NPACI Rocks Cluster Distribution:
Users Guide

User's Guide for NPACI Rocks version 3.2.0 Edition

NPACI Rocks Cluster Distribution: Users Guide :
User's Guide for NPACI Rocks version 3.2.0 Edition
Published 2004
Copyright © 2004 by UC Regents

Table of Contents

List of Tables

Preface

1. User Testimonials

Quotes from Rocks users. Reprinted with permission.

Martin Beaudoin, M.Sc.A, ing. IREQ, Quebec, Canada.

"I am very impressed by the quality of the NPACI Rocks distro. I consider that your current version of NPACI Rocks saved me a lot of time and trouble setting up my Beowulf cluster."

Roy Dragseth, M.Sc., The Computer Center, Univ. of Tromso, Norway.

"The NPACI Rocks Cluster Distribution gave us a turnkey solution that worked out of the box. It was a real time-saver when we started with clusters and has proven its stability and usability in our production environment with scientists and students running very demanding tasks."

Laurence Liew, Scalable Systems Pte Ltd, Singapore.

"Rocks has been the foundation upon which we deliver our Linux High Performance Computing cluster solutions to our PAYING customers. The ease of installation and if needed specialised configurations makes it possible to meet our various customers unique requirements.

The Rocks methodology also means the ability to deliver a robust and scalable cluster solutions to customers. The ease of management afforded by Rocks means customers can concentrate on their scientific computing instead of worrying about the management of their cluster."

Tim Carlson, PhD, PNNL, Richland, WA.

"I am extremely happy with the performance of the NPACI Rocks cluster distribution. I have been installing clusters for the past 5 years and have tried Scyld, OSCAR, and various "roll-your-own" techniques. In my experience, Rocks clusters provide the right combination of stability, maintainability, and ease of installation."

Steve Jones, Iceberg Cluster, Bio-X at Stanford University, Palo Alto, CA.

"As the sole cluster operations administrator charged with the management of our 301 node high performance computing cluster, a distribution such as NPACI Rocks allows me to deliver on the promise I made to provide a stable computational infrastructure. It feels as if I have a crew of admin's and engineers working with me by having the team at NPACI Rocks answering questions & providing assistance."

Frederick P. Arnold, Jr, NUIT, Northwestern University, Evanston IL.

"We tried several clustering alternatives before settling on Rocks as our default system two years ago. It has proven easy to install, configure, extend, and use. It is extremely robust in production, and now forms the core computing environment for Northwestern Chemistry's Theory Group. It also evangalizes well to other groups once they see it in operation."

2. Introduction

From a hardware component and raw processing power perspective, commodity clusters are phenomenal price/performance compute engines. However, if a scalable "cluster" management strategy is not adopted, the favorable economics of clusters are offset by the additional on-going personnel costs involved to "care and feed" for the machine. The complexity of cluster management (e.g., determining if all nodes have a consistent set of software) often overwhelms part-time cluster administrators, who are usually domain application scientists. When this occurs, machine state is forced to either of two extremes: the cluster is not stable due to configuration problems, or software becomes stale, security holes abound, and known software bugs remain unpatched.

While earlier clustering toolkits expend a great deal of effort (i.e., software) to compare configurations of nodes, Rocks makes complete Operating System (OS) installation on a node *the basic* management tool. With attention to complete automation of this process, it becomes faster to reinstall all nodes to a known configuration than it is to determine if nodes were out of synchronization in the first place. Unlike a user's desktop, the OS on a cluster node is considered to be *soft state* that can be changed and/or updated rapidly. This is clearly more heavywieght than the philosophy of configuration management tools [Cfengine] that perform exhaustive examination and parity checking of an installed OS. At first glance, it seems wrong to reinstall the OS when a configuration parameter needs to be changed. Indeed, for a single node this might seem too severe. However, this approach scales exceptionally well, making it a preferred mode for even a modest-sized cluster. Because the OS can be installed from scratch in a short period of time, different (and perhaps incompatible) application-specific configurations can easily be installed on nodes. In addition, this structure insures any upgrade will not interfere with actively running jobs.

One of the key ingredients of Rocks is a robust mechanism to produce customized distributions (with security patches pre-applied) that define the complete set of software for a particular node. A cluster may require several node types including compute nodes, frontend nodes file servers, and monitoring nodes. Each of these roles requires a specialized software set. Within a distribution, different node types are defined with a machine specific Red Hat Kickstart file, made from a Rocks Kickstart Graph.

A Kickstart file is a text-based description of all the software packages and software configuration to be deployed on a node. The Rocks Kickstart Graph is an XML-based tree structure used to define RedHat Kickstart files. By using a graph, Rocks can efficiently define node types without duplicating shared components. Similiar to mammalian species sharing 80% of their genes, Rocks node types share much of their software set. The Rocks Kickstart Graph

easily defines the *differences* between node types without duplicating the description of their *similarities*. See the Bibliography section for papers that describe the design of this structure in more depth.

By leveraging this installation technology, we can abstract out many of the hardware differences and allow the Kickstart process to autodetect the correct hardware modules to load (e.g., disk subsystem type: SCSI, IDE, integrated RAID adapter; Ethernet interfaces; and high-speed network interfaces). Further, we benefit from the robust and rich support that commercial Linux distributions must have to be viable in today's rapidly advancing marketplace.

2.1. Rocks and SQL

Wherever possible, Rocks uses automatic methods to determine configuration differences. Yet, because clusters are unified machines, there are a few services that require "global" knowledge of the machine -- e.g., a listing of all compute nodes for the hosts database and queuing system. Rocks uses an SQL database to store the definitions of these global configurations and then generates database reports to create service-specific configuration files (e.g., DHCP configuration file, /etc/hosts, and PBS nodes file).

2.2. Goals

Since May 2000, the Rocks group has been addressing the difficulties of deploying manageable clusters. We have been driven by one goal: *make clusters easy*. By *easy* we mean easy to deploy, manage, upgrade and scale. We are driven by this goal to help deliver the computational power of clusters to a wide range of scientific users. It is clear that making stable and manageable parallel computing platforms available to a wide range of scientists will aid immensely in improving the state of the art in parallel tools.

3. Contact

There are several Mailman lists that you can join to follow discussions, get announcements, or be a developer.

3.1. Discussion Lists

npaci-rocks-discussion[1]

> The place where users or others can discuss additions, improvements, techniques, and anything else that pertains to clusters. Unmoderated, anyone can join.

npaci-rocks-devel[2]

> This is the developers list where people who are contributing software to Rocks can discuss detailed architecture. This list is unmoderated, but additions are password protected.

3.2. Email

distdev[3]

> You can contact the cluster development group directly at SDSC[4] if you have other questions.

4. Collaborators

4.1. Research Groups

San Diego Supercomputer Center, UCSD

- Philip Papadopoulos
- Mason Katz
- Greg Bruno
- Federico Sacerdoti
- Bill Link
- Nadya Williams

Millennium Group at UC Berkeley

- David Culler
- Matt Massie
- Albert Goto
- Bret Chun
- Philip Buonodanna
- Eric Fraser

Scalable Systems Pte Ltd in Singapore

- Laurence Liew
- Najib Ninaba

The Open Scalable Cluster Environment in Thailand

- Putchong Uthayopas

4.2. Companies

Compaq Computer Corporation (Now HP)

Compaq has donated several x86 and ia64 servers to the Rocks group for development, and production clustering. We gratefully acknowledge the support of Compaq Computer Corporation, especially Sally Patchen, the California Educational Accounts Manager.

Dell

Dell has loaned us several x86 and ia64 boxes to make sure Rocks always supports their server hardware. They have also provided extremely valuable bug reports, and feature requests. We thank Dell for helping make Rocks stronger.

4.3. Contributors

This list is invariably incomplete. We would like to include all those who have contributed patches to the Rocks system. Please contact us if your name has been erroneously omitted. Names appear in alphabetical order.

- Fred Arnold, NUIT, Northwestern University, Evanston IL
- Tim Carlson, PNNL, Richland WA
- Roy Dragseth, University of Tromso, Norway.
- Steve Jones, Bio-X at Stanford University, Palo Alto, CA
- Doug Nordwall, PNNL, Richland WA
- Glen Otero, Callident, San Diego CA

Notes

1. https://lists.sdsc.edu/mailman/listinfo.cgi/npaci-rocks-discussion
2. https://lists.sdsc.edu/mailman/listinfo.cgi/npaci-rocks-devel
3. mailto:distdev@sdsc.edu
4. http://www.sdsc.edu

Chapter 1. Installing a Rocks Cluster

1.1. Getting Started

The advantage of using Rocks to build and maintain your cluster is simple. Building clusters is straightforward, but managing its software can be complex. This complexity becomes most unmanagable during cluster installation and expansion. Rocks provides mechanisms to control the complexity of the cluster installation and expansion process, and provides performance monitoring tools as well.

This chapter describes the steps to build your cluster and install its software.

1.1.1. Supported Hardware

Since Rocks is built on top of RedHat Linux releases, Rocks supports all the hardware components that RedHat supports, but only supports the x86 and IA-64 architectures (no Alpha, SPARC or Yamhill).

Processors

- x86 (ia32, AMD, etc.)
- IA-64 (Itanium, McKinley, etc.)
- x86_64 (AMD Opteron)

Networks

- Ethernet (All flavors that RedHat supports, including Intel Gigabit Ethernet)
- Myrinet (Lanai 9.x)

1.1.2. Physical Assembly

The first thing to manage is the physical deployment of a cluster. Much research exists on the topic of how to physically construct a cluster. The cluster cookbook[1] can be a good resource. A majority of the O'Reilly Book[2] *Building Linux Clusters* is devoted to the physical setup of a cluster, how to choose a motherboard, etc. Finally, the book *How to Build a Beowulf* also has some good tips on physical construction.

We favor rack-mounted equipment[3] (yes, it is more expensive) because of its relative reliability and density. There are Rocks clusters, however, that are built from mini-towers. Choose what makes sense for you.

The physical setup for a Rocks Cluster contains one or more of the following node types:

- *Frontend*

  Nodes of this type are exposed to the outside world. Many services (NFS, NIS, DHCP, NTP, MySQL, HTTP, ...) run on these nodes. In general, this requires a competent sysadmin. Frontend nodes are where users login in, submit jobs, compile code, etc. This node can also act as a router for other cluster nodes by using network address translation (NAT).

  Frontend nodes generally have the following characteristics:

  - Two ethernet interfaces - one public, one private.
  - Lots of disk to store files.

- *Compute*

  These are the workhorse nodes. They are also disposable. Our management scheme allows the complete OS to be reinstalled on every compute node in a short amount of time (~10 minutes). These nodes are not seen on the public Internet.

  Compute nodes have the following characteristics:

  - Power Cable
  - Ethernet Connection for administration
  - Disk drive for caching the base operating environment (OS and libararies)
  - Optional high-performance network (e.g., Myrinet)

- *Ethernet Network*

  All compute nodes are connected with ethernet on the private network. This network is used for administration, monitoring, and basic file sharing.

- *Application Message Passing Network*

  All nodes can be connected with Gigabit-class networks and required switches. These are low-latency, high-bandwidth networks that enable high-performance message passing for parallel programs.

The Rocks cluster architecture dictates these nodes types are connected as such:
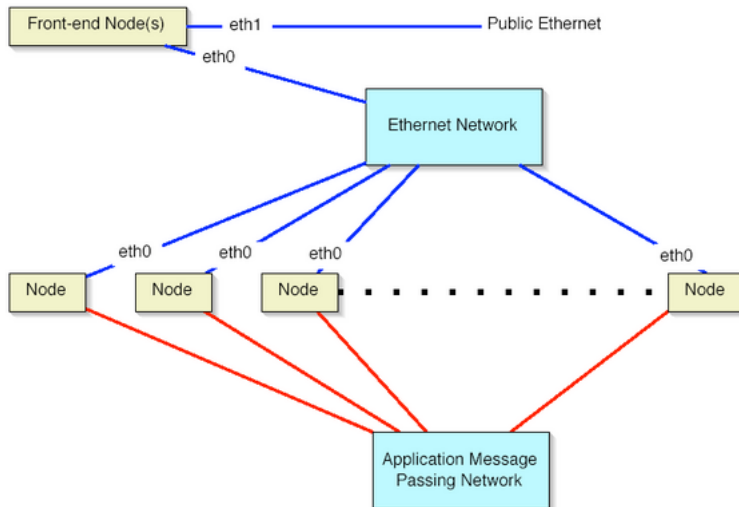
On the compute nodes, the Ethernet interface that Linux maps to *eth0* must be connected to the cluster's Ethernet switch. This network is considered *private*, that is, all traffic on this network is physically separated from the external public network (e.g., the internet).

On the frontend, two ethernet interfaces are required. The interface that Linux maps to *eth0* must be connected to the same ethernet network as the compute nodes. The interface that Linux maps to *eth1* must be connected to the external network (e.g., the internet or your organization's intranet).

Once you've physically assembled your cluster, each node needs to be set to boot *without a keyboard*. This procedure requires setting BIOS values and, unfortunately, is different for every motherboard. We've seen some machines where you cannot set them to boot without a keyboard.

1.1.3. Get Rocks Software

If you are building an x86 cluster, download the Rocks Base bootable CD and the HPC roll found in Software for x86. Then find 2 blank CD-Rs and burn both images to their respective media.

If you are building an IA-64 cluster, download the Rocks Base + HPC DVD Software for ia64. Then burn the image onto a blank DVD-R.

If you are building an x86_64 (Opteron) cluster, download the Rocks Base bootable CD and the HPC roll found in Software for x86_64. Then find 2 blank CD-Rs and burn both images to their respective media.

1.2. Install and Configure Your Frontend

This section describes how to install your Rocks cluster frontend with the Rocks Base CD coupled with the HPC Roll CD.

⬛ The minimum requirement in order to bring up a frontend is to have the Rocks Base CD and the HPC Roll CD. That is, using only the Rocks Base CD will result in a non-functional frontend.

1. Insert the Rocks Base CD into your frontend machine and reset the frontend machine.

2. After the frontend boots off the CD, you will see the boot screen:

When you see the screen above, type:

```
frontend
```

⬛ The "`boot:`" prompt arrives and departs the screen quickly. It is easy to miss. If you do miss it, the node will assume it is a *compute* appliance, and the frontend installation will fail and you will have to restart the installation (by rebooting the node).

If the installation fails, very often you will see a screen that complains of a missing /tmp/ks.cfg kickstart file. To get more information about the failure, access the kickstart and system log by pressing Alt-F3 and Alt-F4 respectively.

3. After you type `frontend` the installer will start running. Soon, you'll see a screen that looks like:
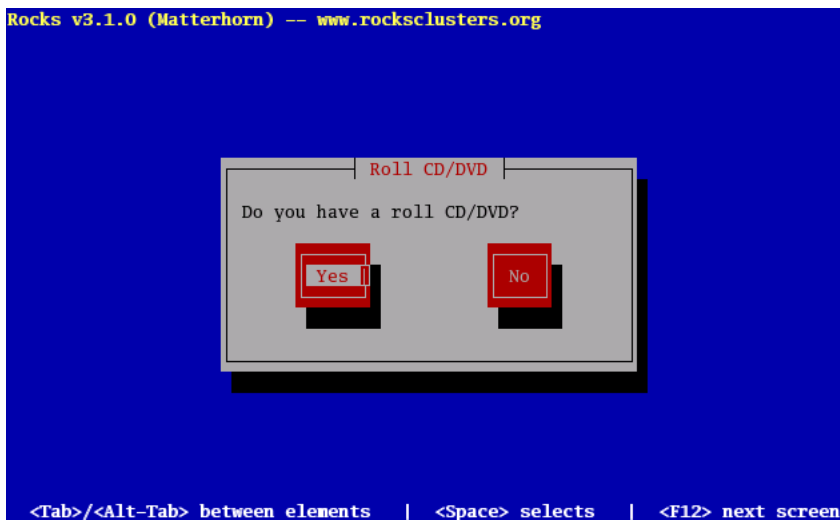
To add the HPC Roll, select 'Yes' by pressing the space bar.

4. After the CD/DVD drive ejects the Rocks Base media, put the HPC Roll CD into the drive.

To install the HPC Roll, select 'Ok' by pressing the space bar.

5. When the HPC Roll is discovered, you'll see window with the message:

Found Roll 'hpc'

6. Then you are asked if you have another Roll to add.
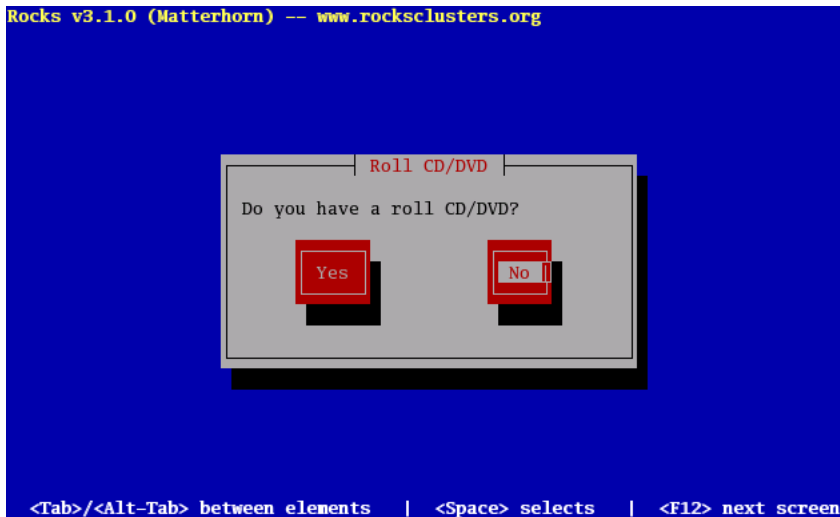
If you are just adding the HPC roll, then answer 'No' by hitting the 'Tab' key (which highlights the 'No' button) then hit the space bar.

If you adding another Roll, hit the spacebar and the CD will eject. Remove the HPC roll and put in the next Roll and answer 'Yes'. (This is the same procedure as described in Step 4 for the HPC roll).

7. After you've added the rolls, you'll be asked to put the Rocks Base CD back into the drive.

   Put the Rocks Base CD back into the drive, then answer 'Ok' by hitting the space bar.

   If you are building an IA64-based frontend, you may see an error message on the screen regarding `volname` and the installation will continue to ask for the Rocks Base DVD. This occurs because the DVD drive in the frontend is a low-cost drive and it requires a driver to be preloaded in order to support the `volname` command.

   A workaround is to reboot the frontend with the Rocks Base DVD, but this time at the ELILO boot prompt, type:

   ```
   frontend driverload=ide-scsi hdb=ide-scsi
   ```

   If your DVD is recognized by Linux as `hda`, you'll have to change the above line to:

   ```
   frontend driverload=ide-scsi hda=ide-scsi
   ```

8. Then you'll see the *Cluster Information* screen:

```
Rocks v3.1.0 (Matterhorn) -- www.rocksclusters.org
┤ Cluster Information ├
   These fields are optional, but fill in as many as possible.

Cluster Name:   The name of this cluster.
Our Cluster_____
Owner:          The organization that owns this cluster.
Production_____
Contact:        The administrative contact (email).
admin@place.org_____
URL:            The website for this cluster.
http://www.place.org/_____
LatLong:  Your Latitude and Longitude (like 'N32.87 W117.22')
unspecified_____

            OK        Back

<Tab>/<Alt-Tab> between elements  |  <Space> selects  |  <F12> next screen
```

This information is used by Ganglia to uniquely identify this cluster.

This information is optional.

Fill out the form (or take the defaults) then hit the 'Ok' button.

9. The disk partitioning screen allows you to select *automatic* or *manual* partitioning.

```
Rocks v3.1.0 (Matterhorn) -- www.rocksclusters.org
┤ Disk Partitioning Setup ├
Automatic Partitioning sets partitions based on the
selected installation type. You also can customize
the partitions once they have been created.

The manual disk partitioning tool, Disk Druid,
allows you to create partitions in an interactive
environment. You can set the file system types,
mount points, partition sizes, and more.

   Autopartition      Disk Druid      Back

<Tab>/<Alt-Tab> between elements  |  <Space> selects  |  <F12> next screen
```

To select automatic partitioning, hit the `Autopartition` button. This will partition the frontend like:

Table 1-1. Frontend -- Default Root Disk Partition

Partition Name	Size
/	6 GB
swap	1 GB
/export	*remainder of root disk*

To manually partition your frontend machine, select `Disk Druid`.

> If you select manual partitioning, you must specify at least 6 GBs for the root partition and you must create a separate `/export` partition.

Automatic partitioning is the default (and recommended).

10. The private cluster network configuration screen allows you to set up the networking parameters for the ethernet network that connects the frontend to the compute nodes.

```
Rocks v3.1.0 (Matterhorn) -- www.rocksclusters.org



                      ┤ Network Configuration for eth0 ├

        Network Device: eth0 - (private cluster network)
        Description: Intel Corp.|82546EB Gigabit Ethernet Controller (Copper)

        [ ] Configure using DHCP
        [*] Activate on boot

        IP Address            10.1.1.1_____
        Netmask               255.0.0.0_____


              ┌────┐                        ┌──────┐
              │ OK │                        │ Back │
              └────┘                        └──────┘




    <Tab>/<Alt-Tab> between elements  |  <Space> selects  |  <F12> next screen
```
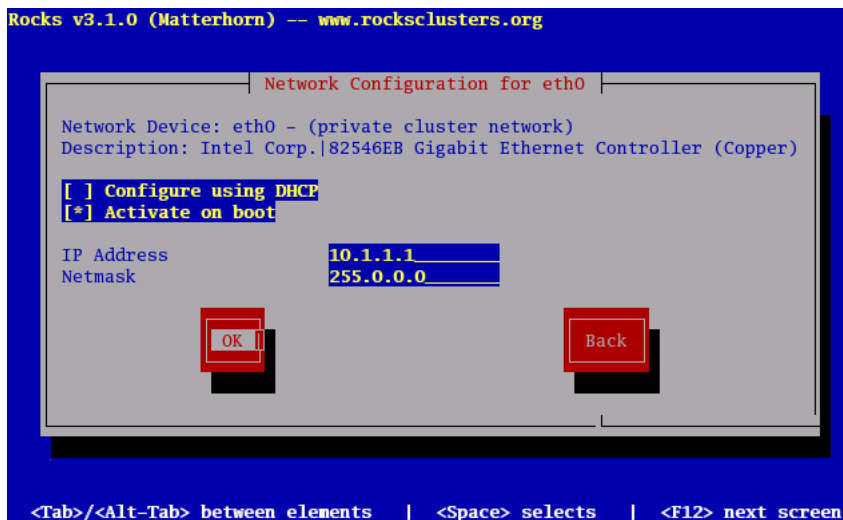
> It is recommended that you accept the defaults (by hitting the Tab key until the Ok button is highlighted, then hitting the Enter key).

