



The OSCAR Revolution
Date: Saturday, June 01, 2002
Topic: Networking

Richard Ferri

Richard describes the history and goals of the Open Source Cluster Application Resource.

``Serve no whine before its time" is a bad pun attributed to Rob Pennington of NCSA at the very first OSCAR meeting, held in April 2000 at a hotel a stone's throw from Oak Ridge National Lab. A varied cast representing the national labs, academia and industry was assembled to discuss what was known at the time as the CCDK (Community Cluster Development Kit), which would morph into the OCG (Open Cluster Group) and their first project, OSCAR (the Open Source Cluster Application Resource).

The cast had broken clusters down into components and had assigned ``czars" (leaders) and ``whiners" (interested parties) for each component. The czars were to lead each component group, and the whiners were to whine loudly and often enough to make sure things got done on schedule, meeting the group's requirements. From that very first meeting when the czars and whiners were named, it was clear that OSCAR development would be different from all other software development that had gone before. After all, where else would one find companies like IBM, Dell, SGI and Intel working closely together to produce open solutions in a hotly contested space like clustering?

The original idea for OSCAR came about over dinner at a DOE-sponsored cluster meeting at Argonne National Lab, where Dr. Timothy Mattson, a research scientist at Intel, and Dr. Stephen Scott, a research scientist at Oak Ridge National Lab, discussed the problem of getting Linux clusters accepted into the mainstream. The problem, they decided, was that it was just too difficult for noncomputer programmers to assemble their own cluster. Books like *How to Build a Beowulf* (Sterling, et. al.) would help the computer savvy understand the concepts and construct his or her first cluster, but there were still daunting problems. There was an enormous amount of code to download, all at differing levels of reliability, support, integration and documentation. Sometimes the documentation for various packages was dated and contradictory. There were many Linux distributions to choose from, each trying to distinguish themselves by being slightly different from the next distribution. This meant that some commands worked differently or that different packages had to be installed to get a service to work properly.

The problem, they decided, was that with everyone trying to build their own cluster to tap into cheap cluster computing, each cluster was being built from scratch. There had to be some economy in compiling the best available software, practices and documentation in a single spot, integrating the package on different types of hardware and making it available to users for free (as in free beer). This concept, making clusters easy to build for the nonprogrammer, is a central tenet of OSCAR.

First Meeting

The historic first meeting in Oak Ridge was attended by Tim Mattson and Stephen Scott, the leaders of the OCG; Gabriel Bonner from SGI; Dave Lombard from MSC.Software; Rob Pennington of NCSA; Greg Lindahl, now of Conservative Computers; Ken Briskey and myself from IBM; Greg Astfalk from HP; and Clay Taylor from MPI Software Technologies. Shortly after the first meeting, Broahn Mann from Veridian joined to bring his parallel scheduling skill to the team, as did Jeremy Enos and Neil Gorsuch from NCSA (who implemented SSH on OSCAR) and Mike Brim from Oak Ridge National Lab (who wrote most of the integration scripts and packaging). Most recently, Jeff Squyres and Brian Barrett from Indiana University joined the OSCAR Project representing LAM/MPI. The disparate group agreed on three major core principles:

1. That the adoption of clusters for mainstream, high-performance computing is inhibited by a lack of well-accepted software stacks that are robust and easy to use by the general user.
2. That the group embraces the open-source model of software distribution. Anything contributed to the group must be freely distributable, preferably as source code under the Berkeley open-source license.
3. That the group can accomplish its goals by propagating best-known practices built up through many years of hard work by cluster computing pioneers.

With these principles firmly in place, the group used a divide-and-conquer method to list the components that comprise clusters. The component groups decided on the best-known, open-source solutions for each component and presented the information to the group at large. Taken collectively, these best-known practices for each component comprised a viable cluster solution. Even with the component solutions in hand, there was a massive and time-consuming integration effort by Oak Ridge National Lab, led by Mike Brim and Brian Luethke, and a separate test effort, which was led by Jenwei Hsieh, Tau Leng and Yung-Chin Fang from Dell. Through their efforts, and face-to-face and remote-integration parties, OSCAR eventually morphed into something to share with the rest of the community.

