OSCAR

Jeremy Enos
Systems Engineer
NCSA Cluster Group
www.ncsa.uiuc.edu

February 1, 2002 Trieste, Italy

OSCAR – An Overview

- Open Source Cluster Application Resources
- Cluster on a CD – automates cluster install process
- IBM, Intel, NCSA, ORNL, MSC Software, Dell
- NCSA "Cluster in a BOX" base
- Wizard driven
- Nodes are built over network
- OSCAR <= 64 node clusters for initial target
- Works on PC commodity components
- RedHat based (for now)
- Components: Open source and BSD style license

Why OSCAR?

- NCSA wanted "Cluster-in-a-Box" Distribution
  - NCSA's "X-in-a-Box" projects could lie on top
  - X = Grid, Display Wall, Access Grid
- Easier, faster deployment
- Consistency among clusters
- Lowers entry barrier to cluster computing
  - no more "Jeremy-in-a-Box"
- Other organizations had the same interest
  - Intel, ORNL, Dell, IBM, etc.
  - NCSA jumps on board to contribute to OSCAR

OSCAR USAGE

http://clusters.top500.org/

TOP500 Poll Results

What Cluster system(Distribution) do you use?

Other	24%
Oscar	23%
Score	15%
Scyld	12%
MSC.Linux	12%
NPACI Rocks	8%
SCE	6%

233 votes (Feb. 01, 2002)

OSCAR Basics

- **What does it do?**
  - OSCAR is a cluster packaging utility
  - Automatically configures software components
  - Reduces time to build a cluster
  - Reduces need for expertise
  - Reduces chance of incorrect software configuration
  - Increases consistency from one cluster to the next
- **What *will* it do in the future?**
  - Maintain cluster information database
  - Work as an interface not just for installation, but also for maintenance
  - Accelerate software package integration into clusters

OSCAR Basics

How does it work?

- **version 1.0, 1.1**
  - LUI = Linux Utility for cluster Install
    - Network boots nodes via PXE or floppy
    - Nodes install themselves from rpms over NFS from the server
    - Post installation configuration of nodes and server executes
- **version 1.2+**
  - SIS = System Installation Suite
    - System Imager + LUI = SIS
    - Creates image of node filesystem locally on server
    - Network boots nodes via PXE or floppy
    - Nodes synchronize themselves with server via rsycn
    - Post installation configuration of nodes and server executes

Components

- **OSCAR includes (currently):**
  - C3 – Cluster Management Tools (ORNL)
  - SIS – Network OS Installer (IBM)
  - MPI-CH – Message Passing Interface
  - OpenSSH /OpenSSL – Secure Transactions
  - PBS – Job Queuing System
  - PVM – Parallel Virtual Machine
- **Current Prerequisites:**
  - Networked PC hardware with disk drives
  - Server machine with Redhat installed
  - Redhat CD(s)
  - 1 head node + N compute nodes

Installation Overview

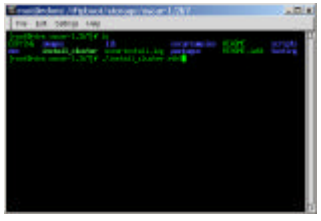
- Install RedHat
- Download OSCAR
- Print/Read document
- Copy RPMS to server
- Run wizard (install_cluster)
  - Build image per client type (partition layout, HD type)
  - Define clients (network info, image binding)
  - Setup networking (collect MAC addresses, configure DHCP, build boot floppy)
  - Boot clients / build
  - Complete setup (post install)
  - Install test suite
- Use cluster

OSCAR 1.2 Step by Step

- Log on to server as root
- mkdir –p /tftpboot/rpm
- copy all RedHat rpms from CDs to /tftpboot/rpm
- download OSCAR tarball
- tar –zxvf oscar-1.2.tar.gz
- cd oscar-1.2
- ./install_cluster