But for those who have unique circumstances that requires different values for the internal ethernet connection, we have exposed the network configuration parameters.

11. The public cluster network configuration screen allows you to set up the networking parameters for the ethernet network that connects the frontend to the outside network (e.g., the internet).

```
Rocks v3.1.0 (Matterhorn) -- www.rocksclusters.org


         ┌──────────────┤ Network Configuration for eth1 ├──────────────┐

          Network Device: eth1 - (public cluster network)
          Description: Intel Corp.|82546EB Gigabit Ethernet Controller (Copper)

          [ ] Configure using DHCP
          [*] Activate on boot

          IP Address              198.202.77.60___
          Netmask                 255.255.255.0___

                   ┌────┐                          ┌──────┐
                   │ OK │                          │ Back │
                   └────┘                          └──────┘




  <Tab>/<Alt-Tab> between elements   |   <Space> selects   |   <F12> next screen
```
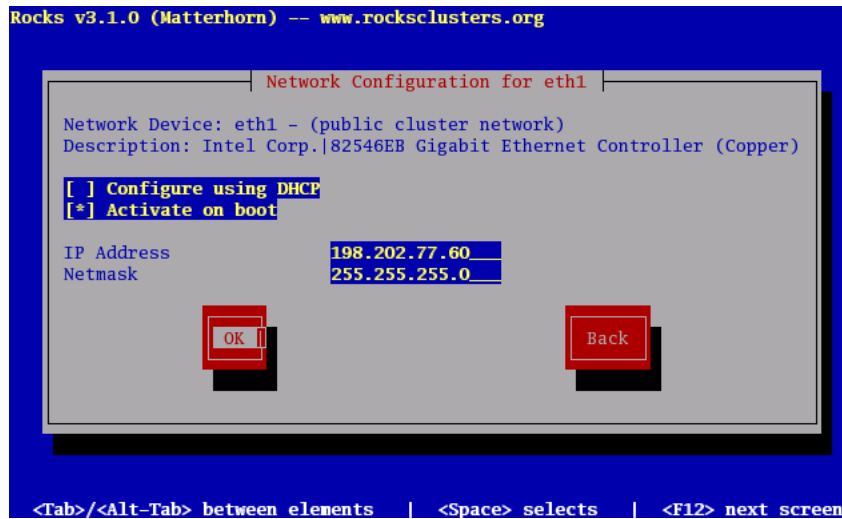
The above window is an example of how we configured the external network on one of our frontend machines.

12. Configure the the *Gateway* and *DNS* entries:

```
Rocks v3.1.0 (Matterhorn) -- www.rocksclusters.org




               ┤ Miscellaneous Network Settings ├

                  Gateway:        198.202.74.8___
                  Primary DNS:    198.202.75.26___
                  Secondary DNS:  _____
                  Tertiary DNS:   _____

                     ┌────┐        ┌──────┐
                     │ OK │        │ Back │
                     └────┘        └──────┘




  <Tab>/<Alt-Tab> between elements   |   <Space> selects   |   <F12> next screen
```
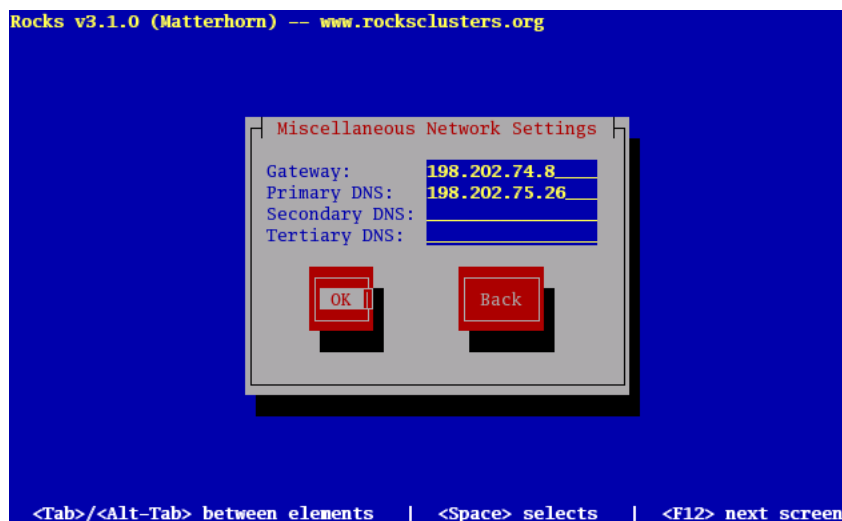
13. Configure the name for your frontend:
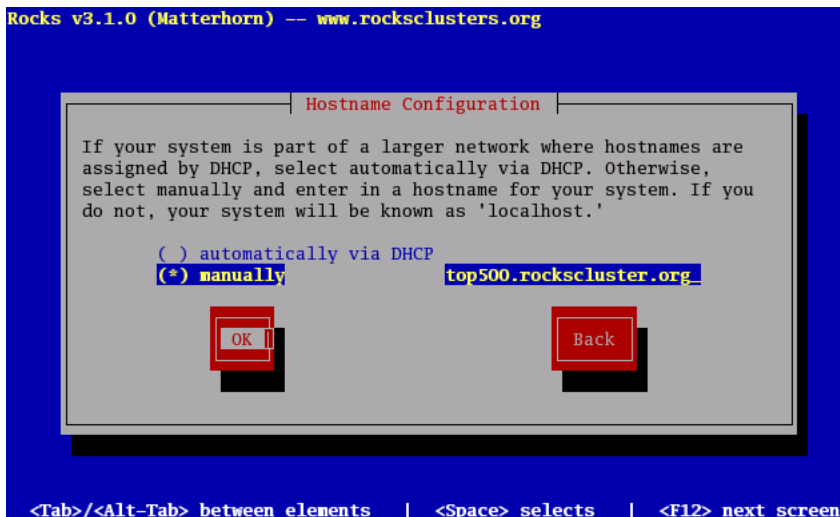
The name can be a fully qualified domain name or just the hostname portion of a fully qualified domain name (e.g., like the default `frontend-0`). If you plan on adding the Grid Roll (or other Globus PKI services) the hostname must be the primary FQDN for your host.

   Choose your hostname carefully. The hostname is written to dozens of files on both the frontend and compute nodes, if the hostname is changed cluster services will no longer be able to find the frontend machine. Some of these services include: SGE, Globus, NFS, AutoFS, and Apache.

   Do not choose to name your frontend "automatically via DHCP". This will prevent the frontend's DNS server from being included in name resolve path and will cause major problems in your cluster.

   If you absolutely need to use DHCP, be sure to add the line 'nameserver 127.0.0.1' to `/etc/resolv.conf`.

14. Input the root password:

```
Rocks v3.1.0 (Matterhorn) -- www.rocksclusters.org

                    ┤ Root Password ├

         Pick a root password. You must type it
         twice to ensure you know what it is and
         didn't make a mistake in typing. Remember
         that the root password is a critical part
         of system security!

         Password:          ****************____
         Password (confirm): ****************____

                OK                    Back


     <Tab>/<Alt-Tab> between elements  |  <Space> selects  |  <F12> next screen
```
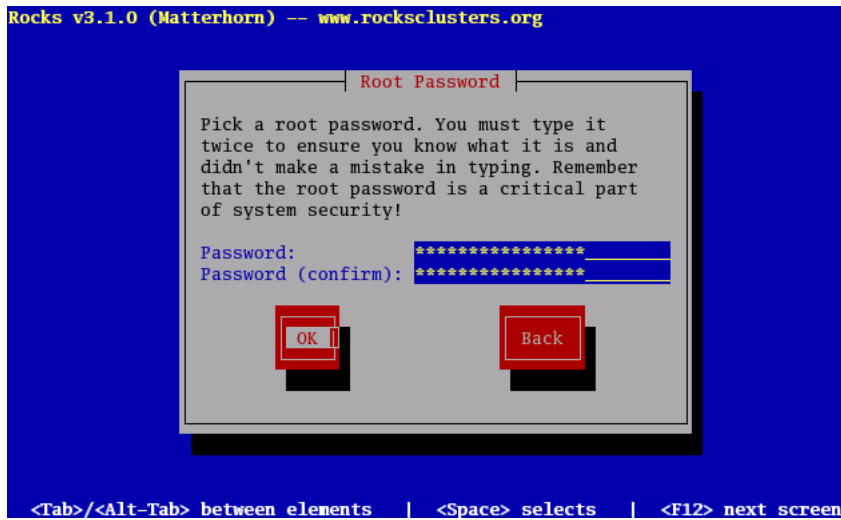
15. The frontend will format it's file systems:

```
Rocks v3.1.0 (Matterhorn) -- www.rocksclusters.org




                    ┤ Formatting ├

         Formatting /export file system...

                            11%



     <Tab>/<Alt-Tab> between elements  |  <Space> selects  |  <F12> next screen
```
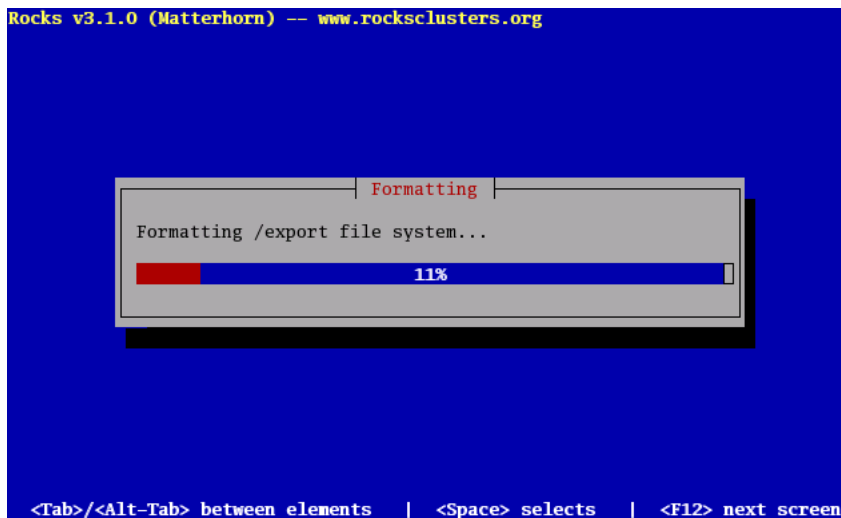
16. Then install the base packages:

```
Rocks v3.1.0 (Matterhorn) -- www.rocksclusters.org


                        ┤ Package Installation ├

    Name   : libvorbis-1.0-7-i386
    Size   : 1116k
    Summary: The Vorbis General Audio Compression Codec.


    ┌─────────────────────────────────────────────────────┐
    │                        0%                            │
    └─────────────────────────────────────────────────────┘
                      Packages        Bytes         Time
    Total    :             647        1888M       0:02:06
    Completed:             101         282M       0:00:18
    Remaining:             546        1606M       0:01:47

    ┌─────────────────────────────────────────────────────┐
    │█████████            14%                              │
    └─────────────────────────────────────────────────────┘



  <Tab>/<Alt-Tab> between elements   |   <Space> selects   |   <F12> next screen
```

And the installer will copy the contents of the Rocks Base to the hard disk on the frontend.

17. Then the installer will ask for each of the roll CDs you added at the beginning of the frontend installation.

    Put the appropriate roll CD in the drive when prompted and hit 'Ok'.

18. After the last roll CD is installed, the machine will reboot.

    We have observed an intermittent problem when the frontend reboots. The frontend will hang and display
    the error message: *GRUB Loading Stage2....* To resolve this error, see Grub Stage2 Error.

1.3. Install Your Compute Nodes

1. Login to the frontend node as `root`.

2. Run a program which captures compute node DHCP requests and puts their information into the Rocks MySQL database:

   ```
   # insert-ethers
   ```

   This presents a screen that looks like:
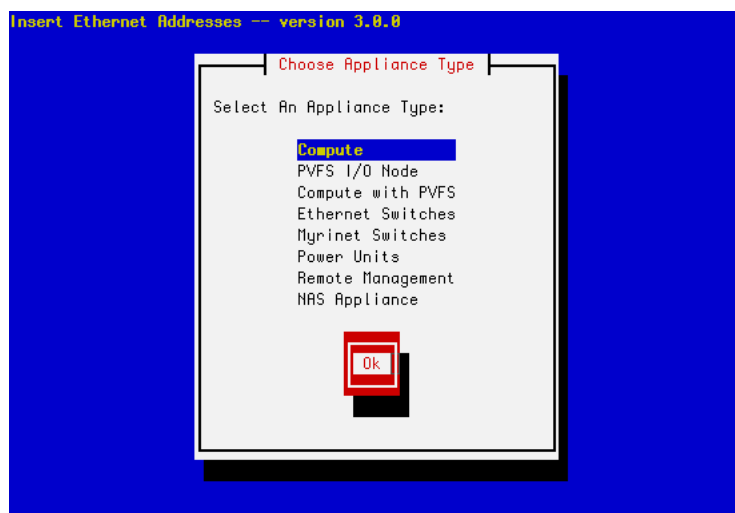
   ⚠ If your your frontend and compute nodes are connected via a managed ethernet switch, you'll want to select 'Ethernet Switches' from the list above. This is because the default behavior of many managed ethernet switches is to issue DHCP requests in order to receive an IP address that clients can use to configure and monitor the switch.

   When `insert-ethers` captures the DHCP request for the managed switch, it will configure it as an ethernet switch and store that information in the MySQL database on the frontend.

   As a side note, you may have to wait several minutes before the ethernet switch broadcasts its DHCP request. If after 10 minutes (or if `insert-ethers` has correctly detected and configured the ethernet switch), then you should quit `insert-ethers` by hitting the `F1` key.

   Now, restart `insert-ethers` and continue reading for a procedure on how to configure your compute nodes.

   Take the default selection, `Compute`, hit 'Ok'.

3. Then you'll see:
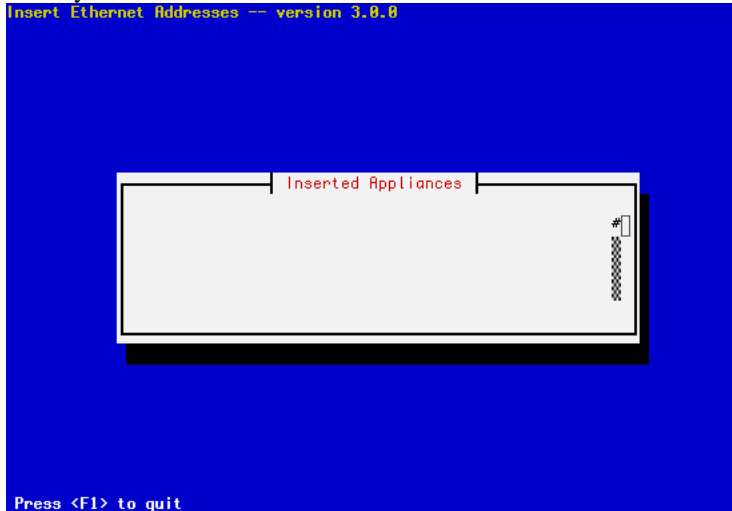
This indicates that `insert-ethers` is waiting for new compute nodes.

4. Take the CD (the same one you used to install the frontend) and put it in your first compute node -- this is the bottom compute node in your first cabinet.

> If you don't have a CD drive in your compute nodes, you can use PXE (Network Boot).

> If you don't have a CD drive in your compute nodes and if the network adapters in your compute nodes don't support PXE, see Using a Floppy to PXE boot.

5. Power up the first compute node.

6. When the frontend machine receives the DHCP request from the compute node, you will see something similar to:
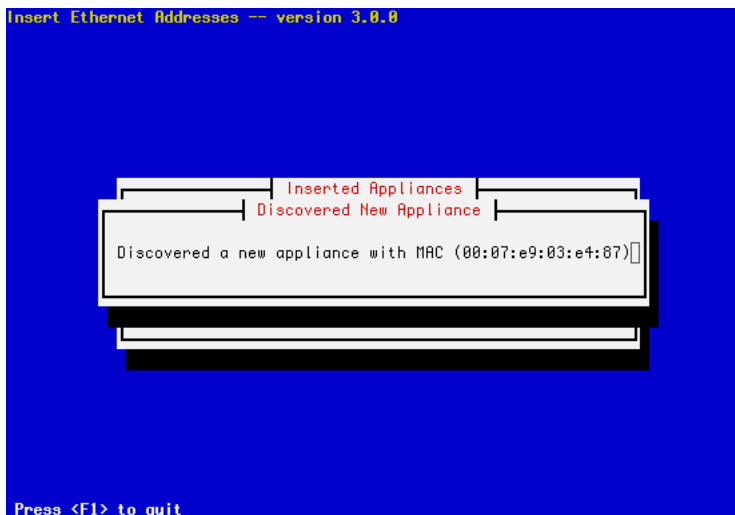
This indicates that `insert-ethers` received the DHCP request from the compute node, inserted it into the database and updated all configuration files (e.g., `/etc/hosts, /etc/dhcpd.conf`, PBS and Maui files, and NIS).

The above screen will be displayed for a few seconds and then you'll see the following:
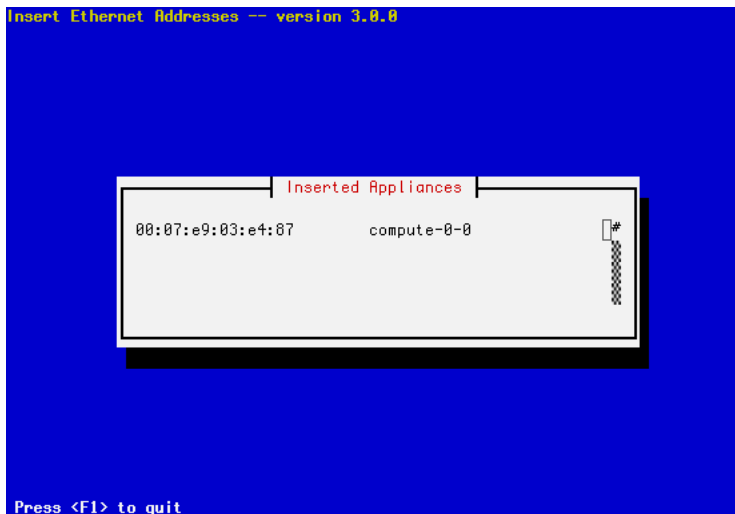
You will see this type of output for each compute node that is successfully identified by `insert-ethers`.

7. At this point, you can monitor the installation by using `ssh`. Just extract the name of the installing compute node from the `insert-ethers` output (in the example above, the compute node name is `compute-0-0`), and

execute:

```
# ssh compute-0-0
```

8. When the installation is complete, the CD will eject. Take the CD out of the tray and put it into the next compute node above the one you just installed and hit the power button.

9. After you've installed all the compute nodes in a cabinet, quit `insert-ethers` by hitting the 'F1' key.

10. After you've installed all the compute nodes in the first cabinet and you wish to install the compute nodes in the next cabinet, just start `insert-ethers` like:

```
# insert-ethers --cabinet=1
```

This will name all new compute nodes like `compute-1-0`, `compute-1-1`, ...

1.4. Cross Kickstarting

Rocks supports heterogeneous clusters that contain nodes of different hardware architectures with a process called cross-kickstarting. To support an architecture different than its own, a frontend needs to expand its local distribution with additional packages. This section describes how to install distributions for other architectures on your frontend.

While the cross-kickstarting process is identical for all architectures, problems exist when rebuilding an ia64 distribution on an x86 frontend. Therefore for the 3.2.0 release, cross-kickstarting is limited to x86 and x86_64 architectures.

Start with a frontend node, as described by Install Frontend, or upgrade frontend. Follow the instructions below for every desired architecture.

For this example, we assume the "x86_64" architecture, and the "3.2.0-0" Rocks version. The frontend is an x86.

1. Retrieve the Rocks base CD and any desired rolls for x86_64 (see downloads).

Mount the Rocks base CD on /mnt/cdrom. This can be done without actually burning the CD, using the command: `mount -o loop rocks-base-3.2.0-0.x86_64.iso /mnt/cdrom`. Then copy its contents into the local mirror:

```
# rocks-dist --arch=x86_64 copycd
```

2. Unmount the base CD image. For each x86_64 roll, mount it to /mnt/cdrom as above, then copy its contents to the appropriate location with:

```
# rocks-dist copyroll
```

3. Rebuild your distribution for the new architecture.

```
# cd /home/install
# rocks-dist --arch=x86_64 dist
```

This requires that you have built the distribution for your native architecture first.

Now your frontend is prepared to cross-kickstart compute nodes and other cluster appliances of different architectures.

The Rocks 3.2.0 Shasta release does not support PXE cross-kickstart installs; you must boot non-native compute nodes from a native-architecture Rocks boot disk or Rocks base CD. Non-native means the compute node has a different architecture than its frontend.

1.5. Upgrade Your Existing Frontend

If you already have a Rocks cluster, you may choose to upgrade the existing frontend. Below describes the procedure (as well as an explanation of some of the internals).

- Prep your compute nodes for PXE installation.