The Software Stack

It took nearly a full year, but OSCAR had a beta demonstration at SC2000 in Dallas, Texas at the Oak Ridge National Lab booth in November 2000. The beta was run on a heterogeneous cluster of servers provided by Dell and SGI. The first release was announced shortly thereafter and made a successful debut at LinuxWorld Expo in New York City in February 2001, at the Intel booth. Since then, there have been continuous improvements in the OSCAR software stack, which currently includes:

- 1 Linux installation: SIS (system installation suite). SIS is an open-source cluster installation tool based on the merger of LUI (the Linux utility for cluster install) and the popular SystemImager. SIS, developed by Michael Chase-Salerno and Sean Dague from IBM, made its debut in the 1.2.1 version of OSCAR. Most recently, Brian Finley of Bald Guy Software, the creator of SystemImager, has been attending the OSCAR meetings and looking for free beer, as in free beer.

- 1 Security: OpenSSH--the most common way to allow secure connections in a Linux environment. OpenSSH is a collection of packages that handles secure connections, server-side SSH services, secure key generation and any other functions used to support secure connections between computers.
- 1 Cluster management: for cluster-wide management operations, OSCAR uses the Cluster Command and Control (C3) management package developed at Oak Ridge National Lab by Stephen Scott and Brian Luethke, an East Tennessee State University student working at ORNL. C3 provides a "single-system illusion" so that a single command affects the entire cluster. C3 remains installed on the cluster nodes for later use by cluster users and administrators.
- 1 Programming environments: Message-Passing Interface (MPI) and Parallel Virtual Machine (PVM). Most cluster users write the software that runs on the cluster. There are many different ways to write software for clusters. The most common approach is to use a message-passing library. Currently, compilers or math libraries installed by OSCAR come from the Linux distribution. Both LAM/MPI and MPICH have been available since OSCAR 1.1.
- 1 Workload management: Portable Batch System (PBS) from Veridian and Maui Scheduler (developed by Maui High Times Computing Center). To time-share a cluster, some type of workload or job management is needed. Maui acts as a job scheduler for OSCAR, making all resource allocation and scheduling decisions. PBS is the job server/launcher and in addition to launching and killing jobs, handles job queues.

MSC.Software and OSCAR

MSC.Linux, a distribution developed by the Systems Division of MSC.Software Corporation, is of special importance in the acceptance of OSCAR. Shortly after the 1.0 version of OSCAR was available, MSC.Software announced their own cluster solution, the MSC.Linux Version 2001 operating system. This 2001 offering was in large part based on OSCAR, the first commercial offering based on the work of the OCG. MSC.Software's Joe Griffin added a Webmin interface to LUI (the first OSCAR cluster installation tool), which generated LUI bottom-line commands for multiple nodes to provide an easy-to-use interface in defining the nodes of the cluster and what resources to install on each. One of the original intents of the OCG was that commercial companies would see the value in the open OSCAR software stack and build their own proprietary or open stacks around the OSCAR stack. In so doing, companies using OSCAR would be freed from the mundane chores associated with building a cluster, such as providing the basic infrastructure, and could concentrate instead on more cutting-edge improvements to distinguish their offering.

Working Together

Like other far-flung open-source projects, it was clear from the beginning that doing the work of the consortium face to face would not always be an option. The travel expense was simply too great, and it was difficult to align so many schedules. To coordinate the work, the

group held open weekly phone conferences and would rely on mailing lists and an occasional meeting at a workshop or expo. There were face-to-face "integration parties" held quarterly, one at Intel in Hillsboro, Oregon and another at NCSA in Illinois. But for integrations held between meetings, a new construct was developed, called DIP Day, for distributed integration party. The intent of DIP Days was that everyone working on the project that had a cluster would set aside those days to work on OSCAR, jointly and remotely. Everyone would download the OSCAR package and install and run it, reporting any bugs to the group. On DIP Days, programmers were expected to provide fixes in real time, so that multiple iterations of the code could be tested shortly. Several conference calls with the entire team were held every DIP Day to assess progress and assign new work and priorities. By loosely coordinating the group between DIPs and face-to-face meetings, OSCAR made great strides in reliability and function.

