OSCAR Clusters

John Mugler, Thomas Naughton[*] and Stephen L. Scott[†]
Oak Ridge National Laboratory, Oak Ridge, TN

Brian Barrett[‡] Andrew Lumsdaine and Jeffrey M. Squyres[§]
Indiana University, Bloomington, IN

Benoît des Ligneris, Francis Giraldeau[¶]
Université de Sherbrooke, Québec, Canada

Chokchai Leangsuksun[‖]
Louisiana Tech University, Ruston, LA

Abstract

The Open Source Cluster Application Resources (OS-CAR) is a cluster software stack providing a complete infrastructure for cluster computing. The OS-CAR project started in April 2000 with its first public release a year later as a self-installing compilation of "best practices" for high-performance classic Beowulf cluster computing. Since its inception approximately three years ago, OSCAR has matured to include cluster installation, maintenance, and operation capabilities and as a result has become one of the most popular cluster computing packages worldwide. In the past year, OSCAR has begun to expand into other cluster paradigms including Thin OSCAR, a diskless cluster solution, and High-Availability OSCAR, embracing fault tolerant capabilities. This paper will cover the current status of OSCAR including its two latest invocations – Thin OSCAR and High-Availability OSCAR – as well as the details of the individual component technology used in the creation of OSCAR.

1 Introduction

The growth of cluster computing in recent years has primarily been fueled by the price performance quotient. The hardware investment to obtain a supercomputer caliber system extends the capabilities to a much broader audience. This advancement enables individuals to have exclusive access to their own super computer.

The capability of individual computing clusters involves the well known exchange of hardware costs for software costs. This software is necessary to build, configure and maintain the ever growing number of distributed heterogeneous machines that make up these clusters. The Open Cluster Group (OCG) was formed to address cluster management needs. The first working group formed by OCG was the Open Source Cluster Application Resources (OSCAR) project. The OSCAR group began by pooling "best practice" techniques for High Performance Computing (HPC) clusters into an easy to use toolkit. This included the integration of HPC components with a basic wizard to drive the installation and configuration.

The initial HPC oriented OSCAR has evolved with the base cluster toolkit being distilled into a more general cluster framework. The separation of the HPC specific aspects enables the framework to be used for other cluster installation and management schemes. This has led to the creation of two additional OCG working groups that leverage the basic OSCAR framework. The "Thin-OSCAR" working group uses the framework for diskless clusters. The "HA-OSCAR" group is focusing on high availability clusters. The relationship of OCG working groups and the framework is depicted in Figure **??**.

The following sections will briefly discuss the OCG umbrella organization and the current working groups. The basic OSCAR framework will be discussed followed by a brief summary of the HPC specific components. Then the diskless (Thin-OSCAR) and high availability (HA-OSCAR) working groups will be covered followed by concluding remarks.
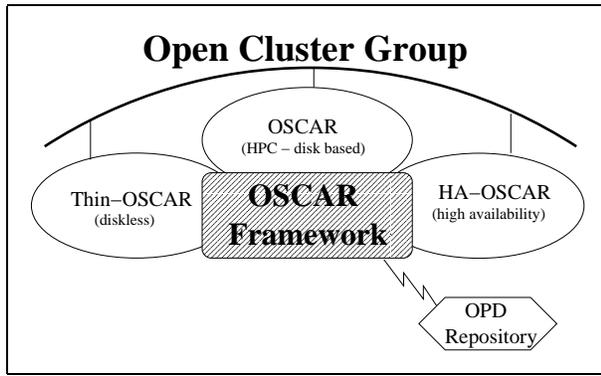
Figure 1: The Open Cluster Group (OCG) working groups share the OSCAR framework for cluster installation and management.

2 Background

The OSCAR working group was the first working group that was formed under the umbrella Open Cluster Group (OCG) organization. The OCG was formed after a handful of individuals met in April 2000 to discuss their forays into cluster construction and management [**?**]. The consensus was that much effort was being duplicated and a toolkit to assist with the integration and configuration of current "best practices" would be beneficial to the participants and the HPC community in general.

