Rocks Cluster Distribution: Users Guide

User's Guide for Rocks version 4.1 Edition

Rocks Cluster Distribution: Users Guide :
User's Guide for Rocks version 4.1 Edition
Published May 2006
Copyright © 2006 UC Regents

Table of Contents

List of Tables

Preface

1. User Testimonials

Quotes from Rocks users. Reprinted with permission.

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

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

"The 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."

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

"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."

—Laurence Liew, Scalable Systems Pte Ltd, Singapore.

"I am extremely happy with the performance of the 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."

—Tim Carlson, PhD, PNNL, Richland, WA.

"As the sole cluster operations administrator charged with the management of our 301 node high performance computing cluster, a distribution such as 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 Rocks answering questions & providing assistance."

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

"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."

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

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

San Diego Supercomputer Center, UCSD

- Philip Papadopoulos
- Mason Katz
- Greg Bruno
- Nadya Williams

- Federico Sacerdoti (past member)

Scalable Systems Pte Ltd in Singapore

- Laurence Liew
- Najib Ninaba
- Eugene Tay
- Sivaram Shunmugam
- Tsai Li Ming

High Performance Computing Group, University of Tromso

- Roy Dragseth
- Svenn Hanssen
- Tor Johansen

The Open Scalable Cluster Environment, Kasetsart University, Thailand

- Putchong Uthayopas
- Thadpong Pongthawornkamol
- Somsak Sriprayoonsakul
- Sugree Phatanapherom

Flow Physics and Computation Division, Stanford University

- Steve Jones

Korea Institute of Science and Technology Information (KISTI)

- Jysoo Lee
- Yuchan Park
- Jeongwoo Hong
- Taeyoung Hong
- Sungho Kim

Sun Microsystems

Sun Microsystems has supported Rocks through their gracious hardware donations, most notably the 129-node cluster that was built in two hours on the vendor floor at SC 2003[5].

Advanced Micro Devices

AMD has loaned us several Opteron and Athlon boxes to make sure Rocks always supports their latest chip architectures. In addition, AMD co-sponsored Rocks-A-Palooza I, the first Rocks All Hands Meeting.

Dell

Dell has loaned us several x86, x86_64 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.

SilverStorm Technologies

SilverStorm Technologies (formerly Infinicon Systems) donated 64 nodes worth of Infiniband gear in order to provide an appropriate development platform for the Rocks team to produce the first version of the IB Roll for SilverStorm fabrics.

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.

4.1. 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
- Justin Boggs, Advanced Micro Devices, Sunnyvale CA
- Tim Carlson, PNNL, Richland WA
- Sandeep Chandra, GEON Group, SDSC, San Diego CA
- Robert Konecny, The Center for Theoretical and Biological Physics, UCSD, San Diego CA
- Matt Massie, Millennium Project, UC Berkeley, CA
- Doug Nordwall, PNNL, Richland WA
- Vladmir Veytser, NEES Project, SDSC, San Diego CA
- Matt Wise, Advanced Micro Devices, Sunnyvale 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
5. http://www.rocksclusters.org/movies/rockstar.mov

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.

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, x86_64 and IA-64 architectures.

Processors

- x86 (ia32, AMD Athlon, etc.)
- x86_64 (AMD Opteron and EM64T)
- IA-64 (Itanium)

Networks

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

1.1.2. Minimum Hardware Requirements

Frontend Node

- **Disk Capacity:** 16 GB
- **Memory Capacity:** 512 MB
- **Ethernet:** 2 physical ports (e.g., "eth0" and "eth1")

Compute Node

- **Disk Capacity:** 16 GB
- **Memory Capacity:** 512 MB
- **Ethernet:** 1 physical port (e.g., "eth0")

1.1.3. 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:
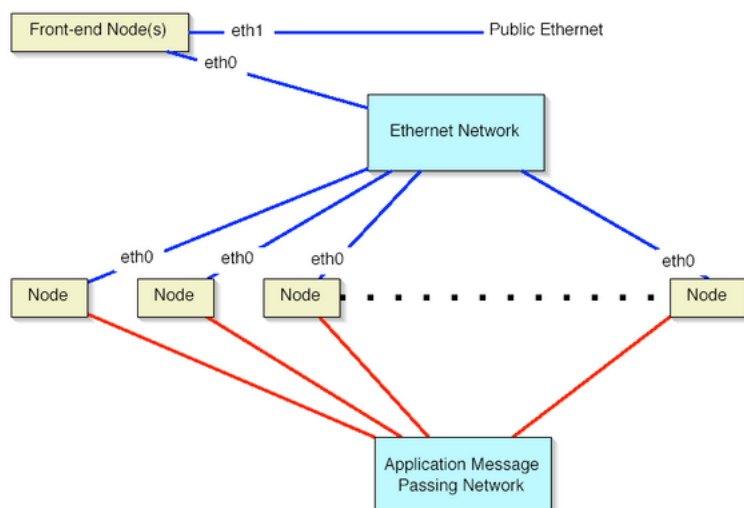
On the compute nodes, the Ethernet interface that Linux maps to *eth0* must be connected to the cluster's Ethernet switch. This network is considered *private*, that is, all traffic on this network is physically separated from the external public network (e.g., the internet).