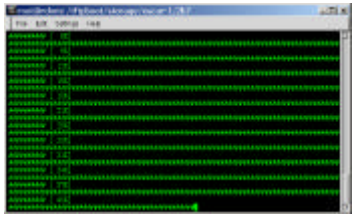
OSCAR 1.2 Step by Step

After untarring, run the install_cluster script…



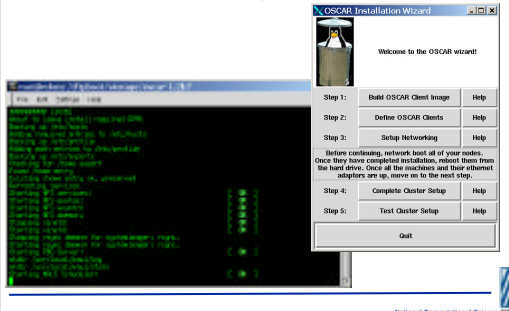
OSCAR 1.2 Step by Step

Sets up server and installs necessary software rpms



OSCAR 1.2 Step by Step

After starting services, dumps you into GUI wizard



OSCAR 1.2 Step by Step

Step 1: Build OSCAR Client Image

Build image with default *or* custom rpm lists and disk table layouts.

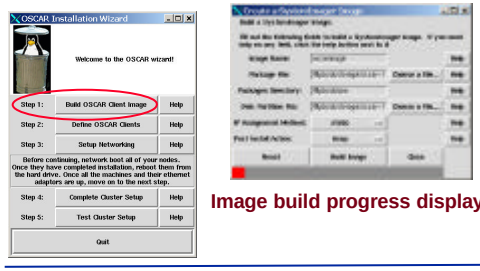
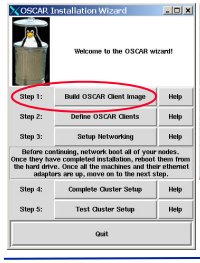


Image build progress displayed

OSCAR 1.2 Step by Step

Step 1: Build OSCAR Client Image

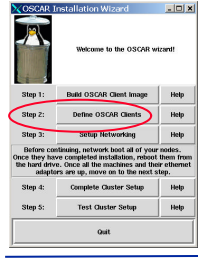
Image build complete.

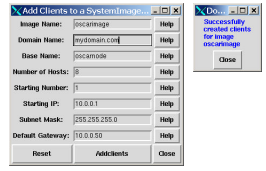


OSCAR 1.2 Step by Step

Step 2: Define OSCAR clients

Associate image(s) with network settings.

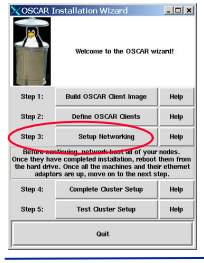


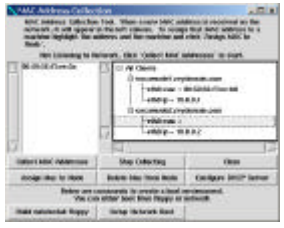


OSCAR 1.2 Step by Step

Step 3: Setup Networking

Collect MAC addresses and configure DHCP





OSCAR 1.2 Step by Step

Intermediate Step: Network boot client nodes

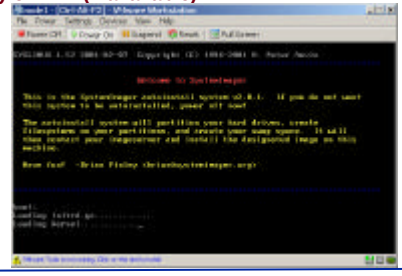
If the nodes are PXE capable, select the NIC as the boot device. Don't make this a static change, however.

Otherwise, just use the autoinstall floppy disk. It is less convenient than PXE, but a reliable failsafe.

OSCAR 1.2 Step by Step

Intermediate Step: Boot Nodes

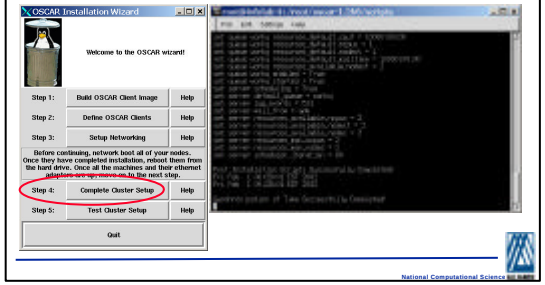
Floppy or PXE (if available)



OSCAR 1.2 Step by Step

Step 4: Complete Cluster Setup

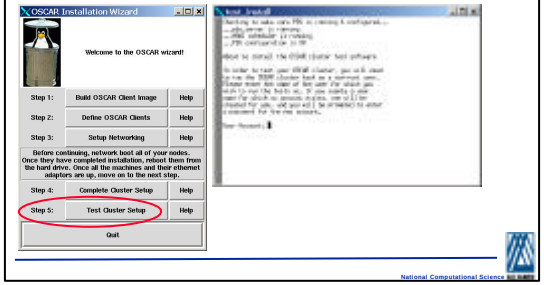
Output displayed in terminal window.