This initial meeting led to subsequent discussions and ultimately a public release of the cluster installation, configuration, and management toolkit OSCAR v1.0 in April 2001. This initial release addressed OCG's mission statement to make cluster computing simpler while making use of the commonly used open source software solutions. In the years that have followed, the OSCAR working group has enhanced the toolkit with features such as modular OSCAR packages and improved cluster management facilities.

The OSCAR project is comprised of a mixture of industry and academic/research members. The overall project is directed by a steering committee that is elected every two years from the current "core organizations". This "core" list is composed of those actively contributing to project development. The 2003 core organizations include: Bald Guy Software (BGS), Dell, IBM, Intel, MSC.Software, Indiana University, the National Center for Supercomputing Applications (NCSA), Oak Ridge National Laboratory (ORNL), and Université de Sherbrooke.

There have also been new OCG working groups created to address other cluster environments. These new working groups are named "Thin-OSCAR" and "HA-

OSCAR". The "Thin-OSCAR" project provides support for diskless clusters. The "HA-OSCAR" group is focused on high availability clusters. The different groups focus on different cluster environments but leverage much of the core facilities offered by the base OSCAR framework.

3 OSCAR Framework

The standard OSCAR release targets usage in a High Performance Computing (HPC) environment. The base OSCAR framework is not necessarily tied to HPC. Currently the framework and toolkit are simply referred to as OSCAR and are used to build HPC clusters. To avoid confusion throughout this paper, a distinction will be made by using OSCAR to refer to the entire toolkit and OSCAR framework for the base facilities.

The framework includes the set of base or "core" packages needed to build and maintain a cluster. There are two other package classifications: "included" and "third-party". The *included* class of packages includes commonly used HPC applications for OSCAR. These packages are closely maintained and tested for compatibility with each OSCAR release. The *third-party* distinction is provided for all other OSCAR packages (see Section **??**).

The core components enable a user to construct a virtual *image* of the target machine using System Installation Suite (SIS). There is also an OSCAR database (ODA) that stores cluster information. The final two components include a parallel distributed "shell" tool set called C3 and an environment management facility called Env-Switcher.

The fundamental function of OSCAR is to build and maintain clusters. This is greatly comprised of software package management. The guiding principle behind OSCAR and OCG is to use "best practices" when available. Thus, the Red Hat Package Manager (RPM) [**?**] is leveraged by OSCAR.[1] RPM files are pre-compiled binary versions of the software with meta data that is used to manage the addition, deletion, and upgrade of the package. RPM handles the conflict and dependence analysis. This add/delete/upgrade capability is a key strength of RPM. This is made use of by the framework in "OSCAR Packages".

3.1 System Installation Suite

The System Installation Suite (SIS) is based on the well known SystemImager tool [**?**, **?**]. SystemImager is used

---

[1]The underlying framework is designed to be as distribution agnostic as possible. The RPM name is slightly misleading but the system is available on distributions other than Red Hat.

to build an image – a directory tree that comprises an entire filesystem for a machine – that is used to install cluster nodes. The suite has two additional components: System Installer and System Configurator. These two components extend the standard SystemImager to allow for a description of the target to be used to build an image on the head node. This image has certain aspects generalized for on-the-fly customization via System Configurator. This dynamic configuration phase enables the image to be more general so items such as the network interface card are not in the SIS image. This capability allows for heterogeneity within the cluster nodes, while leveraging the established SystemImager management model.

SIS is used to "bootstrap" the node installs – kernel boot, disk partitioning, filesystem formatting, and base OS installation. The image used during the installation can also be used to maintain the cluster nodes. Modifying the image is as straight-forward as modifying a local filesystem. Once the image is updated, `rsync`[2] is used to update the local filesystem on the cluster nodes. This method can be used to install and manage an entire cluster, if desired. This image based cluster management is especially useful for maintaining diskless clusters and is used by the Thin-OSCAR working group, (see Section ??).