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.

1.2. Install and Configure Your Frontend

This section describes how to install your Rocks cluster frontend.

 The minimum requirement to bring up a frontend is to have the following rolls: Kernel Roll CD, Base Roll CD, HPC Roll CD and the OS Roll CD(s).

The Base+HPC+Kernel Meta Roll CD can be substituted for the individual Base, Kernel and HPC Rolls.

1. Insert the Kernel Roll CD or the Base+HPC+Kernel Roll CD into your frontend machine and reset the frontend machine.

 For the remainder of this section, we'll use the example of installing a *bare-bones* frontend, that is, we'll be using using the Base+HPC+Kernel Roll, OS - Disk 1 Roll and the OS - Disk 2 Roll.

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

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:
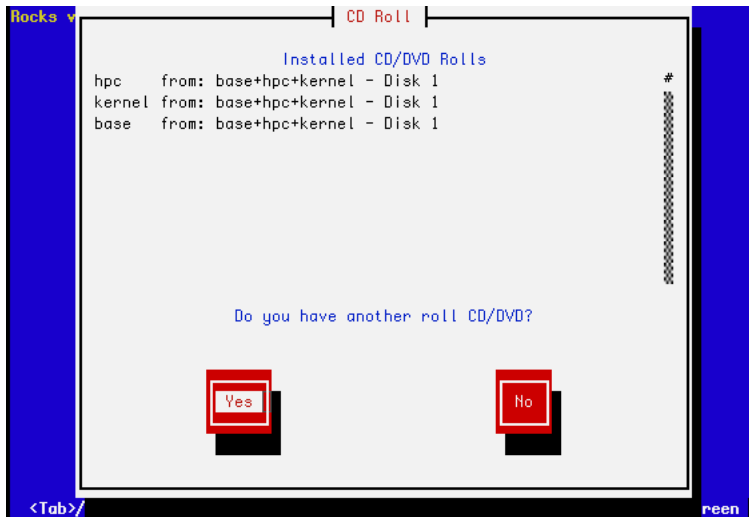
This screen indicates that the required rolls from the Base+HPC+Kernel Roll (Base, HPC and Kernel) have been discovered and added.

You still need to add the OS Rolls to your frontend. To do this, select 'Yes' by pressing the space bar.

4. After the CD/DVD drive ejects the Base+HPC+Kernel Roll media, you'll see the screen:
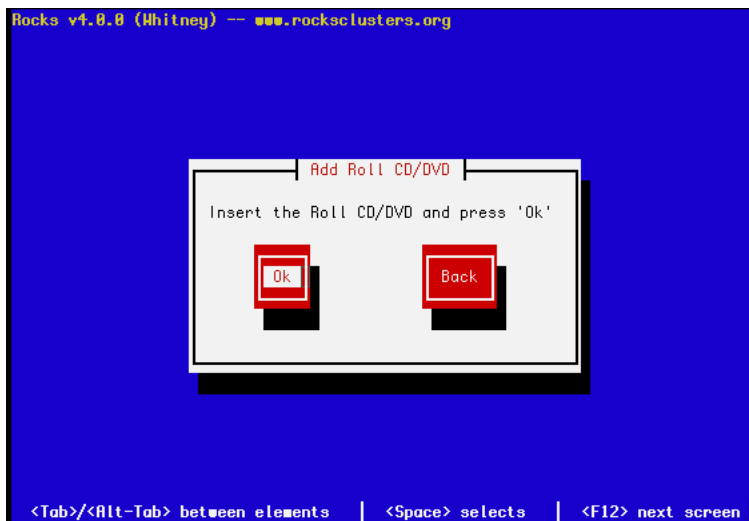
Put the OS - Disk 1 Roll CD into the drive and select 'Ok' by pressing the space bar.