  While the cluster is on-line and before you upgrade the frontend, first prep the compute nodes for a PXE installation. If your compute nodes support PXE installs, then execute the following commands to ensure the compute nodes will PXE the next time they boot:

```
# ssh-agent $SHELL
# ssh-add
# cluster-fork 'touch /boot/grub/pxe-install'
# cluster-fork '/boot/kickstart/cluster-kickstart --start'
# cluster-fork '/sbin/chkconfig --del rocks-grub'
```

Now you can shutdown your compute nodes. After the frontend upgrade is complete, you'll run `insert-ethers` and then turn on your compute nodes one at a time. This is similar to the first time you brought up your cluster, but this time you don't have to initially boot each compute node with the CD or manually force a PXE boot by interacting with each compute node's BIOS screens. This procedure should allow you to upgrade your cluster faster than when you initially installed it.

If your compute nodes don't support PXE, then you will have to use the CD to upgrade your compute nodes after the frontend upgrade is completed.

- Insert the Rocks installation CD into your frontend machine and reboot the frontend machine.

- When you see the "boot:" prompt, type:

  ```
  frontend upgrade
  ```

- At this point, the installation follows the same steps as a *normal* frontend installation (See the section: Install Frontend) -- with one exception:

  🛑 You must manually configure your partitions. Additionally, you must reformat your `/` partition and your `/boot` partition (if it exists).

1.5.1. Frontend Upgrade Internals

We have added code to the Red Hat installer to support upgrading a frontend. Early in the upgrade process (before the root partition is reformatted), our code mounts the existing root partition and extracts the following files:

```
/etc/passwd
/etc/shadow
/etc/gshadow
/etc/group
/etc/auto.home
/etc/fstab
/etc/exports
```

After all the packages have been installed, our code then merges the contents of the old files with the new files that are now in the root partition. If there is an common entry in both the old and new files, the entry from the new file is put in the resulting merged file and the old entry is discarded.

All non-root partitions will be preserved and remounted when the frontend completes its upgrade. For example, a standard Rocks frontend has a root partition and a partition named `/export`. The root partition will be reformatted, while the `/export` will remain intact and will be automatically remounted once the frontend completes its upgrade.

1.6. Installing a Frontend over the Network

This section describes installing a Rocks frontend from a Central server over the wide area network.

1. Use a Rocks CD (see downloads) to boot the frontend, or alternatively a small bootdisk (also available on the downloads page). When the splash screen appears, type:

   ```
   frontend central=name.your.org
   ```

   at the `boot:` prompt, where "name.your.org" is the fully-qualified domain name of your central server, without http:// or any other prefix or suffix. If you use the special name "Rocks", you will attempt to contact `central.rocksclusters.org`, a central server at SDSC.

   You can also upgrade your frontend over the network, by adding the keyword `upgrade` boot command (in addition to `frontend central=name`).

   Advanced users may want to activate Rocks *ekv* service on the frontend during the install. Ekv allows you to interact with the install screens over the network. To do this, add the keyword `ekv` to the boot command. Once the install has begun you can ssh to the frontend. When connected, you will see the RedHat and Rocks install screens. Unfortunately, when ekv is activated, the screens on the physical console (the monitor that is physically connected to the frontend) will get mangled to some extent.

2. After the frontend boots off the central, you will see the roll choice screen:
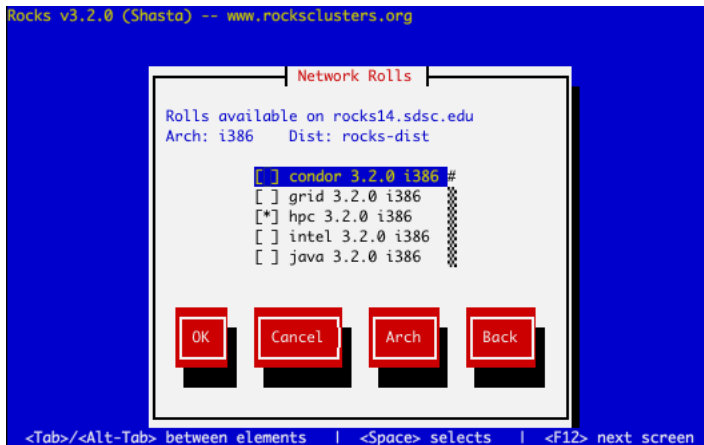
Choose your desired rolls, and press "Ok".

> If you need a roll for another architecture, choose your rolls for this arch first, press ok. Then say "yes" to another roll server, and choose the same one as before. You will see this screen again. Press the "arch" button, and choose rolls for the non-native architecture.

3. You will then see the option to choose another central server to get more rolls.
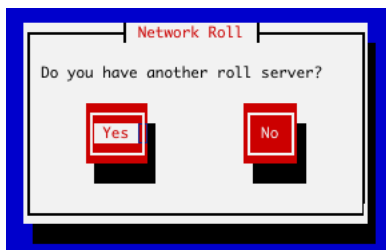
When you are finished adding new network rolls (ones obtained over the network), answer "No" to the new central server question.

4. Now you can augment your frontend with rolls from a CD. The screens and process for installing CD-based rolls is described in Install Frontend.

5. When you are finished installing CD-based rolls, you will enter into the familiar Rocks installation windows. These may change depending on what rolls you have selected. Again the Install Frontend section has details for this process.

6. The contents of the Rocks Base CD will now be transferred to your frontend over the network. Keep in mind that this may involve over 650MB of transfers, which are done via http using standard wget. This phase will take some time, and is affected by network congestion and bandwidth between you and the central server.

   Anaconda will then install the base packages, followed by the retrieval of network rolls from each central server where they reside. Rocks will then install the rolls on your frontend, then proceed with the post-section and other elements of a normal frontend install.

Your frontend should now be installed and ready to initialize compute nodes (see section install compute nodes).

Advanced users should know that there exists another `boot:` option for frontend central. If you have multiple distributions on central (i.e. one for english, one for italian), you may specify `dist=name`, such as: "boot: frontend central=name.my.org dist=it-dist". This specifies the top-level distribution directory, that central will expect to find at "/home/install/external/it-dist".

In addition, if you simply type `frontend central` (no equal sign), the frontend will attempt to obtain its kickstart file from any DHCP server that will answer. With such a dhcpd server, you can even PXE the frontend!

1.7. Make a Frontend into a Central Server

A Central Server is a Rocks Frontend node that can kickstart other frontends and provide rolls over the network. By default, all Frontends are central servers. This section describes the steps required to prep a frontend for central service.

The Wide-Area Kickstarting process is a newly-introduced advanced feature. Its interface is not polished, and will likely change in future releases.

Start with a frontend node, as described by Install Frontend, or upgrade frontend.

1. Make a central distribution.

```
# cd /home/install/external
# ./makecentral.sh
```

   The makecentral.sh script simply runs the `rocks-dist central` command.

2. Open www access to the node (see Enable WWW access).

3. Open web access to the `install/external` web directory. Although the kickstart file given to external frontends is minimal and does not contain passwords, it is not advisable to open access to this directory without considering its security implications. When access to this directory is granted, your rocks distribution (and all its packages) is visible to external hosts on the Internet.

   The access necessary is affected by Apache server directives. Apache provides advanced access control via ACL and other mechanisms. We recommend familiarizing yourself with these features. However, the commands below allow you to quickly open access to everyone for testing.

   Edit the `/etc/httpd/conf/rocks-central.conf` file. Uncomment the line as indicated in the file.

```
<Directory /var/www/html/install/external>
        Options FollowSymLinks Indexes ExecCGI
        AllowOverride None
        Order deny,allow
        Allow from 10.0.0.0/255.0.0.0
        Allow from 127.0.0.1
        # Uncomment the following line to activate this central (insecure).
        #Allow from all
        Deny from all
</Directory>
```

Ensure that the hostname on central is fully qualified. Specifically, the `PublicHostname` value in the `app_globals` table of the database must be correct and reachable from the outside world.

Your Frontend node should now be ready to function as a central server.

1.7.1. Adding Rolls

To add a roll to a central server, you do not need to reinstall the central. To make additional rolls available for external frontend consumption, simply:

```
1. Insert roll CD in drive, mount it as /mnt/cdrom.

2. # rocks-dist copyroll
```

Roll developers can install their roll on central directly. This process assumes you have built your roll on the central server.

```
# make roll
# mount -o loop roll-*.iso /mnt/cdrom
# rocks-dist copyroll
```

From your `/rocks/src/roll/[name]` directory.

Notes

1. http://www.pgroup.com

2. http://www.oreilly.com

3. images/Meteor10-2000.png

Chapter 2. Start Computing

2.1. Launching Interactive Jobs

2.1.1. Using mpirun

Mpirun on Rocks clusters is used to launch jobs that are linked with the Ethernet device for MPICH.

You must run HPL as a regular user (that is, not root).

If you don't have a user account on the cluster, create one for yourself with:

```
# useradd username
```

For example, to interactively launch the benchmark "High-Performance Linpack" (HPL) on two processors:

*   Create a file in your home directory named `machines`, and put two entries in it, such as:

    ```
    compute-0-0
    compute-0-1
    ```

*   Download the the two-processor HPL configuration file[1] and save it as `HPL.dat` in your home directory.
*   Now launch the job from the frontend:

    ```
    $ /opt/mpich/gnu/bin/mpirun -nolocal -np 2 -machinefile machines /opt/hpl/gnu/bin/xhpl
    ```

2.1.2. Using mpirun for Myrinet

Mpirun for Myrinet on Rocks clusters is used to launch jobs that are linked with the Myrinet device for MPICH.

You must run HPL as a regular user (that is, not root).

If you don't have a user account on the cluster, create one for yourself with:

```
# useradd username
```

For example, to interactively launch the benchmark "High-Performance Linpack" (HPL) on two processors:

*   Create a file in your home directory named `machines`, and put two entries in it, such as:

    ```
    compute-0-0
    compute-0-1
    ```

*   Download the the two-processor HPL configuration file[2] and save it as `HPL.dat` in your home directory.
*   Now launch the job from the frontend:

    ```
    $ /opt/mpich/myrinet/gnu/bin/mpirun -np 2 -machinefile machines /opt/hpl/myrinet/gnu/bin/xhpl
    ```

2.1.3. Cluster-Fork

Often we want to execute parallel jobs consisting of standard UNIX commands. By "parallel" we mean the same command runs on multiple nodes of the cluster. We use these simple parallel jobs to move files, run small tests, and to perform various administrative tasks.

Rocks provides a simple tool for this purpose called `cluster-fork`. For example, to list all your processes on the compute nodes of the cluster:

```
$ cluster-fork ps -U$USER
```

By default, cluster-fork uses a simple series of ssh connections to launch the task serially on every compute node in the cluster. Cluster-fork is smart enough to ignore dead nodes. Usually the job is "blocking": cluster-fork waits for the job to start on one node before moving to the next. By using the `--bg` flag you can instruct cluster-fork to start the jobs in the background. This corresponds to the "-f" ssh flag.

```
$ cluster-fork --bg hostname
```

Often you wish to name the nodes your job is started on. This can be done by using an SQL statement or by specifying the nodes using a special shorthand.

The first method of naming nodes uses the SQL database on the frontend. We need an SQL statement that returns a column of node names. For example, to run a command on compute nodes in the first rack of your cluster exectute:

```
$ cluster-fork --query="select name from nodes where name like 'compute-1-%'" [cmd]
```

The next method of requires us to explicitly name each node. When launching a job on many nodes of a large cluster this often becomes cumbersome. We provide a special shorthand to help with this task. This shorthand, borrowed from the MPD job launcher, allows us to specify large ranges of nodes quickly and concisely.

The shorthand is based on similarly-named nodes and uses the `--nodes` option. To specify a node range *compute-0-0 compute-0-1 compute-0-2*, we write *--nodes=compute-0-%d:0-2*. This scheme works best when the names share a common prefix, and the variables between names are numeric. Rocks compute nodes are named with such a convention.

Other shorthand examples:

- Discontinuous ranges:

  compute-0-%d:0,2-3 --> `compute-0-0 compute-0-2 compute-0-3`

- Multiple elements:

  compute-0-%d:0-1 compute-1-%d:0-1 --> `compute-0-0 compute-0-1 compute-1-0 compute-1-1`

- Factoring out duplicates:

  *2*compute-0-%d:0-1 compute-0-%d:2-2* --> `compute-0-0 compute-0-0 compute-0-1 compute-0-1 compute-0-2`


```
$ cluster-fork --nodes="compute-2-%d:0-32 compute-3-%d:0-32" ps -U$USER
```

The previous example lists the processes for the current user on 64 nodes in racks two and three.

The `--nodes` option requires that your nodes are named "compute-X-Y", where X is the rack number, and Y is the node's rank in the rack. If you change the naming prefix for compute nodes, this method will not work.

2.1.3.1. Cluster-Fork and MPD

Cluster-fork can now use the MPD ring to start any task. To activate this option, give the `--mpd` flag to cluster-fork.

```
$ cluster-fork --mpd ps -U$USER
```

Our hope is that these new options to cluster-fork will be useful to you. Be aware that when using --mpd, output lines will not necessarily arrive in order to your console. If you would like to make cluster-fork always use the MPD ring, set the environment variable:

```
ROCKS_JOB_LAUNCHER=mpd
```

2.2. Launching Batch Jobs Using Grid Engine

This section describes instructions and simple scripts we've developed to launch batch scheduled jobs using Grid Engine on Rocks clusters.

Jobs are submitted to Grid Engine via scripts. Here is an example of a Grid Engine script, sge-qsub-test.sh[3] that we use to test Grid Engine. It asks Grid Engine to launch the MPI job on two processors (line 5: `#$ -pe mpi 2`). The script then sets up a temporary ssh key that is used by `mpirun` to instantiate the program (in this case, the program is `xhpl`).

You can submit the job to Grid Engine by executing:

```
qsub sge-qsub-test.sh
```

After the job is launched, you can query the status of the queue by running:

```
qstat -f
```

Grid Engine puts the output of job into 4 files. The 2 files that are most relevant are:

```
$HOME/sge-qsub-test.sh.o<job id>
```

(stdout messages) and

```
$HOME/sge-qsub-test.sh.e<job id>
```

(stderr messages).

The other 2 files pertain to Grid Engine status and they are named:

```
$HOME/sge-qsub-test.sh.po<job id>
```

(stdout messages) and

```
$HOME/sge-qsub-test.sh.pe<job id>
```

(stderr messages).

2.2.1. Cluster-Fork and SGE

SGE, the default Batch System on Rocks clusters, will allocate you a set of nodes to run your parallel job. It will not, however, launch them for you. Instead SGE sets a variable called `$PE_HOSTFILE` that names a file with a set of nodes listed within. In the mpi parallel environment, a special start script parses this file and starts the mpirun launcher. However, if you need to start a non-MPI job via SGE, cluster-fork can help. (See Section 2.1.3 for details on cluster-fork).

Cluster-fork can interpret the PE_HOSTFILE given by SGE. The `--pe-hostfile` option is used for this purpose. For example, to start the 'hostname' command on all nodes allocated by SGE:

```
/opt/rocks/bin/cluster-fork --bg --pe-hostfile $PE_HOSTFILE hostname
```

2.3. Using the High-Performance MPD Job Launcher

MPD is a new high-performance job launcher developed by Argonne National Laboratory, the makers of MPICH. It serves as a drop-in replacement to mpirun, and can be used to launch parallel jobs. MPD can start both MPI and non-MPI parallel applications.

Advantages of MPD:

- Faster Launching. Can start a job on 100 nodes in less than one second.
- Cleanup after Jobs. MPD propagates ^C and ^Z (SIGTERM and SIGINT) signals correctly, allowing us to stop a runaway job with one command on the frontend node.
- Fault Tolerance. MPD can start jobs and deliver signals even in the face of node failures.

Drawbacks of MPD:

- Compatibility. MPI applications must be recompiled to use the MPD job launcher. However, normal applications (those that do not use the MPI libary) do not, and can natively be started with MPD.

- Security. MPD does not use ssh to launch jobs or deliver signals, and does not attempt to encrypt commands. This level of security is only appropriate for clusters with a protected internal network. Rocks clusters built according to the suggestions in this manual fall into this category and can use MPD with relative safety.

- Complexity. MPD relies on a "ring" of daemons between cluster nodes which must be created and maintained.

- Speed. MPD will put more strain on the frontend's NFS file server than the previous launcher. Every node in the parallel job will request a copy of the application executable at nearly the same time, causing the NFS server to suffer (Remember all home directories are served via NFS). Larger jobs are more at risk of toppling the NFS server than small ones.

  Future versions of MPD may use the ring's efficient communication pipes to distribute the executable file itself.

2.3.1. Using MPD for MPI applications

To launch interactive and batch MPI applications, you must compile your program with the MPD version of the MPICH library. This library is identical to regular MPICH and supports the same interface. It is located in:

```
/opt/mpich-mpd/gnu/lib
```

Once your executable has been compiled with the MPD libraries, use the `mpirun` from:

```
/opt/mpich-mpd/gnu/bin/mpirun
```

This MPD version of mpirun operates very similarly to older versions. See the manpage or use --help for full details. You will be able to control all nodes running your job from the console node (usually the frontend), including sending signals to your parallel job.

See earlier sections in this chapter to find instructions for using mpirun in batch and interactive settings.

2.3.2. The MPD Ring and Troubleshooting

For the MPD job launcher to work correctly, a ring of daemons must be threaded through every node in the cluster. he ring is defined as a set of connected mpd daemons, such that each daemon has an open TCP connection to each of its two nearest neighbors. The order of nodes in the ring does not matter, and a new node can enter the ring at any place. Rocks uses a distributed agreement protocol to construct and maintain this ring through node failures and additions. This ring maintainer is called KAgreement-mpd, after a well-known problem in distributed systems.

If one daemon in the ring dies, the remaining MPD nodes will reknit the ring around it. The KAgreement-mpd protocol will restart the dead daemon as soon as possible, enabling it to rejoin the ring.

Once the ring of MPD daemons is in place, parallel jobs can be quickly started. The original ring is composed of mpd daemons running as root. Each has a local pipe that serves as a "console" on the node, through which commands are sent for starting and controlling jobs. Only one console pipe is allowed per job.

When the root ring receives a job-start command, it creates a child ring by forking "mpd" daemons running as the user. This ring lives only to service the job, and will be destroyed when execution completes. The job ring inherits the console pipe from the appropriate root daemon. This ring sets up a tree-based connection structure to handle standard input, output, error streams, as well as any MPI messages that may be sent. All output is collected in one place and sent back though the console pipe.

Without a complete root ring, no jobs can be started. The task of keeping the ring healthy depends on correct and reliable MPD daemons and KAgreement-mpd. Although stable in practice, KAgreement-mpd has not been extensively tested in production. In addition, the MPD daemons themselves are relatively new and untested.

The KAgreement-mpd protocol uses Ganglia-style multicast messages to coordinate every node in the cluster. These messages enable new nodes to join the ring automatically, and gracefully supports temporary network partitions. You can inspect these messages with:

```
$ telnet localhost 8649 | grep mpd
```

You should see two lines of output for every node in the cluster. KAgreement-mpd requires that the `greceptor` daemon (which publishes and listens for user-defined Ganglia metrics) be running on every node in the cluster.

Other ways of troubleshooting the MPD ring is to use the utilities mpdtrace, mpddump, mpdshutdown, mpdjobinfo, etc. Check the contents of `/opt/mpich-mpd/gnu/bin` for the full list of MPD utilities.

2.3.3. Conclusion

The `cluster-fork` command can also use the MPD system to run any standard UNIX command for speedy startup and efficient job control. See the Cluster Fork page for more details.

In conclusion, we have made efforts to provide a fully functioning MPD ring without any intervention or effort from the user. Please be patient with any bugs found, as the KAgreement-mpd protocol is not yet mature. For more information on MPD and its utilities, see manuals and associated documentation present on your frontend machine in "/opt/mpich-mpd/{www,doc,man}".

2.4. Running Linpack

2.4.1. Interactive Mode

This section describes ways to scale up a HPL job on a Rocks cluster.

To get started, you can follow the instructions on how run a two-processor HPL job at Using Mpirun on Ethernet. Then, to scale up the number of processors, add more entries to your `machines` file. For example, to run a 4-processor job over compute nodes `compute-0-0` and `compute-0-1`, put the following in your `machines` file:

```
compute-0-0
compute-0-0
compute-0-1
compute-0-1
```

Then you'll need to adjust the number of processors in `HPL.dat`:

change:

```
1 Ps
2 Qs
```

to:

```
2 Ps
2 Qs
```

The number of total processors HPL uses is computed by multiplying *P* times *Q*. That is, for a 16-processor job, you could specify:

```
4 Ps
4 Qs
```

And finally, you need adjust the `np` argument on the `mpirun` command line:

```
$ /opt/mpich/gnu/bin/mpirun -nolocal -np 4 -machinefile machines /opt/hpl/gnu/bin/xhpl
```

To make the job run longer, you need to increase the problem size. This is done by increasing the *Ns* parameter. For example, to quadruple the amount of work each node performs:

change:

```
1000 Ns
```

to:

```
2000 Ns
```

Keep in mind, doubling the `Ns` parameter *quadruples* the amount of work.

For more information on the parameters in `HPL.dat`, see HPL Tuning[4].

2.4.2. Batch Mode

This section describes ways to scale up a HPL job on a Rocks cluster when submitting jobs through Grid Engine.

To get started, you can follow the instructions on how to scale up a HPL job at Interactive Mode. To increase the number of processors that the job uses, adjust HPL.dat (as described in Interactive Mode). Then get the file `sge-qsub-test.sh` (as described in launching batch jobs), and adjust the following parameter:

```
#$ -pe mpi 2
```

For example, if you want a 4-processor job, change the above line to:

```
#$ -pe mpi 4
```

Then submit your (bigger!) job to Grid Engine.

Notes

1. examples/HPL.dat
2. examples/HPL.dat
3. examples/sge-qsub-test.sh

4.   http://www.netlib.org/benchmark/hpl/tuning.html

Chapter 3. Monitoring

3.1. Monitoring Your Cluster

A Rocks cluster presents a set of web pages to monitor its activities and configuration. The "frontend" node of the cluster serves these pages using its built in Apache webserver. This section describes the web-based monitoring tools available out of the box on all Rocks clusters.

For security, web access is restricted to only the internal cluster network by default. However, since usually only frontend and compute nodes (which have no monitors) reside on this network, some extra effort is required to view the monitoring web pages.