The OSCAR Experience: First Impressions

The first thing one notices when untarring the OSCAR file is that the OSCAR integration and test team has done a thorough job; there is extensive documentation on how to install OSCAR, the system requirements, the licensing (GPL) and the theory behind OSCAR itself. There is a quick start guide for the impatient cluster administrator, as well as a full descriptive text. One also notices that there's nothing additional to download; it's all included in the single OSCAR tar file. OSCAR takes the traditional view of clusters--a single server with N compute nodes; the server is responsible for installing, scheduling and monitoring the compute nodes. Nodes in the cluster should be running homogeneous software, meaning the same distribution and version of Linux. The first command the user enters is `install_cluster`, which does a multitude of things: creates necessary directories; manages NFS and xinetd; installs LAM/MPI, C3, PBS, Maui, OpenSSH, SIS, Perl, SystemImager and MPICH; updates various profiles and configuration scripts; and launches the OSCAR wizard.

If all goes well, you're in for a pleasant surprise, namely, the OSCAR wizard. The OSCAR team felt the wizard would be another distinguishing feature of OSCAR in the field of Linux cluster solutions. The purpose of the wizard is clear--follow the wizard and you too can install a cluster painlessly. Each step along the wizard's path has entry and exit criteria. Once the exit criteria is successfully met, OSCAR gives a success message to indicate it's safe to move on to the next step.

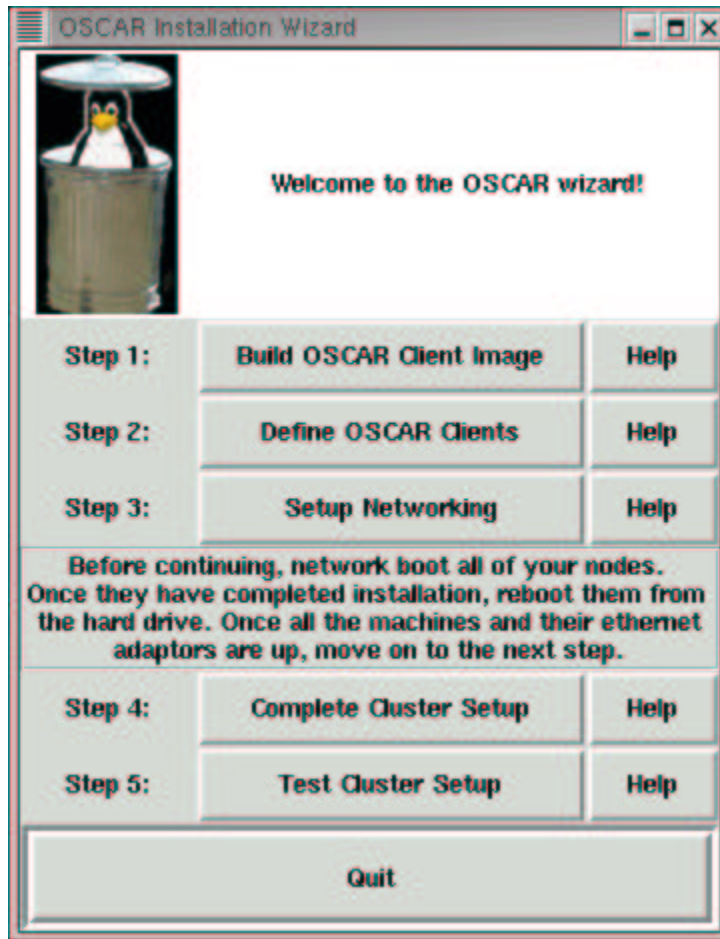


Figure 1. The OSCAR Wizard

Following the wizard, pressing the Build OSCAR client image button brings up the second panel, the Create a SystemImager Image panel.