5. When the OS - Disk 1 Roll is discovered, you'll see window with the message:

```
Found Roll 'os 4.0.0 i386'
```

6. After the OS - Disk 1 Roll is added, you'll see a screen that looks like:
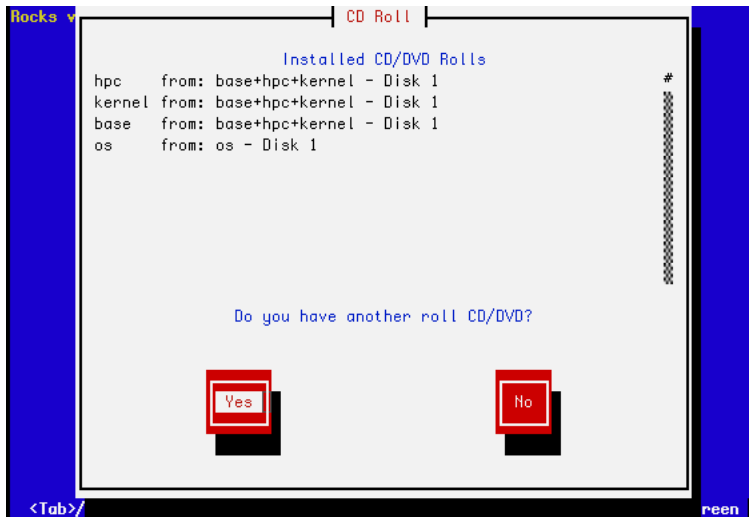
This screen indicates that the required rolls from the Base+HPC+Kernel Roll (Base, HPC and Kernel) and the OS - Disk 1 Roll have been discovered and added.

You still need to add the OS - Disk 2 Roll to your frontend. To do this, select 'Yes' by pressing the space bar.

If this is an Itanium frontend installation, you will only need to add the OS - Disk 1 Roll.

7. Again, you'll see the screen:
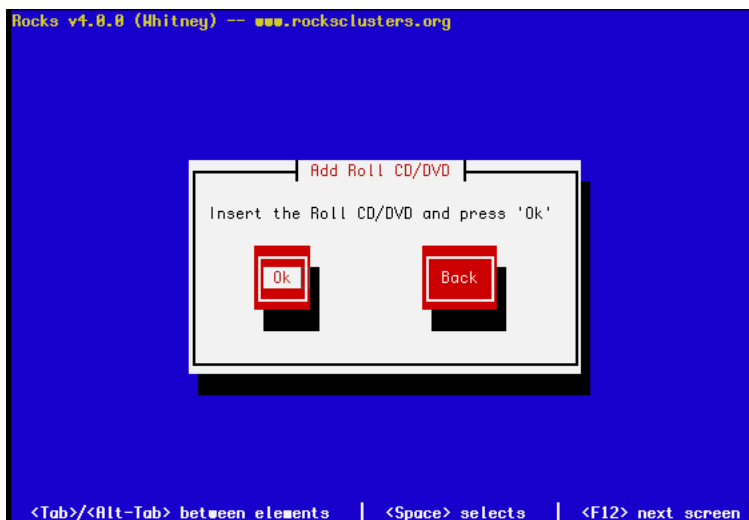
Put the OS - Disk 2 Roll CD into the drive and select 'Ok' by pressing the space bar.

8. When the OS - Disk 2 Roll is discovered, you'll see window with the message:

```
Found Roll 'os 4.0.0 i386'
```

9. After the OS - Disk 2 Roll is added, you'll see a screen that looks like:
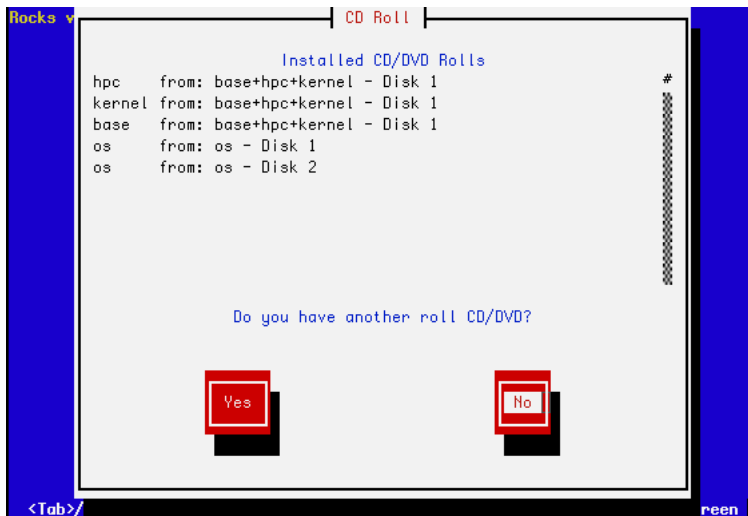
This screen indicates that the required rolls from the Base+HPC+Kernel Roll (Base, HPC and Kernel), the OS - Disk 1 Roll and the OS - Disk 2 Roll have been discovered and added.