OSCAR 1.2 Step by Step

Step 5: Test Cluster Setup

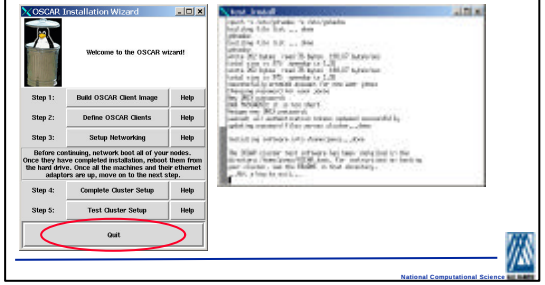
Installs test suite into specified user's home directory.



OSCAR 1.2 Step by Step

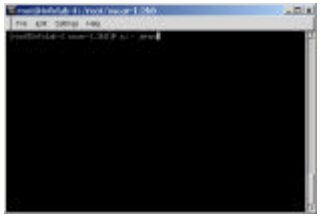
Test suite is installed. OSCAR setup complete!

Exit the wizard and run some tests…



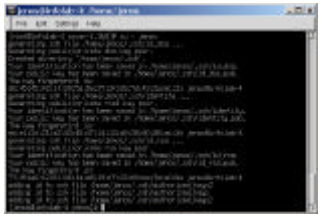
Testing OSCAR

First, "*su - username*" to log in as a user.
Root cannot run the tests.



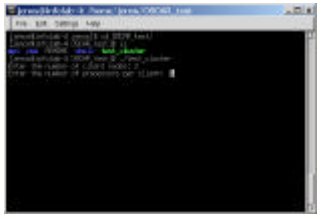
Testing OSCAR

The first time a user logs on to an OSCAR cluster, ssh user keys are generated.



Testing OSCAR

cd OSCAR_test and run *./test_cluster*
Enter number of nodes and procs per node.



Testing OSCAR

The following tests run across the number of nodes and processors you specified.

- PBS
- MPICH
- LAM
- PVM

The output will indicate success or failure.

Questions and Discussion

?

Next up… OSCAR 2.

OSCAR 2.0

Fall, 2002

Timeline

- OSCAR invented
  - First development meeting in Portland, OR, USA
  - September, 2000
- OSCAR 1.0 released
  - February, 2001
  - Real users and real feedback
  - OSCAR 2 design discussion begins
- OSCAR 1.1 released
  - July, 2001
  - RedHat 7.1 support
  - Tidy install process / fix potholes
- OSCAR 1.2 beta released
  - January, 2002
  - SIS integrated
- OSCAR 2.0
  - Fall, 2002

OSCAR 2

- Major Changes - Summary
  - No longer bound to OS installer
  - Components are package based, modular
  - Core set of components mandatory
  - API established and published for new packages
  - Package creation open to community
  - Database maintained for node and package information
  - Add/Remove Node process will be improved
  - Web based wizard
  - Scalability enhancements
  - Security Options
  - Support more distributions and architectures
  - New Features

OSCAR 2 – Install Options

- Without OS Installer
  - Installs on existing workstations w/o re-installing OS
  - Long list of prerequisites
  - Unsupported (at least initially)
- With OS Installer
  - OSCAR has hooks to integrate nicely with installer
  - System Installation Suite
  - RedHat Installer
  - ___?___ Installer

OSCAR 2 - MODEL

OSCAR 2 (The Glue)

OSCAR 2 - MODEL

OSCAR 2 (The Glue)

Core Components

C3	SSH

OSCAR 2 - MODEL

OSCAR 2 (The Glue)

Core Components

C3	SSH
MAUI	SIS
MPICH	LAM
PVM	PVFS
Grid in a box	VMI
Wall in a box	Giganet
Myrinet	Firewall/NAT
Monitoring	X Cluster Tools