The easiest method of viewing the cluster pages is to attach a monitor, keyboard, and mouse to the frontend node of your cluster and configure its X window system.

```
# redhat-config-xfree86
# startx
```

Once this is done, a standard RedHat desktop environment will appear. Point a Mozilla or Konqueror web browser at the URL `http://localhost/` to view the cluster site.

3.1.1. Accessing Cluster Website using SSH Tunneling

The first method of viewing webpages involves sending a web browser screen over a secure, encrypted SSH channel. To to this, follow the steps below.

1. Log into the cluster's frontend node, and supply your password when requested.

   $ ssh mycluster

2. Ensure you have an X server running on your local machine. Start the Mozilla browser on the cluster with the following command. The ssh process will setup an encrypted channel for the mozilla window to operate through.

   $ mozilla --no-remote &

   We recommend using the Mozilla browser over Netscape Navigator. Mozilla handles stylesheets much more effectively, and has a better rendering engine.

3. Wait until the Browser window appears on your local machine. The the URL `http://localhost/` should appear with the cluster home page.

3.1.2. Enabling Public Web Access with Control Lists

To permenantly enable selected web access to the cluster from other machines on the public network, follow the steps below. Apache's access control directives will provide protection for the most sensitive parts of the cluster web site, however some effort will be necessary to make effective use of them.

HTTP (web access protocol) is a clear-text channel into your cluster. Although the Apache webserver is mature and well tested, security holes in the PHP engine have been found and exploited. Opening web access to the outside world by following the instructions below will make your cluster more prone to malicious attacks and breakins.

1. Edit the `/etc/sysconfig/iptables` file. Uncomment the line as indicated in the file.

```
...
-A INPUT -i eth1 -p tcp -m tcp --dport ssh -j ACCEPT
# Uncomment the line below to activate web access to the cluster.
#-A INPUT -i eth1 -p tcp -m tcp --dport www -j ACCEPT
... other firewall directives ...
```

2. Restart the iptables service. You must execute this command as the root user.

```
$ service iptables restart
```

3. Test your changes by pointing a web browser to `http://my.cluster.org/`, where "my.cluster.org" is the DNS name of your frontend machine.

If you cannot connect to this address, the problem is most likely in your network connectivity between your web browser and the cluster. Check that you can ping the frontend machine from the machine running the web browser, that you can ssh into it, etc.

3.1.3. Table of Contents Page

If you can successfully connect to the cluster's web server, you will be greeted with the Rocks *Table of Contents* page. This simple page has links to the monitoring services available for this cluster.
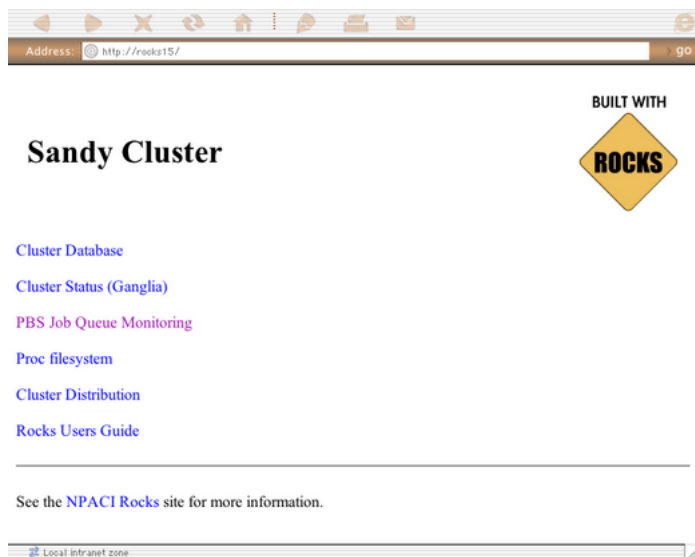
3.2. The Cluster Database

This web application allows you to view and edit the active Rocks SQL database. Rocks uses this database to store data about its configuration, and information about the nodes in this cluster. See the Rocks Cluster Schema[1] page for a description of this database's structure and semantics.
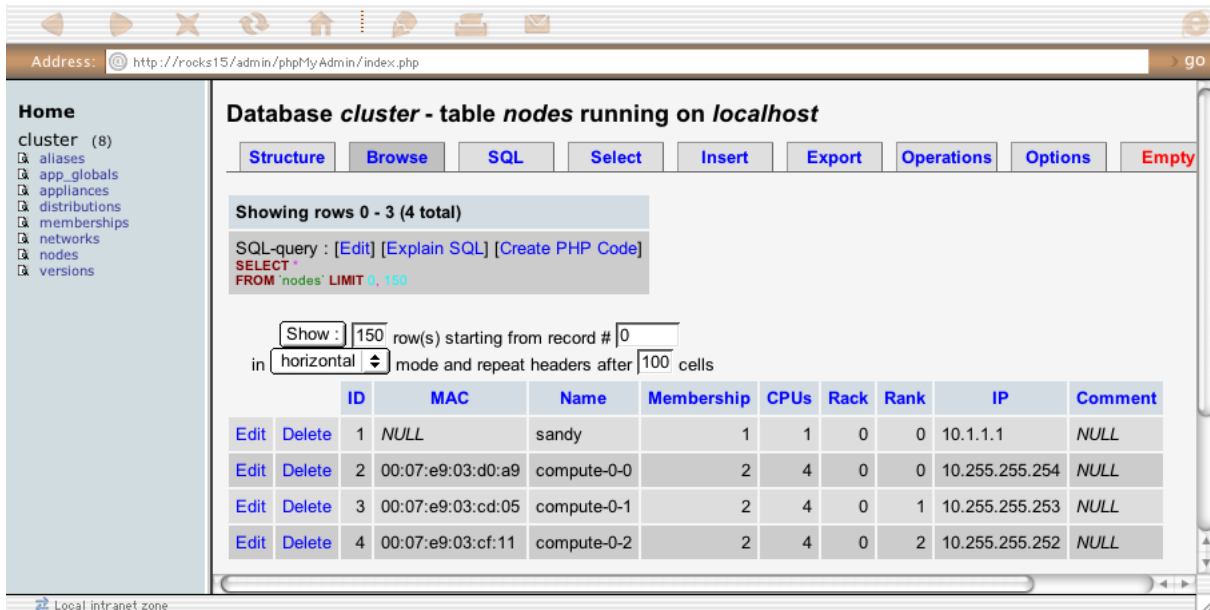
The web database application will allow Queries, Inserts, Updates, and Deletes to the active database. Any changes made via the web application will be immediately visible to any services that consult the database. Because of this ability, we restrict access to this page to only hosts on the internal network. To enable extended access to the database web application, edit the /etc/httpd/conf/rocks.conf file as follows.

```
<Directory "/var/www/html/admin/phpMyAdmin">
        Options FollowSymLinks Indexes ExecCGI
        AllowOverride None
        Order deny,allow
        Allow from 127.0.0.1
        Deny from all
</Directory>
```

Add additional "Allow" directives in this section to specify which additional hosts will be given access to the web database application. The format for the Allow directive is available in the Apache Manual[2].

The PhpMyAdmin[3] web application was designed by Tobias Ratschiller (tobias.ratschiller@maguma.com) in 1998, and is released through Sourceforge.net under the GPL liscence.

3.3. Cluster Status (Ganglia)

The webpages available from this link provide a graphical interface to live cluster information provided by Ganglia monitors[4] running on each cluster node. The monitors gather values for various metrics such as CPU load, free Memory, disk usage, network I/O, operating system version, etc. These metrics are sent through the private cluster network and are used by the frontend node to generate the historical graphs you see on this page.

In addition to metric parameters, a heartbeat message from each node is collected by the ganglia monitors. When a number of heartbeats from any node are missed, this web page will declare it "dead". These dead nodes often have problems which require additional attention, and are marked with the Skull-and-Crossbones icon, or a red background.

This page has many options, most of which are hopefully somewhat self explanitory. There are numerous links and each page shows a myriad of information, so be sure to explore the site carefully. The data is very fresh (usually only a few seconds old), and is updated with each page load. See the ganglia website for more information about this powerful tool.
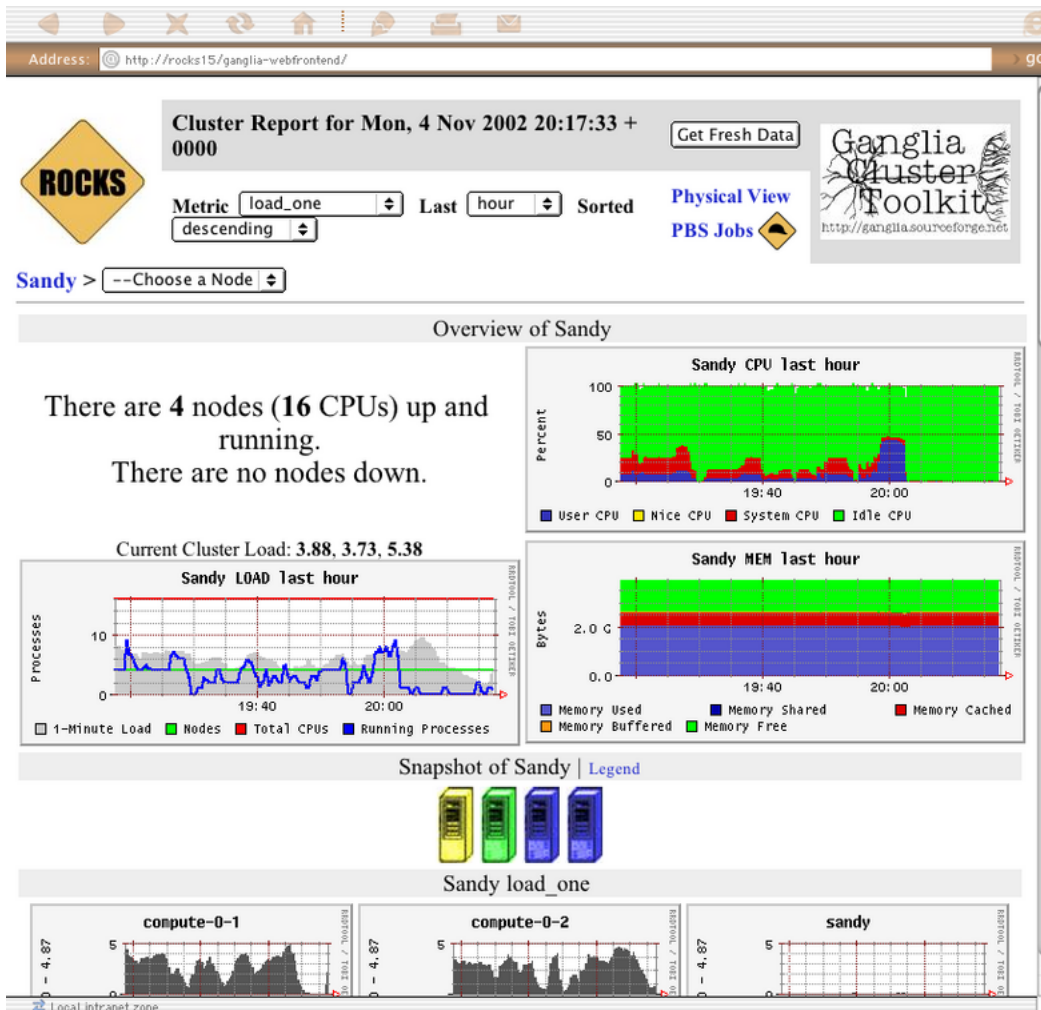
The Rocks Cluster Group maintains a similar web page called *Meta* that collects ganglia information from many clusters built with Rocks software. It may give you a glimpse of the power and scalability of the Ganglia monitors. The meta page is available at http://meta.rocksclusters.org/[5].

Ganglia[6] was designed at Berkeley by Matt Massie (massie@cs.berkeley.edu) in 2000, and is currently developed by an open source partnership between Berkeley, SDSC, and others. It is distributed through Sourceforge.net under the GPL software liscence.

3.4. Cluster Top

This page is a version of the standard "top" command for your cluster. This page presents process information from each node in the cluster. This page is useful for monitoring the precise activity of your nodes.

The Cluster Top differs from standard top in several respects. Most importantly, each row has a "HOST" designation and a "TN" attribute that specifies its age. Since taking a process measurement itself requires resources, compute nodes report process data only once every 60 seconds on average. A process row with TN=30 means the host reported information about that process 30 seconds ago.

For brevity and minimal performance impact, each node only reports as many processes as it has CPUs. The processes shown had the highest %CPU utilization on the node at the time of reporting. Unfortunately the number of processes per node is not currently adjustable. The restriction lies in the structure of the Ganglia monitoring system, which only delivers information and has no faculty for accepting parameters on the fly. However, showing the most CPU intensive processes should give you a good idea of how the CPUs are being utilized.

The process data is gathered by raw processing of the /proc filesystem on each node. Memory statistics differ slightly from standard "ps" output, and are calculated from the /proc/[pid]/statm virtual file.

Process Columns

TN

The age of the information in this row, in seconds.

HOST

The node in the cluster on which this process is running.

PID

The Process ID. A non-negative integer, unique among all processes on this node.

USER

The username of this processes.

CMD

The command name of this process, without arguments.

%CPU

The percentage of available CPU cycles occupied by this process. This is always an approximate figure, which is more accurate for longer running processes.

%MEM

> The percentage of available physical memory occupied by this process.

SIZE

> The size of the "text" memory segment of this process, in kilobytes. This approximately relates the size of the executable itself (depending on the BSS segment).

DATA

> Approximately the size of all dynamically allocated memory of this process, in kilobytes. Includes the Heap and Stack of the process. Defined as the "resident" - "shared" size, where resident is the total amount of physical memory used, and shared is defined below. Includes the the text segment as well if this process has no children.

SHARED

> The size of the shared memory belonging to this process, in kilobytes. Defined as any page of this process' physical memory that is referenced by another process. Includes shared libraries such as the standard libc and loader.

VM

> The total virtual memory size used by this process, in kilobytes.

Onyx Cluster Top

Thu, 4 Sep 2003 17:57:51 +0000   Physical Job Assignments

Show only processes by user: [   ] (Go)

TN	HOST	PID	USER	CMD	%CPU	%MEM	SIZE	DATA	SHARED	VM	Up | Down
2	onyx.local	1606	root	sge_commd	99.90	0.36	100	3192	568	3760	
2	onyx.local	8	root	kscand	11.11	0.00	0	0	0	0	
52	compute-0-2.local	8	root	kscand	3.70	0.00	0	0	0	0	
2	onyx.local	1104	root	gschedule	2.47	44.91	680	460876	2012	463112	
93	compute-0-1.local	2162	root	gschedule	1.24	28.27	680	289308	2044	291352	
16	onyx.local	1277	nobody	gmond	1.23	0.15	92	828	684	1512	
2	onyx.local	1	root	init	0.00	0.04	24	24	416	480	
35	onyx.local	2	root	keventd	0.00	0.00	0	0	0	0	

3.5. Other Cluster Monitoring Facilities

3.5.1. The Proc Filesystem

The next link leads to a standard Apache filesystem view of the Linux `/proc` filesystem. These files and directories do not reside on disk, but are instead dynamically generated by the Linux kernel upon request. They are used to convey dynamic information about resource usage and running processes on the machine. Due to their ethereal nature, the information provided by the /proc files is extremely fresh, and in fact represent the current state of the operating system at the time the file was requested.

However, data contained in these files may reveal information useful to hackers and other malicious parties. In addition to user names and program parameters, this area contains data about local network interfaces and firewalls. Therefore, by default this link is subject to the same "private network only" restriction as the database web interface.

3.5.2. Cluster Distribution

This link displays a filesystem view of the `/home/install/` directory tree on the frontend node. This area holds the repositories of RPM packages used to construct nodes in the cluster, along with the XML kickstart graph that defines the various node types. The distribution used to build the cluster may be examined here.

Knowledge of the software versions present on the cluster is considered sensitive since it may give hackers insight to available security holes. By default, access to this link is restricted to the private network as well.

3.5.3. Kickstart Graph

This link will return an image of the current kickstart graph used to choose software for appliance types. Rocks automatically generates kickstart files based on the nodes and edges in this graph. Rolls can add or alter the graph, and new appliance types may be created. Currently Rocks differentiates appliances only by their starting node in this graph however more complex definitions are possible.

The GIF image returned by this link is generated on the fly, and so is current. It is made by the "dot" application that is part of the GraphVIZ project.

3.5.4. Cluster Labels

The "Make Labels" link generates a PDF document containing labels for each node in the cluster. By saving this document to disk and printing it on standard Avery 5260 address stock, you can easily label the nodes of your cluster.

3.5.5. Rocks Users Guide

The final link on the Table of Contents page leads to the Rocks Users Guide. This is simply a local version of the guide present on the official Rocks website http://www.rocksclusters.org/[7]. In it you will find instructions for addding nodes to your cluster, as well as FAQs which may prove invaluable during troubleshooting. You may have already read much of the guide, as this page resides in it.

These links represent the monitoring tools available on a standard Rocks cluster. With them you may edit the database, view the state of the cluster's resources and the parallel job queue, and examine the software package repository. These tools are actively being developed extended, and additional pages may be added in the future.

3.6. Monitoring Multiple Clusters with Ganglia

Ganglia has the ability to track and present monitoring data from multiple clusters. A collection of monitored clusters is called a *Grid* in Ganglia's nomenclature. This section describes the steps required to setup a multi-cluster monitoring grid.

The essential idea is to instruct the gmetad daemon on one of your frontend nodes to track the second cluster in addition to its own. This procedure can be repeated to monitor a large set clusters from one location.

For this discussion, your two clusters are named "A" and "B". We will choose the frontend on cluster "A" to be the top-level monitor.

1. On "A" frontend, add the line to /etc/gmetad.conf:

   ```
   data_source "Cluster B" B.frontend.domain.name
   ```

   Then restart the gmetad server on "A" frontend.

2. On "B" frontend, add the line to /etc/gmond.conf:

   ```
   trusted_hosts A.frontend.domain.name
   ```

   Then restart the gmetad server on "B" frontend.

3. Take a look at the Ganglia page on "A". It should include statistics for B, and a summary or "roll-up" view of both clusters.

This screenshot is from the iVDGL Physics Grid3 project. It is a very large grid monitored by Ganglia in a similar manner as specified here.

Notes

1. http://www.rocksclusters.org/rocks-documentation/reference-guide/3.2.0/database.html

2. http://httpd.apache.org/docs/howto/auth.html#allowdeny

3. http://www.phpmyadmin.net/

4. http://ganglia.sourceforge.net/

5. http://meta.rocksclusters.org/

6. http://ganglia.sourceforge.net/

7. http://www.rocksclusters.org/
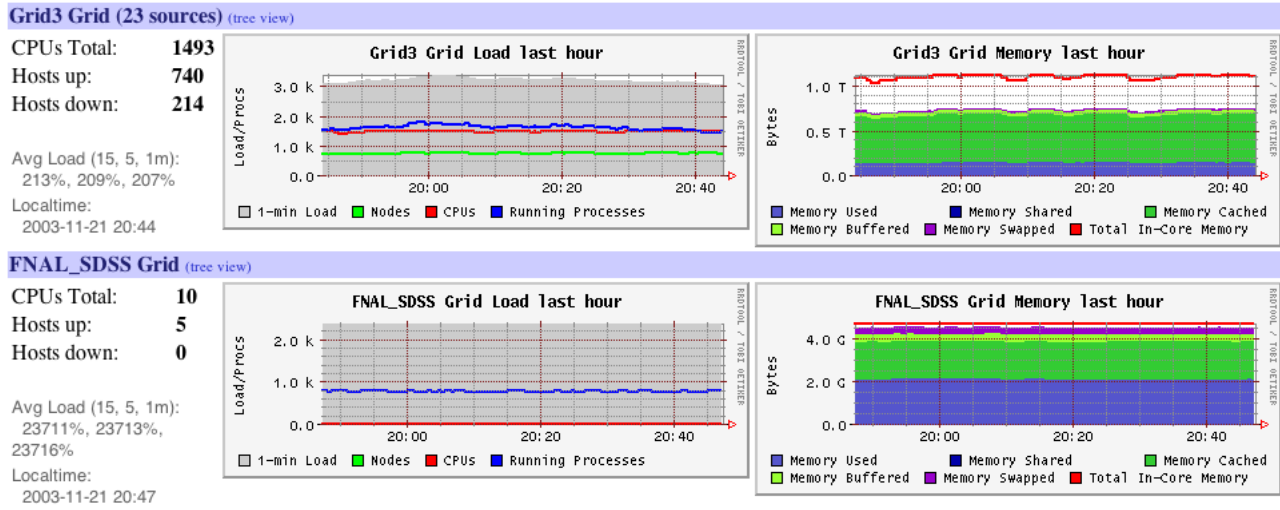
Chapter 4. Cluster Services

4.1. Cluster Services

This chapter presents other miscellaneous services present on Rocks clusters.

4.2. 411 Secure Information Service

The 411 Secure Information Service provides NIS-like functionality for Rocks clusters. It is named after the common "411" code for information in the phone system. We use 411 to securely distribute password files, user and group configuration files and the like.

411 uses Public Key Cryptography to protect files' contents. It operates on a file level, rather than the RPC-based per-line maps of NIS. 411 does not rely on RPC, and instead distributes the files themselves using HTTP (web service). Its central task is to securely maintain critical login/password files on the worker nodes of a cluster. It does this by implementing a file-based distributed database with weak consistency semantics. The design goals of 411 include scalablility, security, low-latency when changes occur, and resiliance to failures.

> Beginning with the Rocks 3.1.0 Matterhorn release, 411 replaces NIS as the default method of distributing /etc/passwd and other login files. We no longer support NIS.

4.2.1. Using the 411 Service

The 411 system intentionally mimics the NIS interface for system administrators. Of course there are elements in 411 which are not present in NIS, namely RSA public and private cryptographic keys. However we have attempted to make 411 as easy to use in the NIS capacity as possible.