At this point, all the required Rolls have been added.

To add an additional Roll, select 'Yes' and input the Roll when prompted (like you did above).

If you have no more Rolls to add, highlight the 'No' button by hitting the Tab key, then select it by hitting the space bar.

10. Then you'll see the *Cluster Information* screen:
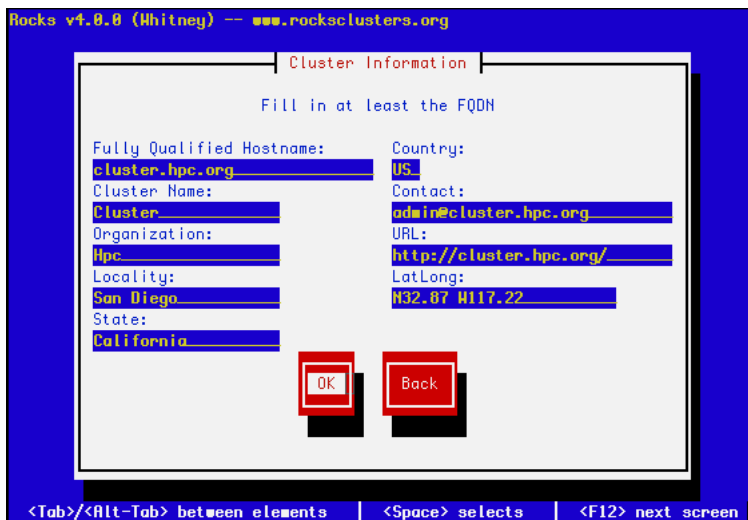
The one important field in this screen is the *Fully Qualified Hostname* (all other fields are optional).

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

If you plan on adding the Grid Roll (or other Globus PKI services) the hostname must be the primary FQDN for your host.

Fill out the form, then hit the 'Ok' button.

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

To select automatic partitioning, hit the Autopartition button. This will repartition and reformat the first discovered hard drive that is connected to the frontend. All other drives connected to the frontend will be left untouched.

The first discovered drive will be partitioned like:

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

Partition Name	Size
/	6 GB
swap	1 GB
/export (symbolically linked to /state/partition1)	*remainder of root disk*

When you use automatic partitioning, the installer will repartition and reformat the *first hard drive* that the installer discovers. All previous data on this drive will be erased. As stated above, all other drives will be left untouched.

The drive discovery process uses the output of `cat /proc/partitions` to get the list of drives.

For example, if the node has an IDE drive (e.g., "hda") and a SCSI drive (e.g., "sda"), generally the IDE drive is the first drive discovered.

But, there are instances when a drive you don't expect is the first discovered drive (we've seen this with certain fibre channel connected drives). If you are unsure on how the drives will be discovered in a multi-disk frontend, then use manual partitioning.

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.

12. 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.
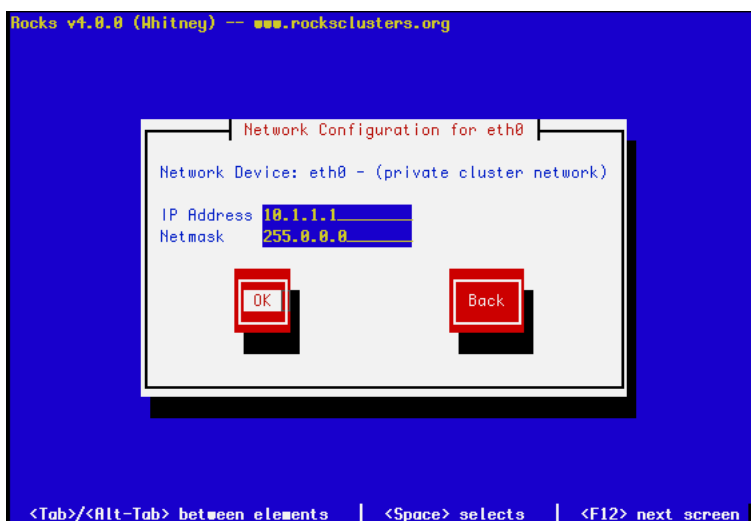
It is recommended that you accept the defaults (by hitting the Tab key until the Ok button is highlighted, then hitting the Enter key).