3.2 C3 Power Tools

The distributed nature of clusters introduces a need to execute commands and exchange files throughout the cluster. The Cluster, Command and Control (C3) tool set offers a comprehensive set of commands to perform parallel command execution across cluster(s) as well as file scatter and gather operations [?, ?]. The tools are useful at both administrative and user levels. The tool set is the product of scalable systems research being performed at ORNL [?].

C3 includes commands to execute (`cexec`) across the entire cluster – or a subset of nodes – in parallel. File scatter and gather (`cpush`/`cget`) operations are also available. The C3 power tools have been developed to span multiple clusters. This multi-cluster capability is not fully harnessed by OSCAR currently but is available for administrators or standard users.

C3 is used internally throughout the OSCAR toolkit to distribute files and perform parallel operations on the cluster. For example, the user management commands, e.g., `useradd`, are made cluster aware using the C3 tools. Since C3 enables standard Linux commands to be run in parallel, administrators can use the tools

to maintain clusters. A common example is the installation of a RPM on all nodes of the cluster, e.g.,

```
shell$ cexec rpm -ivh foo-1.0.i386.rpm
```

3.3 Environment Switcher

Managing the shell environment – both at the system-wide level as well as on a per-user basis – has historically been a daunting task. For cluster-wide applications, system administrations typically need to provide custom, shell-defendant startup scripts that, create and/or augment `PATH`, `LD_LIBRARY_PATH`, and `MANPAGE` environment variables. Alternatively, users could hand-edit their "dot" files (e.g., `$HOME/.profile`, `$HOME/.bashrc`, and/or `$HOME/.cshrc`) to create/augment the environment as necessary. Both approaches, while functional and workable, typically lead to human error – sometimes with disastrous results, such as users being unable to login due to errors in their "dot" files.

Instead of these models, OSCAR provides the `env-switcher` OSCAR package. `env-switcher` forms the basis for simplified environment management in OSCAR clusters by providing a thin layer on top of the Environment Modules package [?, ?]. Environment Modules provide an efficient, shell-agnostic method of manipulating the environment. Basic primitives are provided for actions such as: add a directory to a `PATH`-like environment variables, displaying basic information about a package, and setting arbitrary environment variables. For example, a module file for setting up a given application may include directives such as:

```
setenv FOO_OUTPUT $HOME/results
append-path PATH /opt/foo-1.2.3/bin
append-path MANPATH /opt/foo-1.2.3/man
```

The `env-switcher` package installs and configures the base Modules package and creates two types of modules: those that are unconditionally loaded, and those that are subject to system- and user-level defaults.

Many OSCAR packages use the unconditional modules to append the `PATH`, set arbitrary environment variables, etc. Hence, all users automatically have these settings applied to their environment (regardless of their shell) and guarantee to have them executed even when executing on remote nodes via `rsh`/`ssh`.

Other modules are optional, or a provide one-of-many selection methodology between multiple equivalent packages. This allows the system to provide a default set of applications, that optionally can be overridden by the user (*without* hand-editing "dot" files). A common example in HPC clusters is having multiple Message Passing Interface (MPI) [?, ?] implementations

---

[2]`rsync` is a tool to transfer files similar to `rcp`/`scp` [?].

installed. OSCAR installs both the LAM/MPI [**?**] and MPICH [**?**] implementations of MPI. Some users prefer one over the other, or have requirements only met by one of them. Other users wish to use both, switching between them frequently (perhaps for performance comparisons).

The `env-switcher` package provides trivial syntax for a user to select which MPI package to use. The `switcher` command is used to select which modules are loaded at shell initialization time. For example, the following command shows a user selecting to use LAM/MPI:

```
shell$ switcher mpi = lam-6.5.9
```

3.4   OSCAR Database

The OSCAR database, or ODA, is used to describe the software in an OSCAR cluster. As of OSCAR version 2.2.1, ODA has seven tables in a MySQL database named *oscar* which runs on the head node. Every OSCAR package has an XML meta file named `config.xml` that is used to populate the package database. This XML file contains a description of the package, and the RPM names associated with the package. Thus the database enables a user to find information on packages quickly and easily.