Files listed in `/var/411/Files.mk` are automatically serviced by 411. This means that any file listed there will be kept up to date by the 411 agents on all compute nodes in your cluster. This is done using the makefile `/var/411/Makefile` in a similar fashion to NIS. To force the 411 system to flush all changes, execute the following on the frontend node:

```
# make -C /var/411
```

Note that this command is run by cron every hour on the frontend to propagate password changes, etc to compute nodes. New files can be added to Files.mk as necessary for custom services on the cluster.

We provide a 411 initscript to do some useful tasks. The command "`/etc/init.d/411 commit`" will run the make command in the /var/411 directory as shown above. Running "`/etc/init.d/411 restart`" will force all 411 files to be re-encrypted and change alerts resent for each. This can be useful for troubleshooting purposes. Note that this command is not necessary when restarting or adding nodes to your cluster, as they will pull the latest 411 files from the frontend automatically upon startup.

The 411 service requires multicast support on your cluster's private network for optimum performance. If you suspect the 411 service of not working correctly, please verify that multicast messages sent by the frontend are visible on all compute nodes in your cluster. Using the "tcpdump" command can be useful here.

4.2.2. Structure

4.2.2.1. Overview

The 411 service defines Master and Slave nodes like standard NIS. Master nodes encrypt and serve 411 files (called 411 messages once they are encrypted) using their local Apache web server. Slave nodes, generally the compute nodes in the cluster, retrieve 411 messages using HTTP, decrypt them, and save the resultant file to their local filesystem.

Slave nodes may recognize multiple master servers, and make some attempt to load balance their 411 message retrievals accross the set of master servers to reduce strain on the cluster. Slave nodes have two components: the poller and the listener, which are described in detail below. Master nodes have only the `411put` command, which publishes a new or changed 411 file. It is possible for a node to be both a master and slave node at the same time.

411 Secure Information Service

Frontend

411 files

Multicast (Ganglia): A 411 file changed!

HTTP: Master, give me all your 411 files.

Node

Poller

Listener

Node

Poller

Listener

Node

Poller

Listener

Shared RSA Public Key

Shared RSA Private Key
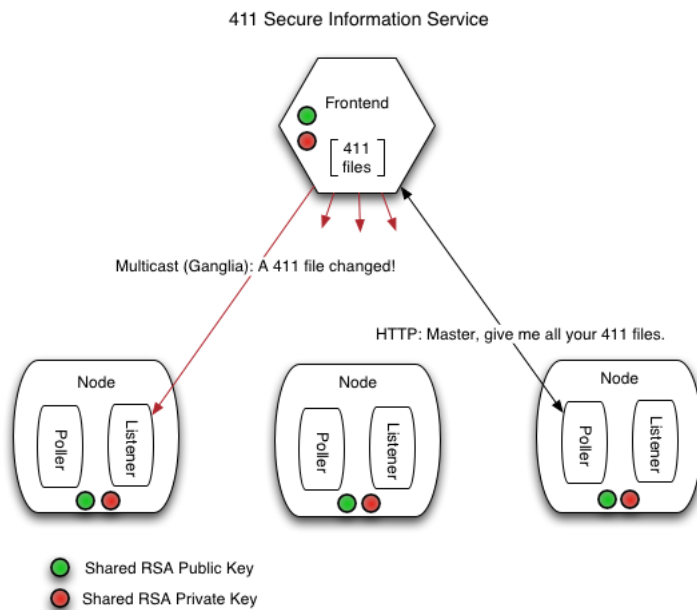
4.2.2.2. Security

Security is achived by using 1024-bit RSA public and private keys. The masters and all slaves in a cluster share the same public and private key. Although this scheme is less secure than using a unique key per host, it reduces the processing requirements of the system dramatically.

When a master server wishes to publish a file using 411, it uses the "411put" command to encrypt the file (or directory) with the cluster public key. This encryption is done only once, no matter how many nodes are present in the cluster. The encrypted file is stored on a plain HTTP server, with no HTTPS requirements. A slave node must posses the cluster private key to decrypt and interpret the 411 message.

411 messages contain encrypted headers that specify the location, owner, and permissions of the file on the master server. In this way a slave securely knows where and how to place the file on its local filesystem. In a correctly functioning 411 system, the slaves look identical to their masters in terms of their 411 files.

We found that widely available HTTPS implementations take an order of magnitude more processing time to serve a file than regular HTTP. Large clusters using 411 would quickly spend all their time maintaining secure HTTPS connections.

411 uses a Hybrid encryption scheme for performance reasons, similar to PGP and GnuPG. A 256-bit random number is chosen as a session key and encrypted with the cluster public key. The session key is used to encrypt

the file's contents and headers with the fast Blowfish symmetric cypher. A new session key is chosen for every file, and the encrypted message is transformed into ascii text using Base64 encoding.

4.2.2.3. Poller

Each slave node polls its master server on average of once per day. During these polls, the slave retrieves all available 411 messages by getting a directory listing of the master's 411 web directory and performs a `411get` on each message it finds. The purpose of the 411 poller is to allow nominally correct operation of the service in case the 411 listener has failed.

The poller is implemented as a `gschedule` event module, and relies on the operation of that daemon. 411 Pollers obtain their master servers by reading a configuration file on their local disk. This file, written in XML, is generated automatically by the 411 listener.

Since the 411 polling daemon operates as the root user on client nodes, it is essential that the 411 http directory on a master server be writable only by root. If any user is allowed to write messages to the master's 411 directory, security can easily be compromized by a user-level file being written as root by a client node.

The 411 message format is simple and easy to mimic, and a clever user with write access to the 411 web-visible directory (currently `/etc/411.d`) could put a file anywhere on the slave node's filesystem. Insure this directory is writable only by root. This state is enforced by standard Rocks, but we want to reiterate its importance.

4.2.2.4. Listener

Slave nodes listen on the Ganglia multicast channel for "411alert" messages from the master. The master will send 411alerts during a 411put operation, just after it has encrypted a 411 file. The alert message serves as a cue to the slave nodes that a file has changed and needs to be retrieved. In this way the 411 system generally achieves a low-latency response to changes.

Using lightweight UDP messages sent over a multicast channel has advantages and disadvantages. The advantages are speed and widespread reception of the alert - the master does not need to know the names of its slaves. The disadvantages are speed and unreliable message delivery. A master may get overwhelmed with 411 get requests, and some slaves may not receive the alert at all.

To prevent flooding the master server with requests, the listeners keep an estimate of the cluster size, and use that estimate to set a random backoff for their requests. A slave does not immediately request the changed file, but waits

some amount of time before asking for it. Currently, the backoff is calibrated to allow only 8 concurrent connections to the master server on average.

The problem of listeners not receiving an alert, perhaps because the message was dropped in the network, is solved by relying on the 411 poller as a backup. Any new or changed files will eventually make it to all correctly functioning slave nodes, regardless of the multicast message reliability.

In Rocks 3.2.0 the 411 service has been strengthened. Instead of sending a single multicast alert when a 411 file has changed, the frontend repeats the alert every 10 seconds for about an hour. Clients will ignore duplicate alerts, so the these repeats incur minimal overhead. The benefit is when viewed over the hour, we have achieved reliable multicast: every client is guarenteed to receive an alert with very high probability. Ganglia uses this strategy to insure reliable UDP multicast as well.

411 is akin to a distributed database, and is not a centralized lookup service like NIS. Unlike NIS, however, it has been designed to scale to hundreds of nodes, and many reports have shown this to be the case. It also uses full RSA-grade cryptographic security. However, all systems have limitations, and 411 can never ensure correct operation over an arbitrarily small time window. As the window (or delta) gets smaller, the chance grows that some node does not receive a 411 file when it should. There are fundamental reasons for this deficiency: any single alert can be lost, and clients do not poll constantly. However over a reasonable time window, 411 works correctly. Tests have shown that 124 nodes converge on a new 411 file in approximately 20s.

4.2.2.5. Denial-of-Service Security and Automatic Configuration

One might observe that a malicious attacker could send bogus 411alert messages to bring down the master's HTTP server. The problem is the alert multicast packet is easy to make, and will cause a large response in the system. The 411 service addresses this problem by cryptographically signing each alert. The master server uses the secret cluster private key to sign the alert message, which the slave nodes verify with the public key. In addition, the master includes a timestamp with each alert so old (valid) alerts cannot be stored by the attacker and reused.

Any 411 alert that does not verify or that carries a timestamp that we have already seen is ignored. In this way as long as the cluster private key is kept secure, only legitimate alerts will carry weight.

The final observation is that any true 411alert must come from a valid master server. We use this fact to automatically learn about new masters. The text of a 411 alert contains the network address of the master that sent it, which cannot be altered without breaking the cryptographic signature. The listeners use this address to create the 411 configuration file, which is used by the poller.

Upon verifying an alert from a master server it has not seen before, a 411 listener writes a new configuration file for the service. Pollers monitor the modification time of this file, and reload its contents on a change. In this way new

master servers are easily and securely added to the system. In the face of unreliable and lost alerts the file must be created by hand, but we do not expect this to be the common case.

In addition, each client node keeps a "quality score" counter for each master server in its configuration file. When a client sucessfully contacts a 411 master, the client increments the saturating score counter for that master. Similarly, if the client fails to contact a particular 411 master server, the client decrements the appropriate score. When a client needs to perform a 411 operation, it chooses the master server with the highest quality rating. In this way dead or overloaded master servers are used less frequently by clients, promoting good health and load balance in the cluster.

4.2.3. Commands

4.2.3.1. 411put

```
411put [--411dir=dir] [--urldir=dir] [--see] [--noalert] [--alert=channel]
[--411name] [--pub] [--priv] [--comment=char] file1 file2 ...
```

Encrypts and publishes files using the 411 secure information service. Will send a multicast message to client nodes by default, alerting them of a changed file.

The following options are available:

- *--comment* The comment character for this file. Used to place a descriptive header without disrupting normal operations. Defaults to "#".

- *--411dir* The local directory to place encrypted 411 messages. Defaults to "/etc/411.d/". Be careful about the permissions of this directory.

- *--urldir* The web directory where 411 messages are available. Defaults to "/411.d/".

- *--see* Shows the encrypted file contents on stdout.

- *--noalert* Suppresses alert message.

- *--alert* Specifies the alert channel, which can be multicast or unicast UDP. Defaults to the ganglia channel (239.2.11.71).

- *--411name* Prints the 411 message name for the file. Provided for convenience.

- *--pub* The location of the cluster public RSA key. Defaults to a 1024 bit key in "/etc/security/cluster-public-key.pem". This file should have permissions 0444 (read by all) and be owned by root.

- *--priv* The location of the cluster private RSA key. Defaults to a 1024 bit key in "/etc/security/cluster-private-key.pem". This file should be owned by root and have permissions 0400 (read only by root).

4.2.3.2. 411get

```
411get [--all] [--master=url] [--conf] [--pub] [--priv] [file]
```

Retrieves and decrypts 411 messages. Prints resulting file to standard out. When invoked with no files, 411get will list the available 411 messages.

The following options are available:

- *--master* The url of a 411 master server to use. Defaults to "http://10.1.1.1/411.d/" or whatever is present in "/etc/411.conf". If given, this master takes precedence over those listed in the configuration file.

- *--conf* The configuration file to use. Defaults to "/etc/411.conf".

- *--all* Retrieves and writes all available 411 messages from the most attractive master. Does not print output to stdout, nor ask for confirmation before overwriting files.

- *--pub* The location of the cluster public RSA key. Defaults to "/etc/security/cluster-public-key.pem".

- *--priv* The location of the cluster private RSA key. Defaults to "/etc/security/cluster-private-key.pem".

To cause a compute node to pull all 411 files from its favorite master server, run the following command as root on the compute node:

```
# 411get --all
```

The master servers, along with their quality score, are listed in the "/etc/411.conf" file on compute nodes.

4.2.4. Conclusion

The 411 service is intended to provide a secure and scalable alternative to NIS. In the future we plan to allow muliple levels of master servers so files can be defined from many locations, enabling on-the-fly addition of users, etc. with an eye on Grid Services. We hope this new service is useful to you, and look forward to your comments.

4.3. Domain Name Service (DNS)

Starting with the Lhotse release, Rocks clusters contain a fully-operational DNS server on the frontend. This name server coordinates the name->ip address mapping for each node in the cluster. In previous versions of Rocks, hostnames were resolved using a NIS map of the /etc/hosts file.

The switch to a full-fledged name service requires more discipline with our naming practices. We choose a domain name for the internal cluster, ".local" by default, and strictly separate internal names from external ones.

One problem with standard UNIX naming (and Linux in particular), is that a machine has only one name. This becomes an issue for machines like the frontend, which have two network interfaces: one on the internal private network, and one on the public network.

While external services such as Globus requires the frontend to be named by its public address, internal systems such as the queuing system (PBS, etc) prefers the frontend to carry the internal local name.

In Rocks, we have made the decision that all ".local" names resolve to an interface on the private cluster network. This includes all nodes and the `eth0` interface of the frontend, and generally these names map to IP addresses in the 10.x.x.x range.

For Globus compatibility, the frontend node is named with its public name. This means a "hostname" command will return its public name, rather than one ending with ".local". Some internal systems are made more complicated by this choice, but those that correctly use the standard resolver library (in libc) have no problems.

New nodes added with "insert-ethers" will automatically be added to the local DNS domain. To see a complete list of node names, execute the following commands.

```
$ host -l local
```

4.3.1. Extending DNS

Rocks provides a mechanism to put external hostnames under the DNS control of your cluster. Generally, external hosts have names served by site-wide DNS servers. However if there is no external DNS server available, you may want to use your frontend's DNS server to handle the name->IP mappings for certain non-cluster nodes.

Since the DNS configuration file is automatically generated by a dbreport, you cannot add static configuration to the standard zone files in /var/named. Instead, put local name mappings in the file:

```
/var/named/rocks.domain.local
```

And reverse mappings (IP->name) in:

```
/var/named/reverse.rocks.domain.local
```

These files are automatically included by the Rocks dns dbreport, which can be refreshed with the command:

```
# insert-ethers --update
```

These files are in the BIND configuration format, just like the standard `rocks.domain` and `reverse.rocks.domain` files that are generated by the Rocks dbreport.

Your external hosts will have names in the .local cluster domain.

Errors in your local DNS files will cause the entire local cluster domain naming to fail. Proceed with caution.

4.4. Mail

Starting with the Shasta release, Rocks has moved to the *Postfix* mail server for cluster-wide email service. RedHat, Apple, and most other Unix-based OSs have switched from Sendmail to Postfix in recent years. Postfix is much easier to configure than its predecessor, a welcome fact to most administrators. Postfix also has more advanced security features and better performance than Sendmail.

In Rocks, the frontend serves as the mail relay for all cluster nodes. This means that compute nodes send mail to the frontend, which forwards it to the outside world.

You can view the mail log on the frontend in the file `/var/log/mail`. Here postfix keeps a record of all incoming and outgoing email messages.

Chapter 5. Customizing your Rocks Installation

5.1. Adding Packages to Compute Nodes

Put the package you want to add in:

```
/home/install/contrib/enterprise/3/public/arch/RPMS
```

Where *arch* is your architecture ("i386", "ia64", etc).

Create a new XML configuration file that will *extend* the current `compute.xml` configuration file:

```
# cd /home/install/site-profiles/3.2.0/nodes/
# cp skeleton.xml extend-compute.xml
```

Inside `extend-compute.xml`, add the package name by changing the section from:

```
<package> <!-- insert your package name here --> </package>
```

to:

```
<package> your package   </package>
```

It is important that you enter the *base name* of the package in `extend-compute.xml` and not the full name.

For example, if the package you are adding is named *XFree86-100dpi-fonts-4.2.0-6.47.i386.rpm*, input *XFree86-100dpi-fonts* as the package name in `extend-compute.xml`.

```
<package>XFree86-100dpi-fonts</package>
```

If you have multiple packages you'd like to add, you'll need a separate `<package>` tag for each. For example, to add both the 100 and 75 dpi fonts, the the following lines should be in `extend-compute.xml`:

```
<package>XFree86-100dpi-fonts</package>
<package>XFree86-75dpi-fonts</package>
```

Now build a new Rocks distribution. This will bind the new package into a RedHat compatible distribution in the directory */home/install/rocks-dist/...*.

```
# cd /home/install
# rocks-dist dist
```

Now, reinstall your compute nodes.

5.2. Customizing Configuration of Compute Nodes

Create a new XML configuration file that will *extend* the current `compute.xml` configuration file:

```
# cd /home/install/site-profiles/3.2.0/nodes/
# cp skeleton.xml extend-compute.xml
```

Inside `extend-compute.xml`, add your configuration scripts that will be run in the *post configuration* step of the Red Hat installer.

Put your bash scripts in between the tags `<post>` and `</post>`:

```
<post>
 < !-- insert your scripts here -->
</post>
```

To apply your customized configuration scripts to compute nodes, rebuild the distribution:

```
# cd /home/install
# rocks-dist dist
```

Then, reinstall your compute nodes.

5.3. Configuring Additional Ethernet Interfaces

For compute nodes, Rocks uses the first ethernet interface (`eth0`) for management (e.g., reinstallation), monitoring (e.g., Ganglia) and message passing (e.g., MPICH over ethernet). Often, compute nodes have more than one ethernet interface. This procedure describes how to configure them.

Additional ethernet interfaces are configured from the frontend via a command line utility named `add-extra-nic`. It manipulates the networks table on the frontend to add information about an extra interface on a node (a description of the networks table can be found in Rocks Cluster Schema[1]) This program just manipulates the database. It does nothing to the actual configuration of a running node.

Once you have the information in the networks table, every time you reinstall, the additional NIC will be configured. The structure supports multiple additional interfaces/node.

- For each node that has an additional ethernet interface, execute:

  ```
  # add-extra-nic --if=<interface> --ip=<ip address> --netmask=<netmask> --gateway=<gateway> -
  -name=<host name> <compute node>
  ```

  Where:

  interface

  > The name of the ethernet interface (e.g., eth1).

  ip address

  > The internet address for the interface (e.g., 192.168.1.1).

  netmask

  > The network mask for the interface (e.g., 255.255.255.0).

  gateway

  > The gateway for this interface (e.g., 192.168.1.254).

  host name

  > Host name for the interface (e.g., fast-0-0).

  compute node

  > The name of the compute node to apply the configuration to (e.g., compute-0-0).

  For example, say you want to configure interface eth1 for compute node compute-0-0 with the IP address 192.168.1.1 with a netmask of 255.255.255.0 with a gateway of 192.168.1.254 and you want to name the new interface fast-0-0. The call to add-extra-nic would look like:

  ```
  # add-extra-nic --if=eth1 --ip=192.168.1.1 --netmask=255.255.255.0 --gateway=192.168.1.254 -
  -name=fast-0-0 compute-0-0
  ```

- Reinstall the nodes that you have defined an additional interface (use shoot-node).

5.4. Compute Node Disk Partitioning

5.4.1. Default Disk Partitioning

The default root partition is setup to be 6 GB. The default swap partition is 1 GB. The remainder of the root disk is setup as the partition `/state/partition1`.

Table 5-1. Compute Node -- Default Root Disk Partition

Partition Name	Size
/	6 GB
swap	1 GB
/state/partition1	*remainder of root disk*

All remaining disk drives will have one partition with the name `/state/partition2, /state/partition3, ...`

For example, the following table describes the default partitioning for a compute node with 3 SCSI drives.

Table 5-2. A Compute Node with 3 SCSI Drives

Device Name	Mountpoint	Size
/dev/sda1	/	6 GB
/dev/sda2	swap	1 GB
/dev/sda3	/state/partition1	*remainder of root disk*
/dev/sdb1	/state/partition2	*size of disk*
/dev/sdc1	/state/partition3	*size of disk*

After the initial installation, all data in the file systems labeled `/state/partitionX` will be preserved over reinstallations.

5.4.2. How to Change the Size of Root and Swap Partitions on Compute Nodes

This section describes a simple method in which to change the size of the default root and swap partitions on compute nodes. If more control over the compute partitioning is desired, see the section Customizing Compute Node Disk Partitions.

First, create the file `extend-auto-partition.xml`.

```
# cd /home/install/site-profiles/3.2.0/nodes/
# cp skeleton.xml extend-auto-partition.xml
```

Above the `<main>` section, insert the following two lines:

```
<var name="Kickstart_PartsizeRoot" val="10000"/>
<var name="Kickstart_PartsizeSwap" val="2000"/>
```

This will increase the root partition from the default 6 GB to 10 GB and it will increase the swap partition from the default 1 GB to 2 GB.

Then apply this configuration to the distribution by executing:

```
# cd /home/install
# rocks-dist dist
```

To reformat all the compute nodes to your specification above, execute:

```
# ssh-add
# cluster-fork 'rm /.rocks-release'
# cluster-fork '/boot/kickstart/cluster-kickstart'
```

5.4.3. Customizing Compute Node Disk Partitions

Create a new XML configuration file that will *replace* the current `auto-partition.xml` configuration file:

```
# cd /home/install/site-profiles/3.2.0/nodes/
# cp skeleton.xml replace-auto-partition.xml
```

Inside `replace-auto-partition.xml`, add the following section:

```
<main>
 <part> / --size 4096 --ondisk hda </part>
 <part> swap --size 1000 --ondisk hda </part>
 <part> /mydata --size 1 --grow --ondisk hda </part>
```

```
</main>
```

This will set up a 4 GB root partition, a 1 GB swap partition, and the remainder of the drive will be set up as `/mydata`. Additional drives on your compute nodes can be setup in a similar manner by changing the `--ondisk` parameter.

In the above example (aside from the `<part>` and `</part>` tags), the remaining syntax follows directly from Red Hat's kickstart. For more information on the `part` and the `clearpart` keywords, see Red Hat Enterprise Linux 3: System Administration Guide[2]

User-specified partition mountpoint names (e.g., `/mydata`) cannot be longer than 15 characters.