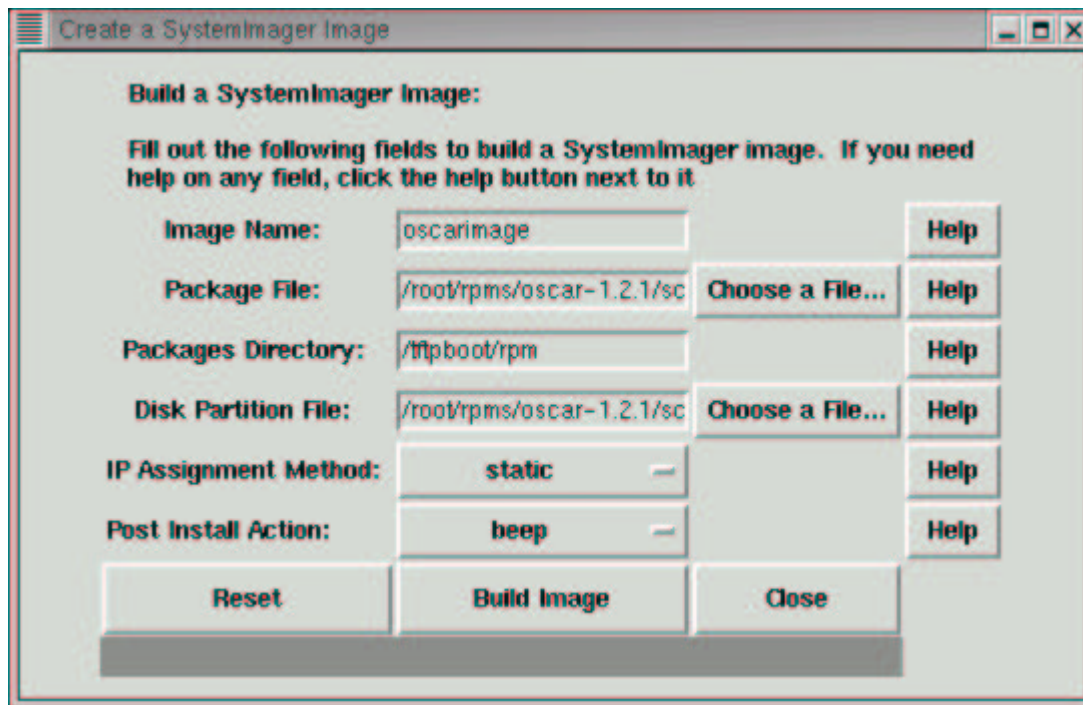


Figure 2. Building a SystemImager Image

The purpose of the SystemImager panel is to create a filesystem on the server that will later be installed on each client. The Image Name field allows the user to create multiple SystemImager images, each with a unique name. The Package File field provides a list of packages that will be installed on the client; OSCAR provides sample lists that meet most user requirements. The Packages Directory tells where the RPMs are coming from, and the Disk Partition File field allows the user to customize the disk partitions. Again, OSCAR provides default disk partition definition files for both IDE and SCSI drives. Pressing the Build Image button starts the process of building a client image on the server. Once complete, it's time to go back to the wizard for step two, defining the OSCAR clients.