ODA provides a command line interface into the database via the `oda` command. This offers easier access into the database without having to use a MySQL client. ODA also provides an abstraction layer between the user and the actual backing store method. The internal organization of data is masked through the use of a uniform interface irrespective of this underlying database engine. The retrieval and update of cluster information is aided by the use of ODA "shortcuts" to assist with common tasks.

ODA is intended to provide OSCAR package developers with a central repository for information about the installed cluster. This alleviates the problem of every package developer having to keep an independent data store, and makes the addition and removal of packages easier.

3.5   OSCAR Packages

An OSCAR package is a simple way to wrap software and a given configuration for a cluster. The most basic OSCAR package is an RPM in the appropriate location in the package directory structure. The modularity of this facility allows for easy addition of new software to the framework. OSCAR packages are most useful, however, when they also provide supplemental documentation and a meta file describing the package. The packaging API provides authors the ability to make use

of scripts to configure the cluster software outside of the RPM itself. The scripts fire at different stages of the installation process and test scripts can be added to verify the process. Additionally, an OSCAR Package Downloader (OPD) (see Section **??**) is provided to simplify acquisition of new packages.

With this modular design, packages can be updated independently of the core utilities and core packages. The OSCAR toolkit is freely redistributable and therefore requires all "selected" packages to adhere to this constraint. However, any software which does not fulfill this requirement can be made available via an OSCAR repository with access via OPD.

The contents and directory structure of a typical OSCAR package are listed below. For further details on the creation of packages for the toolkit, see the OSCAR Architecture document in the development repository [**?**].

`config.xml` – meta file with description, version, etc.

`RPMS/` – directory containing binary RPM(s) for the package

`SRPMS/` – directory containing source RPM(s) used to build the package

`scripts/` – set of scripts that run at particular times during the installation/configuration of the cluster

`testing/` – unit test scripts for the package

`doc/` – documentation and/or license information

3.6   OSCAR Package Downloader

The OSCAR Package Downloader (OPD) provides the capability to download and install OSCAR software from remote package repositories. A package repository is simply an FTP or web site. Given the ubiquitous access to FTP and web servers, any organization can host their own OSCAR package repository and publish their packages on it. There is no central repository; the OPD network was designed to be distributed such that no central authority is required to publish OSCAR packages. Although the OPD client program downloads an initial list of repositories from the OSCAR Working Group web site,[3] arbitrary repository sites can be listed on the OPD command line.

Since package repositories are FTP or web sites, any traditional FTP client or web browser can also be used to obtain OSCAR packages. Most users prefer to use

---

[3]The centralized repository list is maintained by the OSCAR working group. Upon request, the list maintainers will add most repository sites.

the OPD client itself, however, because it provides additional functionality over that provided by traditional clients. OPD offers two interfaces: a simple menu-based mechanism suitable for interactive use and a command-line interface suitable for use by higher-level tools (or automated scripts).

Partially inspired by the Comprehensive Perl Archive Network (CPAN), OPD provides the following high-level capabilities:

- Automating access to a central list of repositories

- Browsing packages available at each repository

- Providing detailed information about packages

- Downloading, verifying, and extracting packages

While the job that OPD performs is actually fairly simple and could be performed manually, having an automated tool for these functions provides ease of use for the end-user, performs multiple checks to ensure that downloaded and extracted properly, and lays the groundwork for higher-level OSCAR package/retrieval tools.

4 OSCAR Toolkit

The integration of common HPC packages is a key feature of the OSCAR toolkit. The focus of this paper is to talk about the framework, however, and its use by the various OCG working groups. A brief highlight of the packages is provided with further details available in other articles [?, ?].