If you would like to use software RAID on your compute nodes, inside `replace-auto-partition.xml` add section that looks like:

```
<main>
 <part> / --size 8000 --ondisk hda </part>
 <part> swap --size 1000 --ondisk hda </part>

 <part> raid.00 --size=10000 --ondisk hda </part>
 <part> raid.01 --size=10000 --ondisk hdb </part>

 <raid> /mydata --level=1 --device=md0 raid.00 raid.01 </raid>
</main>
```

If the user-specified partitioning scheme *is not currently configured* on an installing compute node, then all the partitions on the compute node will be removed and the user-specified partitioning scheme will be forced onto the node.

If the user-specified partitioning scheme *is currently configured* on an installing compute node, then all the partitions on the node will remain intact and only the root partition will be reformatted.

If you want to change the size of an existing partition, you must rename the mountpoint for the partition. This is because the matching logic we wrote only keys off mountpoint names.

If you change the partitioning scheme, *all* partitions will be removed and reformatted. This is because we have been unable to make Red Hat's `clearpart --drives=` work as advertised.

Then apply this configuration to the distribution by executing:

```
# cd /home/install
# rocks-dist dist
```

To reformat all the compute nodes to your specification above, execute:

```
# ssh-add
# cluster-fork 'rm /.rocks-release'
# cluster-fork '/boot/kickstart/cluster-kickstart'
```

5.5. Creating a Custom Kernel RPM

5.5.1. Creating a Custom Kernel RPM using RedHat's Kernel Source

This procedure involves bringing up a compute node with Rocks *first*, then logging on to the compute node to make the custom kernel.

- Login to a compute node.

  For example:

  ```
  # ssh compute-0-0
  ```

- Go to the directory where the Linux kernel source code resides:

  ```
  # cd /usr/src/linux-2.4
  ```

- Build your `.config` file.

  We recommend that you copy the appropriate *config file* from the `configs` directory to `.config`, then edit it to suit your needs. For example, if you want to configure a kernel for a SMP-based `i686`:

  ```
  # cp configs/kernel-2.4.21-i686-smp.config .config
  ```

  To determine the processor architecture of the node, execute:

  ```
  # uname --processor
  ```

- Build a kernel RPM based on your modified `.config`:

```
# make rpm
```

• Copy the resulting RPM back to the Rocks distribution on the frontend.

The final lines of the `make rpm` command indicates the name of the resulting kernel RPM.

For example:

```
# scp /usr/src/redhat/RPMS/i686/kernel-smp-2.4.20-19.7.i686.rpm \
 frontend-0:/home/install/contrib/enterprise/3/public/i386/RPMS/
```

• Rebuild the distribution on the frontend:

```
# ssh frontend-0
# cd /home/install
# rocks-dist dist
```

Your new kernel is now applied to the Rocks distribution.

• Test the new kernel by reinstalling a compute node:

```
# shoot-node compute-0-0
```

• If the kernel works to your satisfaction, reinstall all the compute nodes that you want to run the new kernel.

5.5.2. Creating a Custom Kernel RPM using kernel.org's Source

This procedure involves bringing up a compute node with Rocks *first*, then logging on to the compute node to make the custom kernel.

• Login to a compute node.

For example:

```
# ssh compute-0-0
```

• Download the kernel source tarball from kernel.org. For example:

```
# cd /usr/src
# wget http://www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.26.tar.bz2
```

- Prepare the source tree.

```
# bunzip2 linux-2.4.26.tar.bz2
# tar vfx linux-2.4.26.tar
# rm linux-2.4
# ln -s linux-2.4.26 linux-2.4
```

- Apply the Rocks RPM that allows for kernel RPM building (it is found on the frontend under `/home/install/rocks-dist/...`).

```
# rpm -Uvh --force \
http://frontend-0/install/rocks-dist/enterprise/3/en/os/*/RedHat/RPMS/rocks-kernel-*
```

- Create a `.config` file and put it in `/usr/src/linux-2.4`.

  It is required that the following configuration variables are present in the `.config` file:

  ```
  CONFIG_BLK_DEV_RAM=y
  CONFIG_BLK_DEV_RAM_SIZE=8192
  CONFIG_BLK_DEV_INITRD=y
  ```

- Build a kernel RPM:

```
# cd /usr/src/linux-2.4
# make rpm
```

- Copy the resulting RPM back to the Rocks distribution on the frontend.

  The final lines of the `make rpm` command indicates the name of the resulting kernel RPM.

  For example:

```
# scp /usr/src/redhat/RPMS/i686/kernel-smp-2.4.26-1.i686.rpm \
 frontend-0:/home/install/contrib/enterprise/3/public/i386/RPMS/
```

- Rebuild the distribution on the frontend:

```
# ssh frontend-0
# cd /home/install
# rocks-dist dist
```

Your new kernel is now applied to the Rocks distribution.

- Test the new kernel by reinstalling a compute node:

  ```
  # shoot-node compute-0-0
  ```

- If the kernel works to your satisfaction, reinstall all the compute nodes that you want to run the new kernel.

5.6. Mirroring the Rocks and Red Hat Distributions

5.6.1. Mirroring the Rocks Cluster Distribution

After the frontend is installed, only a portition of the entire Rocks cluster distribution is installed on the frontend (in fact, only the portion that could fit on the first CD is present).

To obtain the entire Rocks cluster distribution, execute:

```
# cd /home/install
# rocks-dist mirror
```

This puts the entire Rocks distribution under the directory `/home/install/ftp.rocksclusters.org`

To recompile the distribution, that is, to incorporate all the packages from the updated mirror, execute:

```
# cd /home/install
# rocks-dist dist
```

This binds all the packages from `ftp.rocksclusters.org` into a new Red Hat compliant distribution under `/home/install/rocks-dist`

5.6.2. Incorporating Multiple Red Hat Mirrors

The following is a procedure on how to mirror an official Red Hat mirror (e.g., in order to incorporate the latest Red Hat packages as they are made available).

First, select a mirror host[3] (we mirror from `andromeda.acs.uci.edu`).

Then find the *root* of the mirror. This the directory that has the directory names `7.2`, `7.3`, `8.0`, `etc.`. Again, using our example, on `andromeda.acs.uci.edu`. this is `mirrors/redhat/linux`.

To pass the above info to `rocks-dist`, create the file `/home/install/rocks-distrc` and place the following in it:

```
<?xml version="1.0" standalone="yes"?>

<rocks-dist>
        <host name="updates" value="andromeda.acs.uci.edu"/>
        <path name="updates" value="mirrors/redhat/linux"/>
</rocks-dist>
```

Now you can download RPMs from the specified mirror site by executing:

```
# cd /home/install
# rocks-dist mirror
```

Now build a new distribution that merges the Rocks mirror (`ftp.rockscluster.org`) and your selected Red Hat mirror:

```
# rocks-dist dist
```

This compiles a new distribution under the directory `/home/install/rocks-dist`.

Now reinstall your compute nodes.

5.7. Making Your Own Cluster Distribution Media

You've modified the base Rocks cluster distribution and now you want to create a media set (CDs or DVD).

First, login to your frontend and obtain the entire Rocks cluster distribution:

```
# cd /home/install
# rocks-dist mirror
```

This puts the entire Rocks distribution under the directory `/home/install/ftp.rocksclusters.org/...`

Now, recompile the distribution in pristine form (without Rolls).

```
# cd /home/install
# rm -rf rocks-dist
# rocks-dist --pristine dist
```

This binds all the packages from `ftp.rocksclusters.org`, all the packages under `/usr/src/redhat/RPMS/...` and all the packages under `/home/install/contrib/...` into a new Red Hat compliant distribution under `/home/install/rocks-dist`

After rebuilding a package, before you try to make a media set, you must re-execute:

```
# cd /home/install
# rocks-dist --pristine dist
```

Now build the media set (in this case, build the CD set):

```
# cd /home/install
# rm -rf cdrom
# rocks-dist --dist=cdrom cdrom
```

This puts the CD set under `/home/install/cdrom/...`.

The first ISO is named `/home/install/cdrom/enterprise/3/en/os/rocks-disk1.iso`, the second is `/home/install/cdrom/enterprise/3/en/os/rocks-disk2.iso`, the third is `/home/install/cdrom/enterprise/3/en/os/rocks-disk3.iso`, etc.

You're done -- you now have a customized cluster distribution ready to burn onto blank CD media.

To build a DVD set, execute:

```
# cd /home/install
# rm -rf dvd
# rocks-dist --dist=dvd dvd
```

This puts the DVD set under `/home/install/dvd/...`. The first ISO is named `/home/install/dvd/enterprise/3/en/os/rocks-disk1.iso`.

5.8. Enabling RSH on Compute Nodes

The default Rocks configuration does not enable rsh commands or login to compute nodes. Instead, Rocks uses ssh as a drop in replacement for rsh. There may be some circustances where ssh does not have exactly the same semantics of rsh. Further, there may be some users that cannot modify their application to switch from rsh to ssh. If you are one of these users you may wish to enable rsh on your cluster.

Enabling rsh on your cluster has serious security implicatation. While it is true rsh is limited to the private-side network this does not mean it is as secure as ssh. Talk to your local security expert about why this might be a bad idea

Enabling rsh is done by modifying the default kickstart graph. First copy the default `rsh.xml` into the site customization directory:

```
# cp /home/install/rocks-dist/enterprise/3/en/os/arch/build/graphs/default/rsh.xml \
  /home/install/site-profiles/3.2.0/graphs/default/
```

Where *arch* is your architecture ("i386", "ia64", etc).

Now edit `/home/install/site-profiles/3.2.0/graphs/default/rsh.xml` and change the following:

```
<!-- Uncomment to enable RSH on your cluster (this is not very secure!)
 <edge from="slave-node" to="xinetd"/>
 <edge from="slave-node" to="rsh"/>
-->
```

Next follow the instruction and uncomment this block. This will force all appliance types that reference the slave-node class (compute nodes, nas nodes, ...) to enable an rsh service that trusts all hosts on the private side network. This uncommented block should look like this.

```
 <edge from="slave-node" to="xinetd"/>
 <edge from="slave-node" to="rsh"/>
```

To apply your customized configuration scripts to compute nodes, rebuild the distribution:

```
# cd /home/install
# rocks-dist dist
```

Then, reinstall your compute nodes.

5.9. Customizing Ganglia Monitors

5.9.1. Enabling fully aware Ganglia daemons

For maximum performance and scalability, the Ganglia *gmond* daemons on compute nodes in the cluster are run in "deaf" mode. While compute nodes report their own ganglia data to the frontend, they do not listen for information from their peers. This reduces the resource footprint of compute nodes.

Running the compute node monitors in deaf mode means they cannot be queried for cluster state. This may be a problem if your parallel jobs use ganglia data for performance analysis or fault tolerance purposes. If you would like to re-enable Ganglia's full functionality on your compute nodes, follow the instructions below.

Ganglia daemons were switched to the deaf mode by default starting in the Matterhorn Rocks release 3.1.0.

- Add a new site-node file called `replace-ganglia-client.xml`.

  Put the following contents in the new file:

  ```
  <?xml version="1.0" standalone="no"?>
  <!DOCTYPE kickstart SYSTEM "@KICKSTART_DTD@">

  <kickstart>

    <description>
    UCB's Ganglia Monitor system for client nodes in the
    cluster.
    </description>

  <post>

  /sbin/chkconfig --del gmetad

  </post>

  </kickstart>
  ```

- Reinstall your compute nodes. They will now have access to the full monitoring tree. This procedure places the compute nodes on the same level monitoring level as the frontend.

Notes

1. http://www.rocksclusters.org/rocks-documentation/reference-guide/3.2.0/database.html

2. https://www.redhat.com/docs/manuals/enterprise/RHEL-3-Manual/sysadmin-guide/s1-kickstart2-options.html

3. http://www.redhat.com/download/mirror.html

Chapter 6. Downloads

6.1. ISO images and RPMS

To get started, the minimum software required for any architecture is the Rocks Base and the HPC Roll.

6.1.1. Software for x86 (Pentium and Athlon)

Table 6-1. Required Software

Name	Download	md5sum
Rocks Base	(FTP)[1] (HTTP)[1]	d9856142813bef9935ea0e4d325a928e
HPC Roll	(FTP)[1] (HTTP)[1]	a05d6b3a94bac6cc6194aac2cd066a15

The following software is optional.

Table 6-2. Optional Software

Name	Summary	Download	md5sum
Grid Engine Roll	Contains the Grid Engine job queueing system.	(FTP)[1] (HTTP)[1]	a11e4588b766d084badab27a1961f971
Grid Roll	Contains the Globus Toolkit as packaged by the NMI group.	(FTP)[1] (HTTP)[1]	fc7a588043b2a45eff1bb8ae67f37090
Intel Roll	Contains the Intel compiler and MPICH built with the Intel compiler.	(FTP)[1] (HTTP)[1]	e0f7ace865576c7346d9b4d17207f3ad
Area51 Roll	Contains system security related services and utilites.	(FTP)[1] (HTTP)[1]	bf63ebf1655b7a38546b6008d1d6d74b

Name	Summary	Download	md5sum
SCE Roll	Contains the Scalable Cluster Environment from Kasetsart University, Thailand.	(FTP)[1] (HTTP)[1]	fae0c03318a94931fd65bc0208fa7342
Java Roll	Contains a JVM.	(FTP)[1] (HTTP)[1]	0459f555de16832b4b6fcc3a37a0770b
PBS Roll	Contains the Portable Batch System job queueing system.	(FTP)[1] (HTTP)[1]	9a23747018d20074cdf64d2896b38277
Condor Roll	Provides tools for distributed high-throughput computing.	(FTP)[1] (HTTP)[1]	d75d7b04ad0e94b15c3cdb74d57e1777

6.1.2. Software for x86_64 (Opteron)

Table 6-3. Required Software

Name	Download	md5sum
Rocks Base	(FTP)[1] (HTTP)[1]	459bb3bee640426b21f6e89c9384d16d
HPC Roll	(FTP)[1] (HTTP)[1]	305c30d5a33aa978f90442866556190c

The following software is optional.

Table 6-4. Optional Software

Name	Summary	Download	md5sum
Grid Engine Roll	Contains the Grid Engine job queueing system.	(FTP)[1] (HTTP)[1]	62bcf19f98e444f42fff8548b1384261

Name	Summary	Download	md5sum
Grid Roll	Contains the Globus Toolkit as packaged by the NMI group.	(FTP)1 (HTTP)1	20f9cb8f84a590c7757c32ee29d8f8c4

6.1.3. Software for ia64 (Itanium)

Table 6-5. Required Software

Name	Download	md5sum
Rocks Base	(FTP)1 (HTTP)1	a8e98833bc1a952566e1a99f83d9a9e1
HPC Roll	(FTP)1 (HTTP)1	7cda3d986c1bd132e6dbdebfe302909a

The following software is optional.

Table 6-6. Optional Software

Name	Summary	Download	md5sum
Grid Engine Roll	Contains the Grid Engine job queueing system.	(FTP)1 (HTTP)1	2b83aefee1890e4b257282b45b57075e
Grid Roll	Contains the Globus Toolkit as packaged by the NMI group.	(FTP)1 (HTTP)1	3e17949390da04c290f91ff32580c424
Intel Roll	Contains the Intel compiler and MPICH built with the Intel compiler.	(FTP)1 (HTTP)1	8a901b4a7eafe78e27fe2e0b2c4f3047
Java Roll	Contains a JVM.	(FTP)1 (HTTP)1	da9339385a784b692d36b39612f3f253

6.2. CVS Access to the Rocks Source Tree

Anonymous read-only access is provided in two forms:

- Browsing[1] using ViewCVS.

- Using the CVS `pserver`.

6.2.1. Read-Only Access to CVS

The following CVS modules are currently available for anonymous read-only access:

Table 6-7. Rocks CVS Modules

Module Name	Description
rocks-src	All Rocks source code and the repackaged contributed source code.
rocks-doc	All Rocks documentation (documentation source to build this web site, Rocks talks, Rocks press releases, etc.).

To checkout a module, first you need to login:

```
$ cvs -d:pserver:anonymous@cvs.rocksclusters.org:/home/cvs/CVSROOT login
```

This will ask for a password. No password is required, just enter an empty password.

After anonymously logging in, checkout a source tree (where *module* is one of the above CVS modules):

```
$ cvs -d:pserver:anonymous@cvs.rocksclusters.org:/home/cvs/CVSROOT checkout module
```

For example, to checkout all the code within the *rocks-src* module, execute:

```
$ cvs -d:pserver:anonymous@cvs.rocksclusters.org:/home/cvs/CVSROOT checkout rocks-src
```

After initial checkout, you can change into this directory and execute cvs commands without the -d option. For example:

```
$ cvs update
```

6.2.2. Read-Write Access to CVS

Send a copy of your public ssh version 1 key (e.g., `~/.ssh/identity.pub`) to `cvs@rocksclusters.org`, and a short note explaining why you need write access. If you do not already have an ssh key on your client machine, simply run `ssh-keygen -t rsa1` to build one.

Once we receive your public key, we will create an account for you and provide instructions on how to access the repository.

Notes

1.   http://cvs.rocksclusters.org/viewcvs/viewcvs.cgi/

Chapter 7. Frequently Asked Questions

7.1. Installation

1. While trying to bring up a compute node, I boot it from the Rocks CD, and when I plug a monitor into the compute node, I see the error message 'Error opening kickstart file /tmp/ks.cfg. No such file or directory' or I see a screen on the compute node asking me to select a language. What went wrong?

A compute node kickstart requires the following services to be running on the frontend:

1. dhcpd

2. httpd

3. mysqld

4. autofs

To check if httpd and mysqld are running:

```
# ps auwx | grep httpd
# ps auwx | grep mysqld
```

If either one is not running, restart them with:

```
# /etc/rc.d/init.d/httpd restart
```

and/or

```
# /etc/rc.d/init.d/mysqld restart
```

The autofs service is called 'automount'. To check if it is running:

```
# ps auwx | grep automount
```

If it isn't, restart it:

```
# /etc/rc.d/init.d/autofs restart
```

Finally, to test if the Rocks installation infrastructure is working:

```
# cd /home/install
# ./kickstart.cgi --client="compute-0-0"
```

This should return a kickstart file.

And to see if there are any errors associated with kickstart.cgi:

```
# ./kickstart.cgi --client="compute-0-0" > /dev/null
```

2. I successfully installed all the Rolls, but during the last stage after the machine reboots, the system hangs with the error: *GRUB* Loading Stage2.... What went wrong?

This is an intermittent problem we've seen in the lab as well. The installation is fine, except that the grub installation program, for an unknown reason, did not run correctly.

Here is a workaround:

- Put the Rocks Base CD in the frontend and boot the frontend.

- At the boot prompt, type:

  ```
  frontend rescue
  ```

- A screen will appear, click the *Continue* button.

- When you see the shell prompt, execute:

  ```
  # chroot /mnt/sysimage
  ```

- Run the grub installation program:

  ```
  # /sbin/grub-install `awk -F= '/^#boot/ { print $2 }' /boot/grub/grub.conf`
  ```

  This should output something similar to:

  ```
  Installation finished. No error reported.
  This is the contents of the device map /boot/grub/device.map.
  Check if this is correct or not. If any of the lines is incorrect,
  fix it and re-run the script `grub-install'.

  # this device map was generated by anaconda
  (fd0)     /dev/fd0
  (hd0)     /dev/hda
  ```

- Exit the chroot environment:

  ```
  # exit
  ```

- Reboot the frontend.

- Take the CD out of the drive and the frontend should come up cleanly.

3. When I try to install a compute node, the error message on the compute node says, "Can't mount /tmp. Please press OK to restart". What should I do?

Most likely, this situation arises due to the size of the disk drive on the compute node. The installation procedure for Rocks formats the disk on the compute node if Rocks has never been installed on the compute node before.

The fix requires changing the way Rocks partitions disk drives. See Partitioning for details.

4. My compute nodes don't have a CD drive and my network cards don't PXE boot, but my compute nodes do have a floppy drive. How can I install the compute nodes?

You will create a boot floppy that emulates the PXE protocol. This is accomplished by going to the web site:

ROM-o-matic.net[1]

Then click on the version number under the *Latest Development Release* (as of this writing, this is version 5.3.7).

Select your device driver in item 1. Keep the default setting in item 2 (Floppy bootable ROM Image). Then click "Get ROM" in item 4.

We suggest using `dd` to copy the downloaded floppy image to the floppy media. For example:

```
# dd if=eb-5.2.4-pcnet32.zdsk of=/dev/fd0
```

Then run `insert-ethers` on your frontend and boot your compute node with the floppy.

7.2. Configuration

1. How do I remove a compute node from the cluster?

On your frontend end, execute:

```
# insert-ethers --remove="[your compute node name]"
```

For example, if the compute node's name is *compute-0-1*, you'd execute:

```
# insert-ethers --remove="compute-0-1"
```

The compute node has been removed from the cluster.

2. Why doesn't startx work on the frontend machine?

Before you can run startx you need to configure XFree86 for your video card. This is done just like on standard Red Hat machines using the `redhat-config-xfree86` program. If you do not know anything about your video card just select "4MB" of video RAM and 16 bit color 800x600. This video mode should work on any modern VGA card.

3. I can't install compute nodes and I have a Dell Powerconnect 5224 network switch, what can I do?

Here's how to configure your Dell Powerconnect 5224:

You need to set the *edge port* flag for all ports (in some Dell switches is labeled as *fast link*).

First, you'll need to set up an IP address on the switch:

- Plug in the serial cable that came with the switch.

- Connect to the switch over the serial cable.

  The username/password is: admin/admin.

- Assign the switch an IP address:

  ```
  # config
  # interface vlan 1
  # ip address 10.1.2.3 255.0.0.0
  ```

- Now you should be able to access the switch via the ethernet.

- Plug an ethernet cable into the switch and to your laptop.

- Configure the ip address on your laptop to be:

  ```
  IP: 10.20.30.40
  netmask: 255.0.0.0
  ```

- Point your web browser on your laptop to 10.1.2.3

- Username/password is: admin/admin.

- Set the *edge port* flag for all ports. This is found under the menu item: *System->Spanning Tree->Port Settings*.

- Save the configuration.

This is accomplished by going to *System->Switch->Configuration* and typing 'rocks.cfg' in the last field 'Copy Running Config to File'. In the field above it, you should see 'rocks.cfg' as the 'File Name' in the 'Start-Up Configuration File'.