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

13. 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 v4.0.0 (Whitney) -- www.rocksclusters.org




              ┌──────┤ Network Configuration for eth1 ├──────┐
              │                                               │
              │  Network Device: eth1 - (public cluster network)  │
              │                                               │
              │  IP Address  101.202.88.23___                 │
              │  Netmask     255.255.255.0___                 │
              │                                               │
              │       ┌──────┐              ┌──────┐          │
              │       │  OK  │              │ Back │          │
              │       └──────┘              └──────┘          │
              │                                               │
              │                                               │
              └───────────────────────────────────────────────┘



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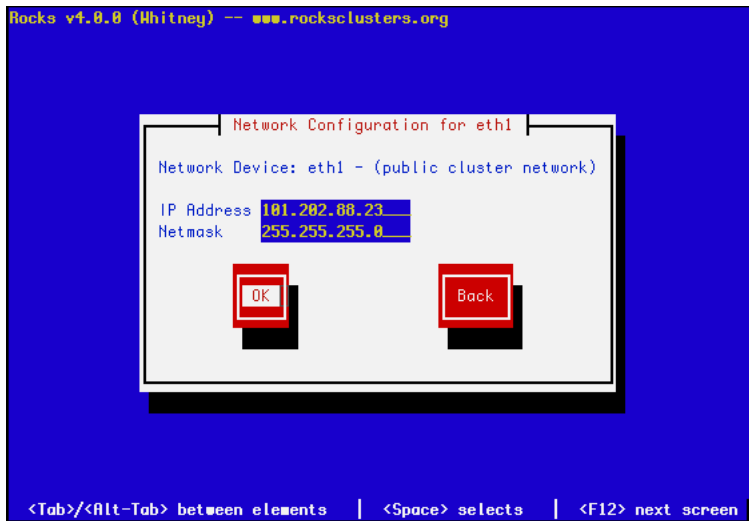
The above window is an example of how we configured the external network on one of our frontend machines.

