions
  – Management, system and applications

• **Extensibility for new Software and Projects**
  – Modular meta-package system / API – “OSCAR Packages”
  – Keep it simple for package authors
  – Open Source to foster reuse and community participation
  – Fosters “spin-offs” to reuse OSCAR framework
Terminology

• OSCAR Package Interface(s)
  – Script/hooks used by oscar package
  – Not what we're talking about.

• User Interface
  – Graphical User Interface (GUI)
  – Command Line Interface (CLI)
Background

• CLI (des Ligneris & Camargos, 2004)
  – Separate command line tools – mirror GUI
  – Work never merged with main devel repository
  – More complex usage scenarios
    • ordering/sequence operations, learning curve

• MetaMenu (Squyres, 2003)
  – State machine with menus at each stage
  – Separate presentation layer, e.g., ncurses, Qt, text
  – Premise: Linux installers (e.g., Anaconda)
    • Engine that does X window setup in ncurses, then use X ...
  – Preliminary design & initial code
  – Change in project direction and participation
User Interfaces

• GUI: Graphical User Interface
  – Good for inexperienced users
  – Lower overhead / no memorization
  – Cumbersome for advanced usage
  – Resource intensive
    • Problem for low bandwidth connections

• CLI: Command Line Interface
  – Good for advanced users (developers, SysAdmins)
  – Increase overhead / memorize commands
  – Flexible for advanced usage
  – Resource friendly
    • Acceptable for low bandwidth connections
Text-based Menus

• CLI + Menus
  – Reduce complexity by using menu based approach
  – Maintain some GUI features via menus
  – More conducive to scripting
    • All input/output text based

• Modes of operation
  – Interactive
    • Present menus and accept responses via STDIN
  – Non-interactive
    • Read answers to menus from saved results (files)
Motivation

• Automation of OSCAR
  – Testing by developers
  – Reporting configuration/setup for diagnostics
  – Easily recreate/duplicate environments
  – Script custom deployments

• Reduce base (core) OSCAR requirements
  – Eliminate need for X environment*
    *NOTE: currently not removed all Qt/Tk, etc dependencies from other parts of OSCAR.
  – Better installation/management remotely
OSCAR Testing

• Supported distributions & architectures grow!
  – Time consuming
  – Often just want sanity tests for devel tree
  – Automate using CLI and saved input

• Automation using Virtual Machines
  – Qemu
  – Xen
  – VMWare
  – etc...
Design

• Mirror GUI using text menus
  – Follow all steps*
    *NOTE: One exception – currently do not process the optional Configurator step. Ironically this step is to provide user-input for customization of OSCAR default settings. Problems processing input format in text only environment.
  – Maintain sequence/order of OSCAR operations
    • OSCAR phases, OPKG API script invocation, image build, network setup & node definition, etc.

• Support non-interactive mode / full automation
  – Read saved states from input files
  – Provide flexible mechanism to boot/build node

• Strive to maintain common code for GUI / CLI

Aside: Reviewer comment similar idea as AIX’s smit tool.
Implementation

• Written in Perl

• Leverage existing OSCAR libraries when possible

• Support full installation
  – interactive & non-interactive modes  (see also: Release Notes)
  – save input for later re-use

• Basic interactive invocation
  – requires single option on command line

  root# ./install_cluster  --cli eth0

• Advanced options for skipping steps
  – useful for developers/testers
  – NOTE: Currently must pass to 'main_cli' directly, not exposed via install_cluster (basic invocation).
Release Notes

• CLI kept fairly isolated from other code
  – Exception: “Network Setup” GUI → MAC.pm
  – MAC.pm Tightly coupled, some code duplication

• CLI can only read MAC addresses from file
  – GUI allows for “from file” and “from network”

• Configurator not supported
  – Due to issues with input/output
  – Work to move data into database will aid
CLI Usage

Usage: install_cluster [OPTION] adapter
Starts the OSCAR install process.
By default, install_cluster uses the Graphical mode.

--cli
Runs the program in command line mode.

--opkgselector file
Passes the file into the selector stage of the install. That stage will not ask for user input.

--buildimage file
Passes the file into the build stage of the install. That stage will not ask for user input.

--defineclients file
Passes the file into the define clients stage of the install. That stage will not ask for user input.

--networkclients file
Passes the file into the setup network stage of the install. That stage will not ask for user input.

--bootscript file
Passes the file to confirm the client nodes have booted with their new images into the main cli.

--help
Display this help and exit.
“selector.4948.clilog”
(--opkgselector FILE)

#######################################################################
# select <packageName> - Select a package to be installed
#   -q - Quiet mode: Don't print out verbose dialog
#unselect <packageName> - Unselect a package to prevent it from being installed
#   -q - Quiet mode: Don't print out verbose dialog
#     list <class> - Lists the packages and their installation status
#           class, and version number
#      file <filename> - Reads in commands from a file
#       help - Prints this message
#     quit/exit - Quits the selector
#######################################################################
quit
“build.4948.clilog”
(--buildimage file)

# Select one

# 1) Image name: oscarimage
# 2) Package file: /root/trunk/oscarsamples/fc-4-i386.rpm
# 3) Distro: fedora-4-i386
# 4) Package Repositories: /tftpboot/oscar/common-rpms,/tftpboot/oscar/fc-4-i386,
   /tftpboot/distro/fedora-4-i386
# 5) Disk Partition File: /root/trunk/oscarsamples/ide.disk
# 6) IP Assignment Method: static
# 7) Post install action: reboot
# 8) Build Image
# 9) Quit

# Select one

# 1) Image name: oscarimage
# 2) Package file: /root/trunk/oscarsamples/fc-4-i386.rpm
# 3) Distro: fedora-4-i386
# 4) Package Repositories: /tftpboot/oscar/common-rpms,/tftpboot/oscar/fc-4-i386,
   /tftpboot/distro/fedora-4-i386
# 5) Disk Partition File: /root/trunk/oscarsamples/ide.disk
# 6) IP Assignment Method: static
# 7) Post install action: reboot
# 8) Build Image
# 9) Quit
“define.4948.clilog”
(--defineclients file)

Select one

#1) Image Name: oscarimage
#2) Domain Name: oscardomain
#3) Base Name: oscarnode
#4) Number of Hosts: 1
#5) Starting Number: 1
#6) Padding: 0
#7) Starting IP: 192.168.0.2
#8) Subnet Mask: 255.255.255.0
#9) Default Gateway: 192.168.0.1
#10) Add Clients
#11) Quit
"mac.4948.cli.log"  
(--networkclients file)
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.
Future Work

• Add support for Configurator
  – Look at latest version that uses the database

• Reduce coupling in MAC.pm
  – Improve code reuse between CLI/GUI
  – Eliminate some code duplication

• Improve OSCAR dependency/packaging
  – Eliminate need for Qt/Tk if only want minimalistic CLI and no X window

  NOTE: Not really a CLI issue, more of an OSCAR core / packaging issue.
Conclusion

• User Interfaces
  – GUI is good: general users
  – CLI is good: advanced/experienced users
  – Text-based menus provide mix of CLI & GUI

• CLI reduces overhead
  – Lower bandwidth / resource consumption

• Testing
  – Ever growing problem, very time consuming
  – Automated testing using non-interactive CLI
  – Very powerful when combined with virtual machines
  – See the “oscar-testing” talk at OSCAR’07! 😊
Questions?

OSCAR Homepage

http://oscar.openclustergroup.org/

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