4. When I mirror, I see output that says `No such file ''`.. Is that OK?

Yes, it is ok to see output like:

```
FINISHED --00:54:08--
Downloaded: 9 bytes in 1 files
00:54:08 URL: ftp://www.rocksclusters.org/%2Fpub/rocks/rocks-dist/updates/7.2/en/os/i386/ [9] -
> "www.rocksclusters.org/pub/rocks/rocks-dist/updates/7.2/en/os/i386/.listing" [1]
No such file ''.


FINISHED --00:54:08--
Downloaded: 9 bytes in 1 files
No such directory '/pub/rocks/rocks-dist/updates/7.2/en/os/noarch'.
```

5. The Myrinet network doesn't appear to fully functioning. How do I debug it?

We use High-Performance Linpack (HPL), the program used to rank computers on the Top500[2] Supercomputer lists, to debug Myrinet. HPL is installed on all compute nodes by default.

To run HPL on the compute nodes, see Interactive Mode.

Then it is just a matter of methodically testing the compute nodes, that is, start with `compute-0-0` and `compute-0-1` and make sure they are functioning, then move to `compute-0-2` and `compute-0-3`, etc.

When you find a suspected malfunctioning compute node, the first thing to do is verify the *Myrinet map* (this contains the routes from this compute node to all the other Myrinet-connected compute nodes).

Examine the map by logging into the compute node and executing:

```
$ /usr/sbin/gm_board_info
```

This will display something like:

```
GM build ID is "1.5_Linux @compute-0-1 Fri Apr  5 21:08:29 GMT 2002."


Board number 0:
```

```
  lanai_clockval    = 0x082082a0
  lanai_cpu_version = 0x0900 (LANai9.0)
  lanai_board_id    = 00:60:dd:7f:9b:1d
  lanai_sram_size   = 0x00200000 (2048K bytes)
  max_lanai_speed   = 134 MHz
  product_code      = 88
  serial_number     = 66692
     (should be labeled: "M3S-PCI64B-2-66692")
LANai time is 0x1de6ae70147 ticks, or about 15309 minutes since reset.
This is node 86 (compute-0-1)  node_type=0
Board has room for 8 ports,  3000 nodes/routes,  32768 cache entries
          Port token cnt: send=29, recv=248
Port: Status  PID
   0:   BUSY 12160  (this process [gm_board_info])
   2:   BUSY 12552
   4:   BUSY 12552
   5:   BUSY 12552
   6:   BUSY 12552
   7:   BUSY 12552
Route table for this node follows:
The mapper 48-bit ID was: 00:60:dd:7f:96:1b
gmID MAC Address                                 gmName Route
---- ----------------- -------------------------------- --------------------
   1 00:60:dd:7f:9a:d4                        compute-0-10 b7 b9 89
   2 00:60:dd:7f:9a:d1                        compute-1-15 b7 bf 86
   3 00:60:dd:7f:9b:15                        compute-0-16 b7 81 84
   4 00:60:dd:7f:80:ea                        compute-1-16 b7 b5 88
   5 00:60:dd:7f:9a:ec                         compute-0-9 b7 b9 84
   6 00:60:dd:7f:96:79                        compute-2-13 b7 b8 83
   8 00:60:dd:7f:80:d4                         compute-1-1 b7 be 83
   9 00:60:dd:7f:9b:0c                         compute-1-0 b7 be 84
```

Now, login to a known good compute node and execute /usr/sbin/gm_board_info on it. If the *gmID*'s and *gmName*'s are not the same on both, then there probably is a bad Myrinet component.

Start replacing components to see if you can clear the problem. Try each procedure in the list below.

1. Replace the cable

2. Move the cable to a different port on the switch

3. Replace the Myrinet card in the compute node

4. Contact Myricom[3]

After each procedure, make sure to rerun the mapper on the compute node and then verify the map (with /usr/sbin/gm_board_info). To rerun the mapper, execute:

```
# /etc/rc.d/init.d/gm-mapper start
```

The mapper will run for a few seconds, then exit. Wait for the mapper to complete before you run `gm_board_info` (that is, run `ps auwx | grep mapper` and make sure the mapper has completed).

6. What should the BIOS boot order for compute nodes be?

This is only an issue for machines that support network booting (also called PXE). In this case the boot order should be cdrom, floppy, hard disk, network. This means on bare hardware the first boot will network boot as no OS is installed on the hard disk. This PXE boot will load the Red Hat installation kernel and install the node just as if the node were booted with the Rocks CD. If you select the boot order to place PXE before hard disk to node will repeatedly re-install itself.

7. How do I export a new directory from the frontend to all the compute nodes that is accessible under `/home`?

Execute this procedure:

• Add the directory you want to export to the file `/etc/exports`.

  For example, if you want to export the directory `/export/disk1`, add the following to `/etc/exports`:

  ```
  /export/disk1 10.0.0.0/255.0.0.0(rw)
  ```

  > This exports the directory only to nodes that are on the internal network (in the above example, the internal network is configured to be `10.0.0.0`)

• Restart NFS:

  ```
  # /etc/rc.d/init.d/nfs restart
  ```

• Add an entry to `/etc/auto.home`.

  For example, say you want `/export/disk1` on the frontend machine (named *frontend-0*) to be mounted as `/home/scratch` on each compute node.

  Add the following entry to `/etc/auto.home`:

  ```
  scratch   frontend-0:/export/disk1
  ```

• Inform 411 of the change:

```
# make -C /var/411
```

Now when you login to any compute node and change your directory to /home/scratch, it will be automounted.

8. How do I disable the feature that reinstalls compute nodes after a hard reboot?

When compute nodes experience a *hard* reboot (e.g., when the compute node is reset by pushing the power button or after a power failure), they will reformat the root file system and reinstall their base operating environment.

To disable this feature:

- Login to the frontend
- Create a file that will override the default:

```
# cd /home/install
# cp rocks-dist/enterprise/3/os/i386/build/nodes/auto-kickstart.xml \
site-profiles/3.2.0/nodes/replace-auto-kickstart.xml
```

- Edit the file site-profiles/3.2.0/nodes/replace-auto-kickstart.xml
- Remove the line:

```
<package>rocks-boot-auto<package>
```

- Rebuild the distribution:

```
# cd /home/install
# rocks-dist dist
```

- Reinstall all your compute nodes

  An alternative to reinstalling all your compute nodes is to login to each compute node and execute:

```
# /etc/rc.d/init.d/rocks-grub stop
# /sbin/chkconfig --del rocks-grub
```

7.3. System Administration

1. What security patches should I be running with Rocks?

Every Rocks release is a snapshot of the most up to date software packages from Red Hat for whatever release Rocks is based on. For example, Rocks 2.2 included all the Red Hat 7.2 updates on the of our release. If you use rocks-dist to mirror your distribution from us you will get updates to our software and some Red Hat updates as we see fit to push them out. There is no automated method of updating the software packages on your frontend machine. This means you will have to treat it like any other machine on the Internet and track security updates yourself and updates vulnerable services. Only you can decide what level of security is appropriate for your site. Some sites prefer to secure only network services while others secure the network and local host services.

The default Rocks configuration sets up and a firewall on the frontend and permits only SSH traffic into the cluster. The minimum security requirement is to track SSH updates and apply them to your frontend when needed.

2. Why can't I re-kickstart compute nodes? Shoot-node fails, and power cycling the machine doesn't force a re-install.

Older BIOS versions required the boot image to reside in the first GB of the hard disk. If the boot image, in this case the kickstart kernel image, resides after the first GB of the disk the image will not be loded by LILO. Update your BIOS and this problem should get fixed. Our long term boot loader strategy is to move to GRUB which should fix this problem even on older machines.

3. I see IP addresses not names in my Ganglia graphs. Why is this?

The DNS system in the cluster sometimes causes Ganglia to record bogus node names (usually their IP addresses). To clear this situation, restart the "gmond" and "gmetad" services on the frontend. This action may be useful later, as it will flush any dead nodes from the Ganglia output.

```
# service gmond restart
# service gmetad restart
```

This method is also useful when replacing or renaming nodes in your cluster.

4. When looking at the Ganglia page, I dont see graphs, just the error:

```
There was an error collecting ganglia data (127.0.0.1:8652): XML error: not
well-formed (invalid token) at xxx
```

This indicates a parse error in the Ganglia gmond XML output. It is generally caused by non-XML characters (& especially) in the cluster name or cluster owner fields, although any ganglia field (including node names) with these characters will cause this problem.

We hope future versions of Ganglia will correctly escape all names to make them XML safe. If you have a bad name, to edit `/etc/gmond.conf` on the frontend node, remove the offending characters, then restart gmond.

7.4. Architecture

1. Why is apache running on my compute nodes?

The default configuration for compute nodes is to start the Apache service. This is enabled to allow us to serve (over HTTP) the Linux /proc filesystem to a future monitoring tool. UCB's Ganglia will remain the prefered monitoring tool, but for highly detailed node information only the complete /proc filesystem will suffice. To disable this feature remove the following line from your distribution's configuration graph.

```
<edge from="slave-node" to="apache"/>
```

Notes

1. http://www.rom-o-matic.net/
2. http://top500.org/
3. mailto:help@myri.com

Chapter 8. Resources

8.1. Discussion List Archive

The lastest archive[1] for the npaci-rocks-discussion list.

The archive[2] before our switch to mailman[3].

Notes

1. https://lists.sdsc.edu/pipermail/npaci-rocks-discussion/

2. http://www.rocksclusters.org/mail-archive/threads.html

3. http://www.gnu.org/software/mailman/index.html

Bibliography

Papers

Configuring Large High-Performance Clusters at Lightspeed: A Case Study , Philip M. Papadopoulos, Caroline A. Papadopoulos, Mason J. Katz, William J. Link, and Greg Bruno, December 2002 , Clusters and Computational Grids for Scientific Computing 2002 , (PDF)[1] .

Leveraging Standard Core Technologies to Programmatically Build Linux Cluster Appliances , Mason J. Katz, Philip M. Papadopoulos, and Greg Bruno, April 2002 , CLUSTER 2002:[2] IEEE International Conference on Cluster Computing , (PDF)[3] .

NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters , Philip M. Papadopoulos, Mason J. Katz, and Greg Bruno, Submitted: June 2002 ,   Concurrency and Computation: Practice and Experience[4] Special Issue: Cluster 2001 , (PDF)[5] (PostScript)[6] .

NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters , Philip M. Papadopoulos, Mason J. Katz, and Greg Bruno, October 2001 ,   Cluster 2001[7] , (PDF)[8] (PostScript)[9] .

Wide Area Cluster Monitoring with Ganglia , Federico D. Sacerdoti, Mason J. Katz, Matthew L. Massie, and David E. Culler, December 2003, IEEE International Conference on Cluster Computing, Hong Kong , (PDF)[10] .

Talks

High Performance Linux Clusters, Guru Session, Usenix. , June 2004 , (Powerpoint)[11] .

NPACI All Hands Meeting, Rocks v2.3.2 Tutorial Session , March 2003 , (PDF)[12] (Powerpoint)[13] .

Managing Configuration of Computing Clusters with Kickstart and XML , March 2002 , (Powerpoint)[14] .

NPACI All Hands Meeting, Rocks v2.2 Tutorial Session , March 2002 , (PDF)[15] (Powerpoint)[16] .

NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters , IEEE Cluster 2001, Newport Beach, CA. , October 2001 , (PDF)[17] .

Introduction to the NPACI Rocks Clustering Toolkit: Building Manageable COTS Clusters , NPACI All Hands Meeting, Rocks v2.0 Tutorial Session. , February 2001 , (Powerpoint)[18] .

Press

Itanium Gets Supercomputing Software , 10 April 2003, C|Net ,   (HTML) [19] .

HP Unveils Industry's First Multi-processor Blade Server Architecture for the Enterprise : Enables Customers to Achieve Adaptive Infrastructures , 26 August 2002, Hewlett-Packard Company ,   (HTML) [20] .

Linux, Intel and Dell - Supercomputing at a Major University : Intel e-Business Center Case Study , August 2002, Intel Corporation ,   (PDF) [21] .

NPACI Rocks Simplifies Deployment of Intel Itanium Clusters , 12 June 2002, NPACI & SDSC Online,   (HTML)[22] .

The Beowulf State of Mind, 01 May 2002, Glen Otero, Linux Journal, (HTML)[24] .

Linux on Big Iron, 25 March 2002, eWeek,   (HTML)[25] .

Keck-funded SDSC Satellite Site Proves a Boon to Campus Scientists , 6 March 2002, NPACI & SDSC Online, (HTML)[26] .

Cal-(IT)2, IBM, SDSC & Scripps Inst. of Oceanography Announce COMPAS, December 2001, Supercomputing Online,   (HTML)[27] .

Texas Advanced Computing Center Adds Two New Clusters, December 2001, IT @ UT,   (HTML)[28] .

Cluster-Management Software Developers Preparing for the TeraGrid, July-September, 2001, NPACI Envision, (HTML)[29] .

SDSC AND COMPAQ ANNOUNCE ALLIANCE TO OFFER HIGH-PERFORMANCE COMPUTING CLUSTERS USING NPACI ROCKS CLUSTERING TOOLKIT AS WEB FREEWARE, 9 July 2001, SDSC Press Room, (HTML)[30] .

High Performace Computing for Proliant, 22 June 2001, Compaq Corporation Web Site,   (HTML)[31] .

NPACI Rocks Open-source Toolkit Improves Speed and Ease of Use in Cluster Configuration, 21 March 2001, NPACI & SDSC Online,   (HTML)[32] .

NPACI Releases Rocks Open-source Toolkit for Installing and Managing High-performance Clusters, 1 November 2000, NPACI & SDSC Online,   (HTML)[33] .

Research

A Cache-Friendly Liquid Load Balancer , Federico D. Sacerdoti, Masters Thesis: June 2002 , UCSD Technical Report , (PDF)[34] .

Notes

1. papers/lyon-2002.pdf

2. http://www-unix.mcs.anl.gov/cluster2002/

3. papers/clusters2002-rocks.pdf

4. http://www3.interscience.wiley.com/cgi-bin/jtoc?Type=DD&ID=88511594

5. papers/concomp2001-rocks.pdf

6. papers/concomp2001-rocks.ps

7. http://www.cacr.caltech.edu/cluster2001/

8. papers/clusters2001-rocks.pdf

9. papers/clusters2001-rocks.ps

10. http://www.rocksclusters.org/rocks-documentation/3.2.0/papers/cluster2003-ganglia.pdf

11. talks/guru.ppt

12. talks/npaci-ahm-2003.pdf

13. talks/npaci-ahm-2003.ppt

14. talks/Rocks-MSR.ppt

15. talks/npaci-ahm-2002.pdf

16. talks/npaci-ahm-2002.ppt

17. talks/cluster2001.pdf

18. talks/npaci-ahm-2001.ppt

19. http://news.com.com/2100-1012-996357.html

20. http://www.hp.com/hpinfo/newsroom/press/26aug02c.htm

21. papers/NUSCaseStudy.pdf

22. http://www.npaci.edu/online/v6.12/rocks.html

23. http://linuxprophet.com/

24. http://linuxjournal.com/article.php?sid=5710

25. http://www.eweek.com/article/0,3658,s=703&a=24535,00.asp

26. http://www.npaci.edu/online/v6.5/keck2.html

27. http://www.supercomputingonline.com/article.php?sid=1273

28. http://www.utexas.edu/computer/news/campus/0112/tacc.html

29. http://www.npaci.edu/envision/v17.3/cluster.html

30. http://www.sdsc.edu/Press/01/070901_npacirocks.html

31. http://www.compaq.com/solutions/enterprise/HPC_linux_clusters.html

32. http://www.npaci.edu/online/v5.6/rocks.html

33. http://www.npaci.edu/online/v4.22/NPACI-Rocks.html

34. papers/Sacerdoti-Thesis.pdf

Appendix A. Release Notes

A.1. Release 3.2.0 - changes from 3.1.0

New Feature - Added the Condor Roll. This brings the distributed high-throughput features from the Condor project to Rocks clusters.

New Feature - Added the Area51 Roll. This roll contains security tools and services to check the integrity of the files and operating system on your cluster.

New Feature - Ganglia RSS news event service.

Enhancement - Improved network handling for compute nodes: any interface may be used for the cluster private network, not simply the default "eth0".

Enhancement - Better support for cross-architecture clusters containing x86 and x86_64 machines.

Enhancement - GM device driver now builds and loads on compute nodes that have a custom kernel (e.g., a kernel from kernel.org).

Enhancement - Software RAID for custom compute node partitioning is supported.

Enhancement - Added variables for root and swap partition. If you only want to change the size of root and/or swap, you only have to reassign two XML variables.

Enhancement - The default root partition size has been increased to 6 GB (up from 4 GB).

Enhancement - SGE ganglia monitor added. The state of all SGE jobs can be tracked from the frontend's web page.

Enhancement - PXE support extended to support floppy-based Etherboot and ia64.

Enhancement - EKV uses ssh instead of telnet for security.

Enhancement - New Myrinet MPICH version 1.2.5..12.

Enhancement, Java Roll -- Updated JDK to version 1.4.2_04

Enhancement - Latest software updates recompiled for three architectures from RHEL source rpms.

Enhancement - Automatic MySQL Cluster database backup.

Enhancement - MAC addresses are included for each node in the "Cluster Labels" output.

Enhancement - Frontend rescue mode on the Rocks Base CD enabled. By typing "frontend rescue" at the boot prompt will give you a shell in which you can examine the state of the frontend.

Bug Fix - 411 hardened. More reliable notification of changed files. Correct Makefile encrypts login files on frontend first-boot.

Bug Fix - Multiple CD drives are supported for bringing up a frontend. If you have more than one CD drive connected to your frontend, the installer will now correctly identify which CD you are using.

Bug Fix - Ganglia metrics are now saved on frontend reboot. After a reboot, all Ganglia history will be restored from the previous boot.

Bug Fix - PVFS compiled with -mcmodel=kernel on Opteron.

Bug Fix - XML escape characters (e.g., &, <, >) are supported in the installation screens (e.g., the Cluster Information screen and the Root Password screen).

Bug Fix, Intel Roll - All the Intel compiler libraries are now copied to the compute nodes.

A.2. Release 3.1.0 - changes from 3.0.0

Base Linux packages compiled from publicly available RedHat Enterprise Linux 3 Source (Advanced Workstation) for all architectures.

Switched to Sun Grid Engine 5.3 as the default batch scheduling system.

More Rolls: NMI/Globus Release 4, Java, Condor, Intel compiler rolls available.

New Architectures: Opteron (x86_64) receives first-class functionality.

Enhancement - New MPICH version 1.2.5.2. More efficient MPD parallel job-launcher handling. MPICH2 included by default as well.

Enhancement - Using latest Myrinet mpich-gm 2.0.8 for all architectures.

Enhancement - Updated SSH version 3.7.1 with no login delay.

Enhancement - 411 Secure Information Service used by default, replacing NIS.

Enhancement - Greceptor replaces Gschedule to support mpdring, 411, cluster-top and others. Achieves an order of magnitude better performance than its predecessor.

A.3. Release 3.0.0 - changes from 2.3.2

Based on RedHat 7.3 for x86 and RedHat Advanced Workstation 2.1 for ia64 (all packages recompiled from publicly available source).

Enhancement - Includes RedHat updated RPMS (and recompiled SRPMs for ia64), as of September 3 2003.

Enhancement - Includes kernel version 2.4.20-20.7 for x86 and version 2.4.18e.37 for ia64. Installation environment includes all drivers from the above kernel packages.

Enhancement - New full-featured DNS server and structured ".local" naming conventions within cluster.

Enhancement - Linpack (`xhpl`) works out of the box for Pentium IV and Athlon.

Enhancement - Added remove node feature to `insert-ethers`.

Enhancement - New layout of all MPICH transports. See `/opt/mpich` on the frontend for the new directory structure.

Enhancement - Add support for 'Rolls'. An x86 Rocks frontend install now requires two CDs: the Rocks Base CD and the HPC Roll. An ia64 frontend still requires only one DVD.

Enhancement - Added 'Grid' Roll. This roll includes all packages from NMI R3.1, which includes Globus, the Simple Certificate Authority, and other packages.

Enhancement - High-Performance, fault-tolerant MPD job launcher made available. Automatic MPD ring creation and healing via KAgreement-mpd protocol. (Currently in beta phase for this release)

Enhancement - New 411 Secure Information Service to replace NIS. (Currently in beta phase for this release)

Enhancement - Latest Ganglia version 2.5.4 including better webfrontend speed and streamlined appearance, and more efficient network and disk metric handling.

Enhancement - New PhpSysInfo page on compute nodes, available along with /proc link on Ganglia host view page.

Enhancement - Ganglia command line tool has new --clustersize and --alive=host options.

Enhancement - Kickstart graph now viewable from frontend web page.

Enhancement - For kickstart graph files, new <file> tags made available, with owner="root.root" and perms="ga+r" attributes. Beta phase of RCS-based tracking of all config file changes made for post-section repeatability.

Enhancement - Kickstart graph ordering is explicit. Previously the evaluation order of individual nodes depended on graph weights. Node dependencies can now be explicitly specified using <order> tags in the graph files.

Bug Fix - UNIX manual pages correctly shown (we extend /etc/man.conf)

Bug Fix - NTP now synchronizes all compute node clocks with the frontend.

Bug Fix - `add-extra-nic` now supports multiple NICs per compute node.

Bug Fix - Ganglia RRD metric histories are archived on physical disk and restored on startup.

Bug Fix - Includes NCSA's OpenPBS scalability patches. Can now launch PBS jobs that require more than 64 processors.

Bug Fix - USB keyboard works on all ia64 Tiger boxes

A.4. Release 2.3.2 - changes from 2.3.1

Bug fix - Memory leaks in the broadcastSSH gmetric python module are fixed.

Bug fix - Gmetad will not crash when long ganglia metric names are introduced in the cluster.

Bug Fix - Building MPICH-GM package correctly for AMD Athlon processors.

Bug Fix - Added PBS directories: `/opt/OpenPBS/sched_priv`, `/opt/OpenPBS/sched_logs`, `/opt/OpenPBS/undelivered`.