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

```
Rocks v4.0.0 (Whitney) -- www.rocksclusters.org




              ┌──────┤ Miscellaneous Network Settings ├──────┐
              │                                               │
              │  Gateway:        198.202.88.20___             │
              │  Primary DNS:    198.202.75.26___             │
              │  Secondary DNS:  _____              │
              │  Tertiary DNS:   _____              │
              │                                               │
              │     ┌──────┐          ┌──────┐                │
              │     │  OK  │          │ Back │                │
              │     └──────┘          └──────┘                │
              │                                               │
              │                                               │
              └───────────────────────────────────────────────┘



   <Tab>/<Alt-Tab> between elements   |   <Space> selects   |   <F12> next screen
```
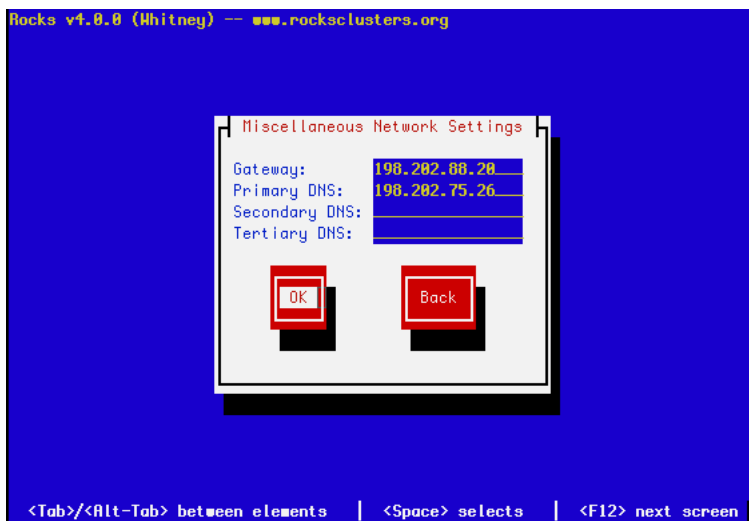
15. Configure the time:
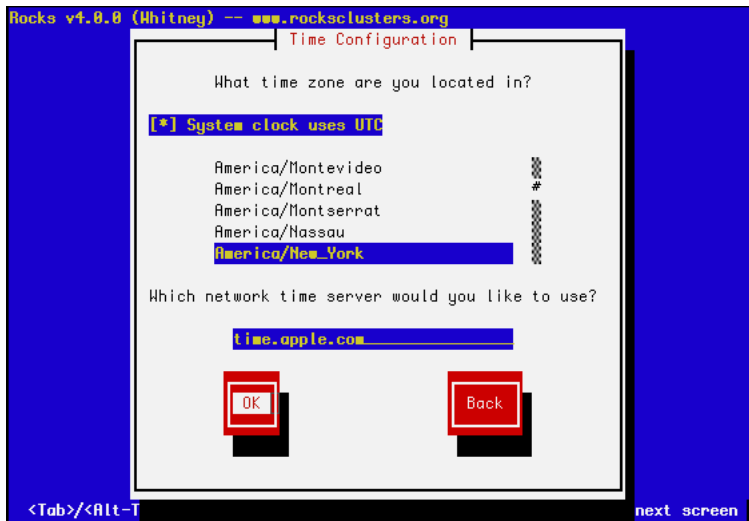
16. Input the root password:
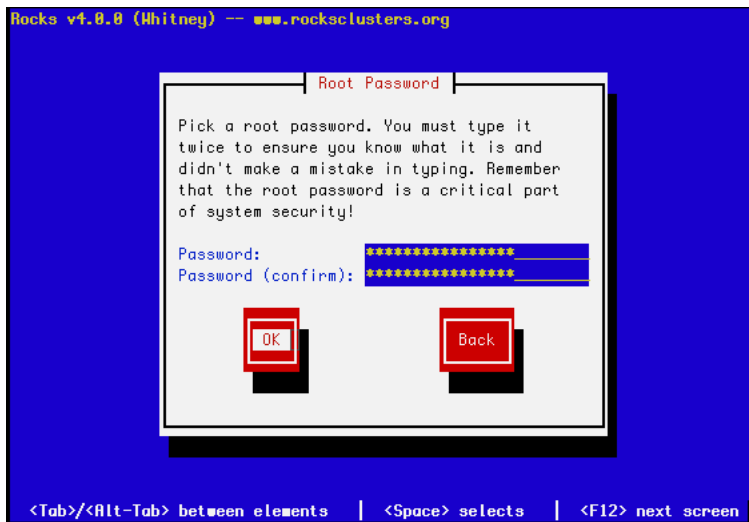
17. The frontend will format it's file systems, then it will ask for each of the roll CDs you added at the beginning of the frontend installation.

```
Rocks v4.0.0 (Whitney) -- www.rocksclusters.org




                      ┤ Install Roll ├

              Put Roll disk 'base+hpc+kernel -
              Disk 1' in the drive


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





 <Tab>/<Alt-Tab> between elements   │  <Space> selects   │  <F12> next screen
```
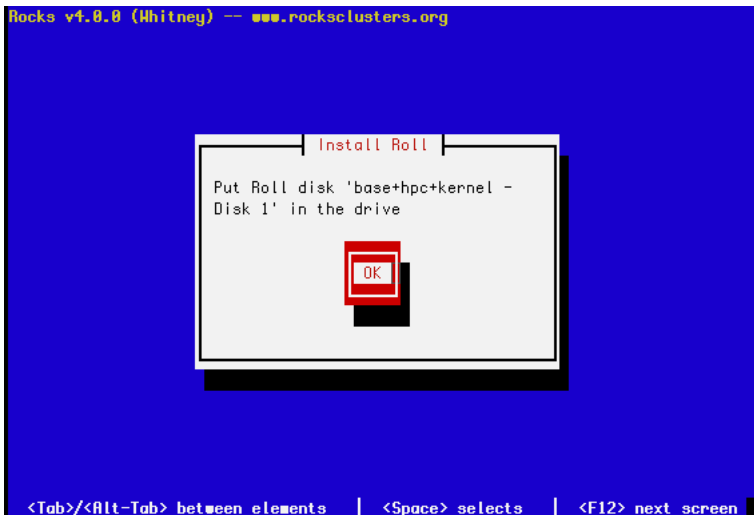
In the example screen above, insert the Base+HPC+Kernel Roll into the drive and select 'Ok'.

The contents of the CD will now be copied to the frontend's hard disk.

After all the Rolls are copied, no more user interaction is required.