The "selected" packages that are included enable a user to install and configure the head node and cluster nodes to run HPC applications. These HPC packages include parallel libraries like LAM/MPI, MPICH and PVM. A batch queue system and scheduler – OpenPBS and MAUI – is included and setup with a reasonable set of defaults. The toolkit sets up common cluster services such as Network File System (NFS) and Network Time Protocol (NTP). Security packages like Pfilter [?] are setup as well as OpenSSH with non-interactive access to all nodes in the cluster from the head node. The testing scripts provided with the packages are executed by the OSCAR framework to validate the cluster installation.

5 Thin-OSCAR

5.1 Goals

Thin-OSCAR [4] is a workgroup dedicated to integrate diskless clustering techniques into OSCAR so that the

---

[4]Workgroup web site : http://thin-oscar.ccs.usherbrooke.ca/

OSCAR infrastructure can use diskless nodes. There are three class of nodes in the thin-OSCAR perspective : diskless, systemless (a disk is present in the node but there is no operating system on the disk) and diskfull (regular OSCAR node with disk). At the time of this writing, only diskless and diskfull nodes are supported out of the box. Systemless nodes are supported to some extent but you will have to fiddle with some config files manually. Moreover, a generic abstraction of a node is necessary to build a generic but comprehensive interface.

5.2 Principle of operation

The thin-OSCAR model is the following : the thin-OSCAR package is a collection of Perl scripts and libraries that are used to transform a regular SIS image (as used by OSCAR) into the two ram disks necessary for diskless and systemless nodes. The first ram disk to be transferred is called the "BOOT image" and is used in order to ensure that the node has network connectivity, NFS client capabilities and to create a raid0 array of ram disk [?]. Once this minimal image (less than 4Mb) has booted, the RUN image from the second ram disk can be transferred. The RUN image is build directly from the SIS image and contains the complete system that will run on the node. Some directories are copied from the SIS image while others are NFS exports (read-only) directly exported from the SIS image directory [?].

In order to build the BOOT image, you will need the SIS image name, modules name to integrate into this ram disk as well as the modules used by your NIC adapter and the kernel version you want to use. The use of the root raid in ram technique is necessary because one of the thin-OSCAR goals is to support the regular kernel (only 4 Mb or ramdisk) without recompilation.

5.3 mini howto

In order to download thin-OSCAR, please use Oscar Package Downloader and select the Université de Sherbrooke repository. Then download thin-OSCAR. Before actually using thin-OSCAR, you have to complete a regular installation from stage 1 (selection of packages) to stage 6 (setup networking). Assign nodes IP to MAC addresses and don't forget to click on the "Setup Network Boot" button.

Once this is done, you are ready to use the thin-OSCAR package. Go to `/usr/lib/oscar/packages/thin-oscar/` and run the `./oscar2thin.pl` script. You will go into an interface where you will have to define your diskless model, link the model to an OSCAR node and then

generate all the necessary ram disk (Configure All). Once this is done, you are finished and can reboot all the diskless nodes. You must know which kernel modules are necessary for your NIC before starting this process in order to setup network connectivity.

5.4 thin-OSCAR future

We will certainly move to a more modern RAM filesystem (tmpfs) as it is now available on regular kernels and toward an automatic detection of NIC modules so that BOOT ramdisk creation can be fully automated. Multicast transfer of the ramdisk is under study as it will shorten the boot time and the network usage during boot (especially for big clusters!).

The proof of concept of the thin-OSCAR has been done and, at the time of this writing, thin-OSCAR is used on a 180 node production diskless cluster [?]. An improved integration into the OSCAR framework is under development and will be available as soon as the OSCAR GUI enables it.

6 HA-OSCAR

High-Availability (HA) computing, once thought as only important to industry applications such as telecommunications, has become critically important to the fundamental mission of high-performance computing. This is because very large and complex application codes are being run on increasingly larger scale distributed computing environments. Since COTS (common off the shelf) hardware is typically employed to construct these environments (clusters), quite often the application code's runtime exceeds the hardware's aggregated mean-time-between-failure rate for the entire cluster. Thus, in order to efficiently run these very large and complex applications, high-availability computing techniques must be employed in the high-performance computing environment (HPC).