OSCAR 2 – API

- Package creation open to community
- Core set of mandatory packages
- Each package must have 3 scripts:
  - Install
  - Uninstall
  - Configure
- Server software is in package form
  - enables distribution of server services
- ODR – OSCAR Data Repository
  - Node information
  - Package information
  - SQL Database or Flat File
  - Readable by nodes via API calls

OSCAR 2 – Web Wizard

- Webmin based
  - http://www.webmin.com
- Possible Interface: 3 Install buttons
  - Simple
    - one click install
    - tested and supported
  - Standard
    - typical combos presented
    - tested and supported
  - Expert
    - every option presented
    - any configuration combination

OSCAR – Scalability Enhancements

- LUI
  - Merging with System Imager (System Installation Suite)
  - Scalability to improve to at least 128 nodes
- PBS
  - Home directory spooling (nfs instead of RSH)
  - Open file descriptor limit
  - Max server connections
  - Job basename length
  - Polling intervals
- Maui
  - Job attributes are limited to N nodes
- SSH
  - Non privileged ports (parallel SSH tasks)
  - User based keys
- Single Head Node model trashed
  - Distribution of server services

OSCAR 2 – Security Options

- Wizard based
  - Security options selected in wizard installer
- Security schemes
  - All Open
  - Nodes isolated to private subnet
  - Cluster firewall / NAT
  - Independent packet filtering per node
- Security is a package, like any other software
- Probably will use “pfilter”
  http://pfilter.sourceforge.net/

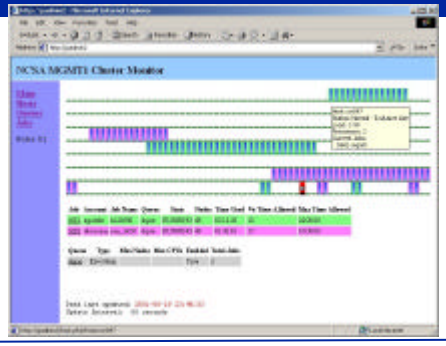
OSCAR 2 – Distribution and Architecture Support

- Distribution support goals
  - Redhat , Debian , SuSE, Mandrake, Turbo
  - Only when we’re satisfied with Redhat OSCAR
- Architectures
  - IA32, IA64, Alpha?

OSCAR 2 – New Features

- High speed interconnect support
  - Myrinet
  - Others to come
- ATLAS, Intel MKL?
- Maui Scheduler
- LAM/MPI
- Monitoring
  - CluMon (work in progress)
  - Performance Co-Pilot (PCP)
  - See http://padmin2.ncsa.uiuc.edu

CluMon



Considerations beyond OSCAR 2

- Diskless node support (lots of interest)
- Compatibility with other cluster packaging tools!
  - NPACI Rocks, SCE, Scyld, etc.
  - Standardized API
  - Cluster Package "XYZ" can interface with Rocks, OSCAR, etc.
- PVFS
  - Still testing
- NFS3
- Cluster of virtual machines (VMware, etc)
  - variable host operating systems (Windows, etc.)
  - multiple machine images
  - imagine where it could take us!

OSCAR Development Path

- version 1.0
  - Redhat 6.2 based
  - Nodes built by LUI (IBM)
  - Proof of concept (prototype)
  - Many steps, sensitive to bad input
  - Flexibility was intention; identify user needs
- version 1.1
  - Redhat 7.1 based
  - Nodes built by LUI
  - More automation for homogenous clusters
  - SSH: user keys instead of host keys
  - Scalability enhancements (ssh , PBS)
  - Latest software versions

OSCAR Development Path (cont.)

- **version 1.2**
  - moved development to SourceForge www.sourceforge.net
  - LUI replaced by SIS
  - Redhat 7.1 based
  - Packages adjust to SIS based model
  - Latest software versions (C3 tools, PBS, MPICH)
  - Start releasing monthly
- **version 1.21 (1.3 beta?)**
  - Redhat 7.2 support
- **version 1.3**
  - Add/Delete node support implemented
  - Security configuration on head node
  - ia64 support

OSCAR Development Path (cont.)