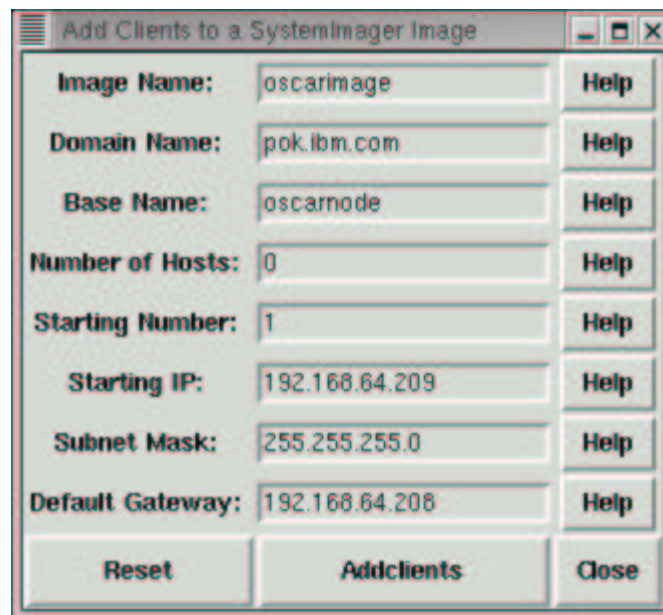


Image Name:	oscarimage	Help
Domain Name:	pok.ibm.com	Help
Base Name:	oscarnode	Help
Number of Hosts:	0	Help
Starting Number:	1	Help
Starting IP:	192.168.64.209	Help
Subnet Mask:	255.255.255.0	Help
Default Gateway:	192.168.64.208	Help
Reset		Addclients
		Close

Figure 3. Adding Clients to a SystemImager Image

From the Add Clients panel, the user can specify a range of IP addresses to be associated with a list of new clients. Each client is associated with an image name using the Image Name field. One can define a set of clients in a range of IP addresses, each having the same netmask and default gateway. Pressing the Addclients button builds client definitions for SIS. Once complete, it's back to step three on the wizard, Setup Networking.

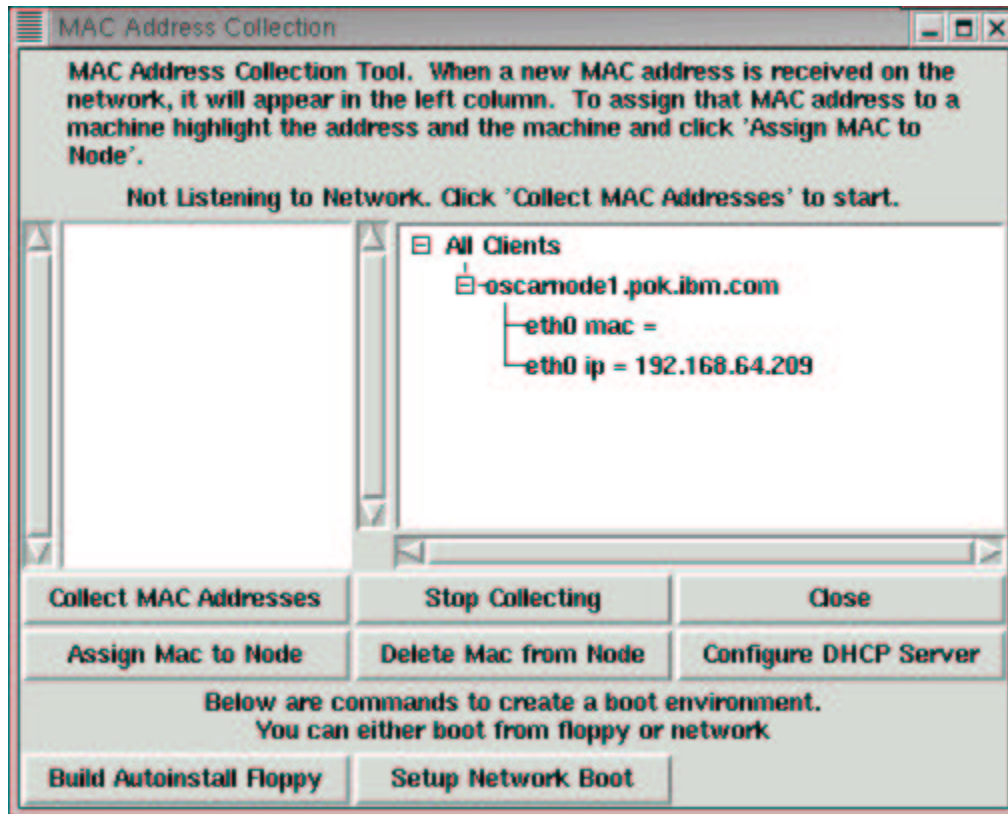


Figure 4. MAC Address Collection

From the Setup Networking panel, MAC addresses are collected for each client in the cluster. If the node is capable of true network (PXE) boot, you simply associate a MAC address with a client, and you're ready to power up the node. If the node is not PXE-enabled, you can write a SystemImager boot diskette from the Build Autoinstall Floppy button. Once the MAC addresses are collected, it's time to press the button to Configure the DHCP Server and boot all the nodes to initiate Linux installation.