18. After the last roll CD is copied, the packages will be installed:

```
Rocks v4.0.0 (Whitney) -- www.rocksclusters.org

                    ┤ Package Installation ├

          Name    : libattr-2.4.16-3-i386
          Size    : 28k
          Summary: Dynamic library for extended attribute support.


          Status: Connecting...

          ┌──────────────────────────────────────────────────┐
          │                       0%                         │
          └──────────────────────────────────────────────────┘
                        Packages      Bytes        Time
          Total    :        656       2362M     0:02:16
          Completed:         71        352M     0:00:20
          Remaining:        585       2010M     0:01:56

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



 <Tab>/<Alt-Tab> between elements   │  <Space> selects   │  <F12> next screen
```
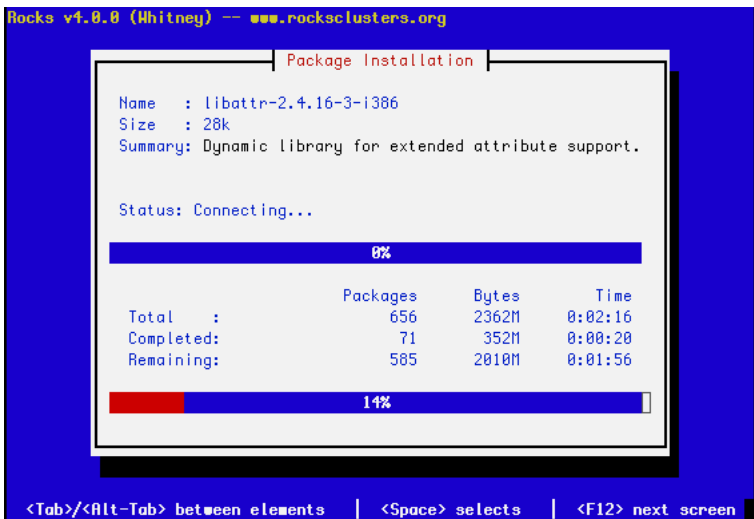
19. Finally, the boot loader will be installed and post configuration scripts will be run in the background. When they complete, the frontend will reboot.

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:
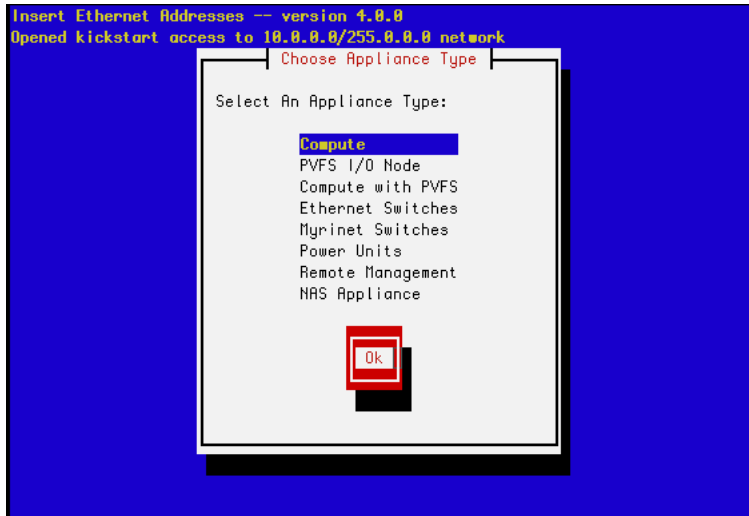
   ![!] If your your frontend and compute nodes are connected via a managed ethernet switch, you'll want to select 'Ethernet Switches' from the list above. This is because the default behavior of many managed ethernet switches is to issue DHCP requests in order to receive an IP address that clients can 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 `F10` 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:
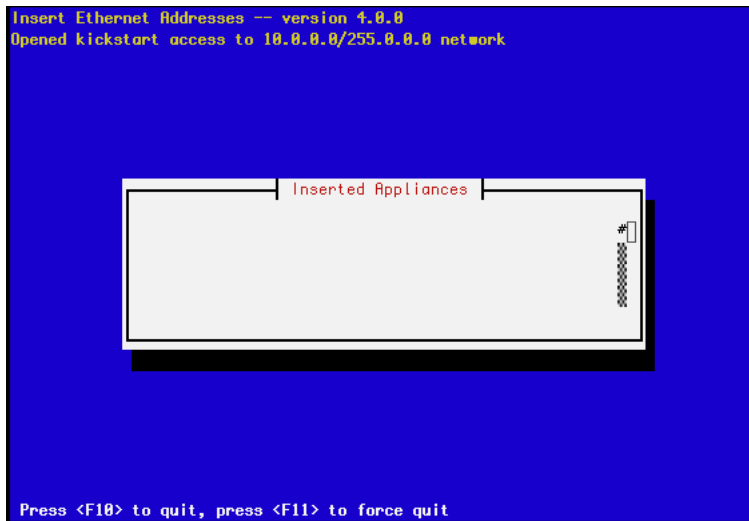
This indicates that `insert-ethers` is waiting for new compute nodes.

4. Take the Kernel Roll CD or the Base+HPC+Kernel Roll CD 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`, DNS and batch system files).

The above screen will be displayed for a few seconds and then you'll see the following:

Figure: insert-ethers has discovered a compute node. The "( )" next to compute-0-0 indicates the node has not yet requested a kickstart file.

You will see this type of output for each compute node that is successfully identified by `insert-ethers`.