- **version 1.4**
  - Grouping support (nodes)
  - GUI replacement: Webmin (command line backend)
  - Core packages read/write configuration to database
    - SSH, C3, SIS, Wizard
  - Package DB API published
    - modular package support
- **version 1.5**
  - Existing packages use database
    - PBS, MPICH, PVM, LAM, Maui

OSCAR Development Path (cont.)

- **version 1.6 (2.0 beta?)**
  - custom security configuration for compute nodes
  - single head node model expires
    - head node holds OSCAR database
    - packages can designate their own head node (e.g. PBS)
  - package writing opened to community
  - the modularity advantage
    - "open packages" and "certified packages"
    - commercial packages can now be offered
    - licensing issues disappear
    - compatibility with other packagers (hopefully)

For Information

- Open Cluster Group Page
  - http://www.openclustergroup.org
- Project Page
  - http://oscar.sourceforge.net/
    - Download
    - Mailing lists
    - FAQ
- Questions?

OSCAR

Workload Management

Jeremy Enos
OSCAR Annual Meeting
January 10-11, 2002

Topics

- Current Batch System – OpenPBS
- How it Works, Job Flow
- OpenPBS Pros/Cons
- Schedulers
- Enhancement Options
- Future Considerations
- Future Plans for OSCAR

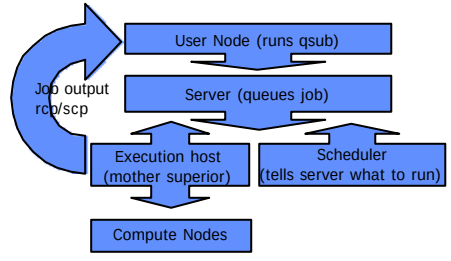
OpenPBS

- PBS = Portable Batch System
- Components
  - Server – single instance
  - Scheduler – single instance
  - Mom – runs on compute nodes
  - Client commands – run anywhere
    - qsub
    - qstat
    - qdel
    - xpbsmon

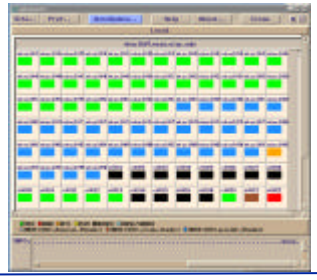
OpenPBS - How it Works

- User submits job with "qsub"
- Execution host (mom) must launch all other processes
  - mpirun
  - ssh/rsh/dsh
  - pbsdsh
- Output
  - spooled on execution host (or in user's home dir)
  - moved back to user node (rcp/scp)

OpenPBS – Job Flow



OpenPBS – Monitor (xpbsmon)



OpenPBS - Schedulers

- **Stock Scheduler**
  - Pluggable
  - Basic, FIFO
- **Maui**
  - Plugs into PBS
  - Sophisticated algorithms
  - Reservations
  - Open Source
  - Supported
  - Redistributable

OpenPBS – in OSCAR2

1. List of available machines
2. Select PBS for queuing system
   1. Select one node for server
   2. Select one node for scheduler
      1. Select scheduler
   3. Select nodes for compute nodes
   4. Select configuration scheme
      - staggered mom
      - process launcher (mpirun, dsh, pbsdsh, etc)

OpenPBS – On the Scale

Pros

- Open Source
- Large user base
- Portable
- Best option available
- Modular scheduler

Cons

- License issues
- 1 year+ devel lag
- Scalability limitations
  - number of hosts
  - number of jobs
  - monitor (xpbsmon)
- Steep learning curve
- Node failure intolerance

OpenPBS – Enhancement Options

- qsub wrapper scripts/java apps
  - easier for users
  - allows for more control of bad user input
- 3rd party tools, wrappers, monitors
- Scalability source patches
- "Staggered moms" model
  - large cluster scaling
- Maui Silver model
  - "Cluster of clusters" diminishes scaling requirements
  - never attempted yet

Future Considerations for OSCAR

- Replace OpenPBS
  - with what? when?
  - large clusters are still using PBS
- Negotiate better licensing with Veridian
  - would allow us to use a later revision ofOpenPBS
- Continue incorporating enhancements
  - test Maui Silver, staggered mom, etc.
  - 3rd party extras, monitoring package