Once all the nodes are installed, each node starts this really annoying and incessant beeping telling the system administrator to pop out the diskette or turn off PXE and reboot the node from the hard drive. Once they are all booted, the nodes are ready to Complete Cluster Setup from the wizard (really just syncing the time between servers and clients and running any package-sensitive postinstallation scripts). The Test Cluster Setup button from the wizard runs short jobs, checking each flavor of scheduler and parallel library.

Once the cluster is fully installed and functioning, there are test scripts to check the overall health of the cluster. Running the test_install script will check to make sure PBS or Maui Scheduler is configured and running, that the C3 tools are installed and that the cluster at that time is ready to start accepting parallel jobs.

OSCAR Futures

As of this writing, OSCAR 1.2.1 is available, which runs under Red Hat Linux 7.1. MSC.Linux Version 2001 is based on OSCAR 1.1. The 1.x1 releases were quite popular in the community--they've been downloaded roughly 25,000 times from SourceForge. However, the biggest problem OSCAR is now facing is that it takes the relatively few OSCAR developers time and effort to integrate all the new software packages that want to be included in OSCAR. The OSCAR 2.0 effort is underway, with an emphasis on establishing component APIs so that anyone with an open software package can integrate their package with OSCAR. The OCG itself is growing. Since Tim Mattson went on Intel sabbatical, Jeff Squyres from the University of Indiana has taken the leadership role in the OSCAR 2.0 architecture and consistency. Ibrahim Haddad from Ericsson [and a frequent *LJ* contributor] has joined the consortium with interesting ideas on how to bring OSCAR to near-telecom levels of reliability. Jim Garlick, representing Lawrence Livermore National Lab also has joined the consortium bringing his real-world large cluster scaling experience and concerns to the group.

Platforms and Distros

At the very first OSCAR meeting, it was agreed that OSCAR should not be tied to a particular Linux distribution or platform. However, to date the OCG's efforts have been largely focused on the Red Hat and MSC.Linux distributions and the IA-32 architecture. This focus will expand in 2002. The purpose of integrating SIS into OSCAR was to be able to support all RPM-based distributions: SuSE, Turbolinux, Red Hat, MSC.Linux and Caldera, and later to support deb-based distributions such as Debian. Additionally, the architecture of SIS makes it easy to port to new platforms. NCSA already has a beta version of OSCAR running on Itanium, and Oak Ridge has done extensive testing with Red Hat 7.2. Given the open API and ability to run on many different distributions and platforms, expect OSCAR and the OCG to expand dramatically this year.

Final Thoughts

The impacts of OSCAR on the Linux clustering community can be viewed in several different perspectives. At the most apparent, OSCAR is providing a useful clustering tool that is usable across various manufacturers' platforms. It has removed much of the ambiguity inherent in assembling software from various web sites by putting together a single, integrated, documented, tested and supported package. It is truly a clustering solution that a nonprogrammer can implement. However, beneath the surface, the OCG is a thriving consortium composed of national labs, academia and industry that are cooperating to bring new open-source solutions to Linux. Along the way, the consortium had to break new ground in ways to cooperate, with unique concepts like DIP Days. In retrospect, the consortium's most important long-term contribution to the community may be in developing new ways to work together for the betterment of open source.

Resources



Richard Ferri is a senior programmer in IBM's Linux Technology Center, where he works on open-source Linux clustering projects such as LUI and OSCAR. He now lives in a rural setting in

upstate New York with his wife Pat, three teen-aged sons and three dogs of suspect lineage.

This article comes from Linux Journal - The Premier Magazine of the Linux Community

<http://www.linuxjournal.com>

The URL for this story is:

<http://www.linuxjournal.com/article.php?sid=5559>