Bug Fix - Added `userdel` that correctly updates the NIS database.

Enhancement - The Rocks-specific Ganglia metrics are much more efficient with a new Python C extension module that publishes ganglia metrics. The PBS job-queue monitor particularly benefits from this new module.

Enhancement - Updated `rocks-boot` package to contain all the modules from the latest kernel-BOOT package.

Enhancement - The Ganglia monitor-core and webfrontend packages have been updated to the latest version 2.5.3.

Enhancement - The frontend is now a fully configured Rocks cluster build host. By checking out all the Rocks source code on a 2.3.2 frontend, one can build all the source code simply by executing `make rpm` in the directory `.../rocks/src/`.

Enhancement - Updated SGE packages from v5.3p2-4 to v5.3p3-1.

Enhancement - Added Rocks version number to `/home/install/contrib` directory structure.

A.5. Release 2.3.1 - changes from 2.3

Bug fix - Now all the installation device drivers from Red Hat's device disks are included (e.g., Broadcom's Ethernet adapters). In Rocks 2.3, only the device drivers found on Red Hat's installation boot floppy were included.

Bug fix - User-specified NIS domains are now supported (in Rocks 2.3, only 'rocks' NIS domain was supported).

Bug fix - User-specified compute node disk partitioning is now supported.

Bug fix - Sun Grid Engine commd port errors during post installation and Sun Grid Engine warnings during `insert-ethers` were fixed.

Bug fix - Building for Pentium II/III and Athlon added to ATLAS RPM. (on a side note, ATLAS is now built against gcc version 3.2).

Enhancement - PVFS upgraded to version 1.5.6.

Enhancement - More detail has been added to the PBS queue monitoring web page (e.g., can view jobs for only one user and can view nodes for one job). Additionally, the monitoring code now more efficent and it has been hardened due to direct experiences on a 300-node Rocks cluster.

Enhancement - The `bssh` service has been moved from a standalone service to a task managed by the Ganglia `gschedule` service.

Enhancement - The ethernet-based MPICH package has been updated to version 1.2.5.

Enhancement - The Myrinet-based MPICH package has been updated to version 1.2.5..9.

Enhancement - `OpenPBS` version 2.3.16 has replaced PBS. Additionally, the *big memory* patch has been applied. Also, the license for OpenPBS requires registration for those that use OpenPBS, so if you use OpenPBS to manage your computational resources, please register at http://www.OpenPBS.org.

Enhancement - The `maui` package has been updated to version 3.2.5.

Enhancement - Updated Myricom's GM to version 1.6.3.

New Feature - Added a link of the main web page of the frontend that allows one to make sheets of labels with the names of all the compute nodes.

New Feature - An alternative version of `gcc` is now installed (version 3.2 is installed in /opt/gcc32/...).

A.6. Release 2.2.1 - changes from 2.2

Bug fix - pvfs and gm modules don't build because the kernel source and kernel binary RPMs were of a different version.

Bug fix - the partitioning on compute nodes only partitioned the first drive. Now all drives on compute nodes are partitioned with a single partition. The default partitioning is: 4 GB root partition, then /state/partition1 is the remainder of the first drive. The second drive, if present, will have one partition labeled "/state/partition2". The third drive, if present, will have one partition labeled "/state/partition3", etc.

Bug fix - the Rocks CD didn't support as many hardware devices as the RedHat CD. All the hardware modules found on the RedHat CD have been added to the Rocks CD (including many, many more).

A.7. Release 2.2 - changes from 2.1.2

Based on RedHat 7.2.

Upgraded Ganglia (provided by Matt Massie of UC Berkeley) to 2.1.1.

Incorporated PVFS RPMs that were graciously provided to us from Najib Ninaba and Laurence Liew who work at Scalable Systems Pte Ltd in Singapore.

insert-ethers looks to see if a Rocks distribution exists. If it doesn't, insert-ethers rebuilds it.

Upgraded MPICH-GM to version 1.2.1..7b.

Added the "stream" memory bandwidth benchmark.

Added functionality to rocks-dist so distributions can be rebuilt without having to mirror the entire distribution.

Implemented a "greedy" partitioning scheme on compute nodes. The default partitioning is: 4 GB root partition, then /state/partition1 is the remainder of the first drive. The second drive, if present, will have one partition labeled "/state/partition2". The third drive, if present, will have one partition labeled "/state/partition3", etc.

Bug fix - added a "watchdog" timer to kickstart. This reboots a kickstarting node if it can't find a kickstart file. This problem was reported by folks trying to kickstart multiple nodes at the same time.

Bug fix - increased the polling intervals for maui so it won't time out when asking PBS about node status on larger clusters.

Bug fix - makedhcp now adds the full pathname to pxelinux.0 when it builds dhcpd.conf.

Bug fix - create a device node for /dev/cdrom.

Bug fix - /var/log/messages is now appropriately rotated.

A.8. Release 2.1.2 - changes from 2.1.1

Many network and storage drivers have been added to the installation CD. For example, SMC 83c170 EPIC/100 (epic100.o), RTL8139 SMC EZ Card Fast Ethernet (8139too.o) and the Promise SuperTrak Driver (pti_st.o) have all been included (as well as about 100 more).

The cluster configuration web form has been simplified.

The initial kickstart file that is generated from the web form is now streamed directly back to the user (rather than displaying the kickstart file, and then asking the user to save the file). This should finally kill the "I saved my kickstart file on Windows" problem.

An option to manually partition a frontend disk has been added to the cluster configuration web form.

The recursive directory /home/install/install/install/... has been eliminated.

Ganglia's axon is now started before pbs-server, as the pbs-server initialization script asks ganglia for the number of processor in each node when it creates one of it's configuration files.

The latest "stable" release of Myricom's GM (1.5) and MPICH-GM (1.2.1..7) packages.

High-Performance Linpack is now precompiled for Myrinet and Ethernet.

A.9. Release 2.1.1 - changes from 2.1

The main change in this release is the use of an XML-based *kickstart graph* to actively manage kickstart files.

Includes support for IA-64 compute nodes. See the Installing IA-64 Compute Nodes HOWTO[1] for detailed information.

A full X server is now installed on frontend machines.

Added PXE support for kickstarting compute nodes.

All compute nodes now install ATLAS and high-performance Linpack -- some slick software from the Innovative Computing Laboratory[2] at the University of Tennessee.

Modified to the PBS server initialization script to dynamically determine the number of CPUs in compute nodes by querying `ganglia`.

Created a `rocks-pylib` package that contains all the common code used by Rocks command line utilties that access the MySQL database, thus giving all the tools the same basic functionality and common user-specified flags.

Patched Red Hat's installation tool (anaconda) so the default behavior is to get kickstart files with HTTP (Red Hat's default is NFS). This frees the installation procedure of requiring NFS for *any* of its functions.

Rewrite of `insert-ethers` to give it the look and feel of a standard Red Hat installation tool.

Now using Red Hat's `pump` instead of `dhclient` for the DHCP client.

Properly create the default PBS configuration file (`/usr/apps/pbs/pbs.default`) so PBS is now operational "out of the box".

Fixed the annoying, but harmless, message `"socket.error: (101, 'Network is unreachable')"` that was seen on frontend boots.

Fixed the annoying, but harmless, message `"user 0 unknown"` that was seen on a compute node's first boot after kickstarting.

Fixed the 444 permissions problem on `/usr/man` and moved all the Rocks man pages into the new home for Linux man pages (`/usr/share/man`).

A.10. Release 2.1 - changes from 2.0.1

The main change in this release is that thanks to RedHat 7.1, we now use the Linux 2.4 kernel.

Based on RedHat 7.1, instead of 7.0.

Linux 2.4.x kernel, instead of 2.2.x.

Cluster-dist has been replaced with Rocks-dist. Command line arguments are very similar, with the `explode` command being removed and replaced with the `--copy` flag. The new Rocks-dist creates smaller distributions, fixes the problem of expensive mirror updating, and simplifies CD building. Also, it no longer deletes the distribution before rebuilding, this means the build directory (where kickstart files reside) is persistent across distribution builds.

Frontend is now a stratum 10 NTP server, so compute nodes will clock sync to the frontend even when the frontend cannot reach an external time source.

Usher daemon now correctly daemonizes, since we patch the GM code to allow processes to fork.

Symbolic links for Ekv and piece-pipe RPMs removed from the build directory, and `"@Control@"` section added to kickstart files.

Pbs_mom_config.h generated in the kickstart build directory.

Added pre-defined types to the models table in the SQL database. Also, removed dead tables from database, and made column order more human friendly.

Add SQL parsing to cluster-[ps|kill|fork] scripts.

Removed cluster-config-compute, and cluster-config-frontend from the "%post" section in the kickstart file. The cluster-config rpm is now build and installed on the fly on each compute-node.

Bumped lilo timeout to 5 seconds.

Added FORCE_UNIPROCESSOR macro test to force sick SMP machines to kickstart as uniprocessor nodes.

Major revision of insert-ethers. Can now be used to replace nodes, and start at arbitrary ranks and basenames.

Minor maui and pbs bug fixes.

Added gm-mpich SHMEM support to mpi-launch.

A.11. Release 2.0.1 - changes from 2.0

Changed to new directory structure according to RedHat. Existing users will have to delete their mirror of www.rocksclusters.org and re-mirror to pickup the current RedHat directory naming scheme. NOTE: you need the new cluster-dist from www.rocksclusters.org to create a new mirror!

Added support to kickstart laptops (still working on this)

Frontend can now have either a DHCP or static address for the external network. For DHCP the DNS information provided from the external DHCP server is inserted into the Rocks Database and propagated to compute nodes.

Increased default DHCP lease time

Replaced Linux's useradd with create-account.

Force glibc-common RPM to be installed. RedHat 7.0 doesn't install this due to errors in the RPM database.

NIS database gets rebuilt on the frontend once an hour.

Create directories on frontend/compute nodes before putting down SSL and SSH keys. Fixed permission on directories.

Ssh-agent now forwards through nodes

Ssh doesn't use privileged port (makes firewalls happy)

cluster-kickstart set real and effect UID to root so all members of the install group can run shoot-node. Previously only root could do this.

Fixed reinstalls on IDE and SCSI hosts (only IDA host worked before, thanks to a RedHat 7.0 change)

Fixed bssh bug

Notes

1. ../howto/ia64.php
2. http://icl.cs.utk.edu/

Appendix B. Kickstart Nodes Reference

B.1. Rocks Base Nodes

B.1.1. 411

The packages and other common elements of the 411 Secure Information Service.

Parent Nodes:

• base

B.1.2. 411-client

Sets up the 411 Secure Information Service for clients. The 411 service will automatically configure itself when a file is published. Also puts all current 411 files from the frontend into the kickstart file for services that cannot tolerate a single 411 failure. Note that 411 can never guarentee full absolute success at any single time. It only offers consistancy over the long term.

Parent Nodes:

• client

B.1.3. 411-server

Sets up the 411 Secure Information Service for Master nodes. Creates the RSA public and private keys for the cluster, and configures Apache for 411.

Parent Nodes:

• server

B.1.4. apache

Apache HTTP Server

Parent Nodes:

• base

• cluster-db

B.1.5. autofs

AutoFS for automounting home directories over NFS or the loopback device.

Parent Nodes:

• autofs-client

• autofs-server

B.1.6. autofs-client

AutoFS Client

Parent Nodes:

• client

Children Nodes:

• autofs

B.1.7. autofs-server

AutoFS server

Parent Nodes:

- server

Children Nodes:

- autofs

B.1.8. base

Base class for all Rocks nodes. This should include compute nodes, frontend nodes, standalone laptops, computer labs, graphics nodes, nfs servers To achieve this level of flexibility this base class should have edges only to those classes that implement the core of Rocks.

Parent Nodes:

- client
- server

Children Nodes:

- 411
- apache
- c-development
- disk-stamp
- elilo
- fstab
- grub
- installclass
- ip-diag
- keyboard
- logrotate

- node

- node-thin

- rpc

- scripting

- ssh

- ssl

B.1.9. c-development

Minimalist C development support. This is everything you need to compile the kernel.

Parent Nodes:

- base

B.1.10. cdr

CDR Tools (burnings, iso, ripping, mp3 enconding)

Parent Nodes:

- devel

B.1.11. central

A Rocks Cluster Central server. Can kickstart other servers over the network.

Parent Nodes:

- server

B.1.12. client

The 'client node' in the graph. This file is used as a connection point for other XML configuration nodes.

Children Nodes:

- 411-client

- autofs-client

- base

- installclass-client

- ntp-client

- ssh-client

- syslog-client

B.1.13. cluster-db

NPACI Rocks Cluster Database

Parent Nodes:

- server

Children Nodes:

- apache

B.1.14. cluster-db-data

Populate cluster database with initial data

Parent Nodes:

- server

B.1.15. cluster-db-structure

Cluster Database SQL table structure. This used to be generated from a dump of the structure on Meteor. Now we just edit this directly.

Parent Nodes:

• server

B.1.16. devel

The 'devel node' in the graph. This file is used as a connection point for other XML configuration nodes.

Parent Nodes:

• server

Children Nodes:

• cdr
• docbook
• emacs
• fortran-development

B.1.17. dhcp-server

Setup the DHCP server for the cluster

Parent Nodes:

• server

B.1.18. disk-stamp

Take a root partition, and make it ours! This is the key to determining, on reinstalls, if we should save partitions (because the stamp is there) or blow away all the paritions on the disk (because the stamp isn't there).

Parent Nodes:

• base

B.1.19. dns-server

Configures a DNS nameserver for the cluster on the frontend. Both forward and reversed zones are defined using the database.

Parent Nodes:

• server

B.1.20. docbook

DOC Book support (needed to build rolls)

Parent Nodes:

• devel

B.1.21. elilo

IA-64 Bootloader support

Parent Nodes:

• base

B.1.22. emacs

Emacs OS

Parent Nodes:

• devel

B.1.23. fortran-development

Fortran

Parent Nodes:

• devel

B.1.24. fstab

Examine the disks on the box we're installing and see if there are existing, non-root partitions which we should preserve.

Parent Nodes:

• base

B.1.25. grub

IA-32 Boot loader support

Parent Nodes:

• base

B.1.26. install

Do everything needed to kickstart compute nodes or, generally speaking, everything needed to kickstart any node from this machine.

Parent Nodes:

• server

B.1.27. installclass

The base installclass files. This graph node must preceed any other installclass graph nodes.

Parent Nodes:

• base

B.1.28. installclass-client

The client installclass files.

Parent Nodes:

• client

B.1.29. installclass-server

The server installclass files.

Parent Nodes:

• server

B.1.30. ip-diag

TCP/IP Network diagnostic tools.

Parent Nodes:

• base

B.1.31. keyboard

Support USB keyboard for ia64

Parent Nodes:

• base

B.1.32. logrotate

Append rules to logrotate to prune files in /var/log

Parent Nodes:

• base

B.1.33. media-server

Root for the kickstart file on the CD/DVD.

Children Nodes:

• server

B.1.34. node

A node is a machine in the cluster. Node's are on a private network and get DHCP/NIS state from the frontend.

Parent Nodes:

• base

B.1.35. node-thin

Turn off a bunch of packages we think we can live without. They take up too much room on the CD. For DVD based systems this is not required Be the ugly american. the only reason why we do this is because we want to be able to fit a rocks-enabled solution onto a single cdrom and the packages below don't directly help people to run parallel applications

Parent Nodes:

• base

B.1.36. ntp

Network Time Protocol

Parent Nodes:

• ntp-client
• ntp-server

B.1.37. ntp-client

Network Time Protocl

Parent Nodes:

• client

Children Nodes:

• ntp

B.1.38. ntp-server

Network Time Protocl

Parent Nodes:

• server

Children Nodes:

• ntp

B.1.39. perl-development

Perl support

Parent Nodes:

• scripting

B.1.40. python-development

Python support

Parent Nodes:

• scripting

B.1.41. rocks-dist

Distribution building with rocks-dist

Parent Nodes:

• server

B.1.42. rpc

RPC support

Parent Nodes:

• base

B.1.43. scripting

Parent Nodes:

• base

Children Nodes:

• perl-development
• python-development
• tcl-development

B.1.44. server

The 'server node' in the graph. This file is used as a connection point for other XML configuration nodes.

Parent Nodes:

- media-server

- server-wan

Children Nodes:

- 411-server

- autofs-server

- base

- central

- cluster-db

- cluster-db-data

- cluster-db-structure

- devel

- dhcp-server

- dns-server

- install

- installclass-server

- ntp-server

- rocks-dist

- syslog-server

- x11-thin

B.1.45. server-wan

A Rocks Cluster machine that has been kickstarted over the wide area network. Used by the central server to construct a minimal kickstart file.

Children Nodes:

- server

B.1.46. ssh

Enable SSH

Parent Nodes:

• base

B.1.47. ssh-client

SSH Config for compute nodes and other non-frontend appliances. We are using one key pair among all SSH servers in the cluster. This implies we do not care about Man-in-the-Middle attacks. We have subverted the protection for these attacks for several releases (broadcastSSH). This logic should not be in the ssh.xml node so the frontend will generate its own keypair.

Parent Nodes:

• client

B.1.48. ssl

Open SSL support

Parent Nodes:

• base

B.1.49. syslog

Setup Syslog

Parent Nodes:

• syslog-client
• syslog-server

B.1.50. syslog-client

Setup Syslog for client machine to forward messages

Parent Nodes:

• client

Children Nodes:

• syslog

B.1.51. syslog-server

Setup Syslog for server to accept forwarded messages

Parent Nodes:

• server

Children Nodes:

• syslog

B.1.52. tcl-development

Tcl support

Parent Nodes:

• scripting

B.1.53. x11

X11 Desktop applications.

Parent Nodes:

• x11-thin

B.1.54. x11-thin

Trimmed down version of X11 for when we don't need sound all all that other GUI nonsense. I just want to run netscape man.

Parent Nodes:

• server

Children Nodes:

• x11

Appendix C. Errata

C.1. Errata for Rocks Version 3.2.0

- No Entries

C.2. Errata for Rocks Version 3.1.0

- rocks-411-3.1.0-2.noarch.rpm[1]. On the first boot of a frontend, the 411 files are not encrypted due to a bug in the Makefile. Compute nodes installed before the /var/411/Makefile has been run again (from a useradd, etc) will not receive 411 login files. This omission causes problems, as expected, specifically with automount and sge configuration.

C.3. Errata for Rocks Version 3.0.0

- openssh-3.1p1-10.i386.rpm[2], openssh-askpass-3.1p1-10.i386.rpm[3], openssh-clients-3.1p1-10.i386.rpm[4], openssh-server-3.1p1-10.i386.rpm[5]. A security vulnerability in the OpenSSH server was discovered on 2003-09-15. This vulnerability has been used by at least one exploit, and affects this and all previous Rocks releases. Note these packages install a "slow" ssh. If you use Rocks 3.1.0 or higher, you should not upgrade.

```
To upgrade:
# rpm -e openssh-askpass-gnome
# cd /where/i/downloaded/the/openssh-pkgs/
# rpm -Uvh openssh*
# service sshd restart
# cp openssh-* /home/install/contrib/7.3/public/[arch]/RPMS/,
 where [arch] is either "i386" or "ia64".

Rebuild your distribution:
# cd /home/install
# rocks-dist dist

Reinstall all compute nodes:
```

```
# cluster-fork /boot/kickstart/cluster-kickstart
```

- ganglia-python-3.0.1-2.i386.rpm[6]. Fixes a memory leak in the "ps" monitoring module. This package should be upgraded on all cluster nodes, and then the "gschedule" service should be restarted.

- Certain motherboards (perhaps those with hot-plug PCI slots) require a "tmpkernel-pcmcia-cs-3.1.27-18.i386.rpm" package which is not present on the Rocks base CD. This causes the installation to fail. Servers in the ProLiant class (ML370G3, DL380G3) are known to have this issue. We have a Patched Base CD[7] for this problem.

C.4. Errata for Rocks Version 2.3.2

- OpenSSH vulnerability (see errata-3.0.0)

- ganglia-webfrontend-addons-2.3.2-3.noarch.rpm[8]. Small fixes to the runtime display and node listings. Previous versions showed incorrect nodes assigned to a job. This package should be installed on the frontend only.

- ganglia-python-2.3.2-2.i386.rpm[9]. Fixes some issues when monitoring the size of OpenPBS jobs. This package should be installed on the frontend only.

- doc-usersguide-2.3.2-3.noarch.rpm[10]. Added the full set of CD disk labels in PDF format. Unfortunately, they do not fit any CD label stock due to problems in the PDF conversion process. However, they could be made useful with some cropping and adjusting and most closely resemble the "Neato US" cd labels.

C.5. Errata for Rocks Version 2.3.1

- OpenSSH vulnerability (see errata-3.0.0)

- Memory leak in ganglia-python package (see errata-3.0.0).

C.6. Errata for Rocks Version 2.3.0

- No Entries

Notes

1. http://www.rocksclusters.org/errata/3.1.0/rocks-411-3.1.0-2.noarch.rpm

2. http://www.rocksclusters.org/errata/Security/openssh-3.1p1-10.i386.rpm

3. http://www.rocksclusters.org/errata/Security/openssh-askpass-3.1p1-10.i386.rpm

4. http://www.rocksclusters.org/errata/Security/openssh-clients-3.1p1-10.i386.rpm

5. http://www.rocksclusters.org/errata/Security/openssh-server-3.1p1-10.i386.rpm

6. http://www.rocksclusters.org/errata/3.0.0/ganglia-python-3.0.1-2.i386.rpm

7. ftp://ftp.rocksclusters.org/pub/rocks/rocks-3.0.0/i386/rocks-disk1-hp.iso

8. http://www.rocksclusters.org/errata/2.3.2/ganglia-webfrontend-addons-2.3.2-3.noarch.rpm

9. http://www.rocksclusters.org/errata/2.3.2/ganglia-python-2.3.2-2.i386.rpm

10. http://www.rocksclusters.org/errata/2.3.2/doc-usersguide-2.3.2-3.noarch.rpm