Using PBS

- **Popular PBS commands**
  - qsub: submits job
  - qstat: returns queue status
  - qdel: deletes a job in the queue
  - pbsnodes: lists or changes node status
  - pbsdsh: just used in scripts- a parallel launcher
- **qsub: Not necessarily intuitive**
  - accepts it's own arguments
  - accepts only scripts, NOT executables
  - scripts can't have arguments either
  - runs tasks ONLY on a single mom (mother superior)
  - 3 methods of using qsub

Using PBS, qsub Method 1:

Type every option per command

- **use qsub and all options to launch a script for each executable**

```
qsub –N jobname –e error.out –o output.out –q queuename\
-l nodes=X:ppn=Y:resourceZ,walltime=NN:NN script.sh
```

- **script.sh**

```
#!/bin/sh
echo Launchnode is $hostname
pbsdsh /my_path/my_executable
#done
```

- **Most flexible**

Using PBS, qsub Method 2:

Type only varying options per command

use qsub and dynamic options to launch a script for each executable

```
qsub -l nodes=X:ppn=Y:resourceZ,walltime=NN:NN script.sh
```

script.sh

```
#!/bin/sh
#PBS –N jobname
#PBS –o output.out
#PBS –e error.out
#PBS –q queuename
echo Launchnode is $hostname
pbsdsh /my_path/my_executable
#done
```

Medium flexibility

Using PBS, qsub Method 3:

- Type fixed arguments in a command, but no need to create a script each time
- use *qsub wrapper* and fixed arguments to generate a script for each executable
  - submitjob nodes ppn walltime queue resource jobname "executable +arg1 +arg2"
- "submitjob" is an arbitrary script that wraps qsub
  - strips fixed arguments off of command line
  - what's left is intended PBS command "executable arg1 arg2"
  - passes that in environment to qsub, which submits helper script: qlaunch
  - qlaunch runs on mother superior (first node) and launches actual PBS command intended

Using PBS: qsub, Method 3

submitjob

```
#!/bin/sh
#
export nodes=$1
export ppn=$2
export walltime=$3
export queue=$4
export resource=$5
export jobname=$6
export outfile=$7
export procs=`expr $nodes \* $ppn`
shift
shift
shift
shift
shift
shift
shift
export PBS_COMMAND=$*
qsub -l walltime=$walltime,nodes=$nodes:ppn=$ppn:$resource\
      -N $jobname \
      -A SLOGNAME \
      -q $queue@server.domain.edu:15001 \
      -o $outfile \
      -e $outfile.err \
      -V \
      /usr/local/bin/qlaunch
```

qlaunch

```
#!/bin/sh
launchname=`/bin/hostname`
echo "Launch node is $launchname"
echo PBS_COMMAND is $PBS_COMMAND
echo
cmdfile=~/.$PBS_JOBID.cmd

# Create the shell script to run the MPI program and use pbsdsh to execute it
cat > $cmdfile <<EOF
#!/bin/sh
cd $cmd_dir
$PBS_COMMAND
EOF

chmod u+x $cmdfile
pbsdsh $cmdfile

rm -rf $cmdfile || echo "Failed to remove temporary command file $cmdfile."
```

An FAQueue

- How do I create a queue?
  - qmgr –c "create queue QUEUENAME"
  - qmgr - c "set queue QUEUENAME PARAM = VALUE"
  - qmgr –c "list queue QUEUENAME"
  - man qmgr (for more information)
- How do I associate nodes with a queue?
  - You don't. Think of a queue as a 3 dimensional box* that a job must fit in to be allowed to proceed. The three dimensions are: "nodes X procs X walltime"
  - *Could technically be more than 3 dimensions
- How do I target specific nodes then?
  - Specify a *resource* on the qsub command. The resource names are defined in /usr/spool/PBS/server_priv/nodes. They are arbitrary strings.

Tips to get started

- Check out the C3 commands
  - cexec, cpush very useful
  - *ls /opt/c3*/bin* (see all the C3 commands)
- Check out PBS commands
  - ls /usr/local/pbs/bin
- Check out the Maui scheduler commands
  - ls /usr/local/maui/bin
- Join the mailing lists!
  - http://oscar.sourceforge.net/
  - Send feedback

Questions and Discussion