The current HPC release of OSCAR is fully suitable for mission critical systems as it contains several individual system elements that exhibit a single-point-of-failure trait. In order to support HA requirements, clustered systems must provide ways to eliminate single-point-of-failures. HA-OSCAR is a focus group with goals to add HA features to the original HPC OSCAR distribution. While HA-OSCAR is still a work in progress, the scope of this effort has been defined into three incremental steps; the creation of the HA-OSCAR white-paper [?], Active-Hot-Standby, and n + 1 Active-Active distributions.

Hardware duplication and network redundancy are common techniques utilized for improving the reliability and availability of computer systems. To achieve the HA-OSCAR cluster system, we must first provide a duplication of the cluster head node. There are different ways for implementing such an architecture, which includes Active-Active, Active-Hot Standby and Active-Cold Standby [?]. Currently, the Active-Hot Standby configuration is the initial model of choice. Figure ?? shows the HA-OSCAR cluster system architecture. We have experimented with and planned to incorporated Linux Virtual Server and Heartbeat mechanisms to our initial Active- Hot Standby HA-OSCAR distribution. However, we will extend the initial architecture to support the Active-Active HA after release of the Hot-Standby distribution. The Active-Active architecture will provide a better resource utilization since both head nodes will be simultaneously active and providing services. The dual master nodes will run redundant OpenPBS, MAUI, DHCP, NTP, TFTP, NFS, rsync and SNMP servers. In the event of a head node outage, all functions provided by that node will fail-over to the second redundant head node and all service requests will continue to be served, although at a reduced performance rate (i.e. in theory, 50% at the peak or busy hours).

An additional HA functionality to support in HA-OSCAR is that of providing a high- availability network via redundant Ethernet ports on every machine in addition to duplicate switching fabrics (network switches, cables, etc.) for the entire network configuration. This will enable every node in the cluster to be present on two or more data paths within its networks. Backed with this Ethernet redundancy, the cluster will achieve higher network availability. Furthermore, when both networks are up, an improved communication performance may be achieved by using techniques such as channel bonding of messages across the redundant communication paths.

7 Conclusion

The Open Cluster Group (OCG) has formed several working groups focused on improving cluster computing management. The first working group, OSCAR, has evolved over time as has the toolkit by the same name. The OSCAR toolkit has also been distilled in order to allow for more general usage. This more general OSCAR framework is being leveraged by subsequent working groups seeking to extend support for new cluster environments. These new environments include the areas of diskless and high availability clusters. These are being pursued by the Thin-OSCAR and HA-OSCAR working groups respectively. These OCG working groups are providing the cluster com-
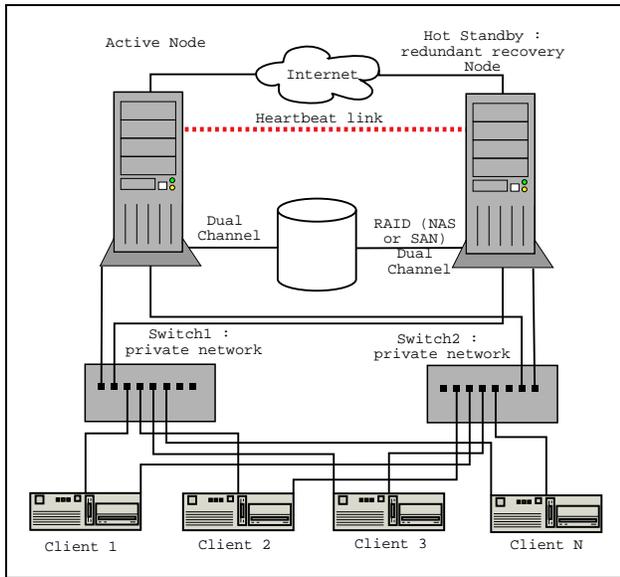
Figure 2: Diagram of HA-OSCAR architecture.

munity with sound tools to simplify and speed cluster installation and management.