As a kickstart file contains 411 keys and other sensitive information in plaintext, it is sent encrypted over the network. In addition, only recognized nodes are allowed to request one. Since insert-ethers is the tool used to identify new nodes, it must be used with care. If security is a concern, be suspicious of unknown MAC addresses in the insert-ethers window.

Figure: The compute node has successfully requested a kickstart file from the frontend. If there are no more compute nodes, you may now quit insert-ethers. Kickstart files are retrieved via https. If there was an error during the transmission, the error code will be visible instead of "*".

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 -p 2200
```

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 'F10' 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.

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 frontend is an x86 (Pentium) and the compute nodes are x86_64 CPUs (Opteron).

1. Retrieve the required Rocks rolls for x86_64 (and optional rolls as desired)

Mount one of the new rolls on /mnt/cdrom. This can be done without actually burning the CD, using the command:

```
# mount -o loop <roll-name>.iso /mnt/cdrom
```

Then copy its contents into the local mirror with:

```
# rocks-dist --install copyroll
```

2. Unmount /mnt/cdrom, and repeat the process for each roll.

3. Rebuild your distribution for the new architecture with the following flags.

```
# 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.

Rocks does not currently support PXE cross-kickstart installs; you must boot non-native compute nodes from a native-architecture Rocks CD that contains the Kernel Roll. In the above example you must install a x86_64 compute node from an x86_64 boot media instead of PXE.

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).

- While the cluster is on-line and before you upgrade the frontend, first backup the compute nodes, and network information from the cluster database. The following two commands are new to Rocks 4.0 and will backup the required information onto the /export partition on the frontend, which will in turn be preserved during the upgrade process. If you skip this step you will need to rediscover all of your compute nodes from scratch, using the same procedure you followed for initial installation. Please note, this procedure will change in the next release of Rocks to further simplify this process.

```
# insert-ethers --dump > /export/insert-ethers.sh
# add-extra-nic --dump > /export/add-extra-nic.sh
# tar -cf /export/ssh.tar /root/.ssh
```

- 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).

- Once the frontend installation is complete login as root and restore your cluster database node and network information.

```
# sh /export/insert-ethers.sh
# sh /export/add-extra-nic.sh
# tar -xf /export/ssh.tar
# insert-ethers --update
# rm /export/insert-ethers.sh /export/add-extra-nic.sh /export/ssh.tar
```

- The last step is to force a re-installation of all of your compute nodes. The following will force a PXE (network install) reboot of all your compute nodes to the release installed on the upgraded frontend machine.

```
# ssh-agent $SHELL
# ssh-add
# cluster-fork /boot/kickstart/cluster-kickstart-pxe
```

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, a process called WAN kickstart. The client frontend will retrieve Rocks Rolls and configuration over the Internet, and use these to install itself.

WAN kickstart requires some initial human coordination between the central server and frontend client. This phase is necessary for each party to authenticate the other. Once the initial kickstart has been completed, however, the client frontend may install or upgrade from the same central without re-authentication (authentication is sticky).

1. Use a Rocks Roll that contains the Kernel Roll for your architecture to boot the client frontend, or alternatively a small bootdisk (also available on the downloads page). When the splash screen appears, type:

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

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 ontact `central-400.rocksclusters.org`, a central server at SDSC.

2. If this is the first time your client frontend has kickstarted from this central server, the central must authenticate the frontend based on its IP address or DNS name. It is also a chance for you to inspect the security credentials of the central server. On the central server, the program `insert-access` must be running with the IP address, network, or domain name of your frontend given as an argument.

```
(On Central)

# insert-access .sdsc.edu
```
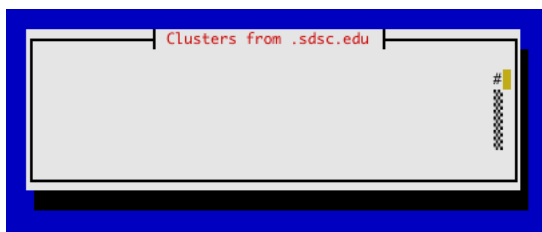

Figure: Central's insert-access program. In this case, frontends from the ".sdsc.edu" domain will be allowed to kickstart from this central; all others will be denied.
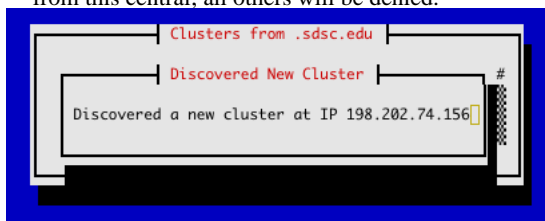
Figure: Central's insert-access detects the frontend's access attempt.
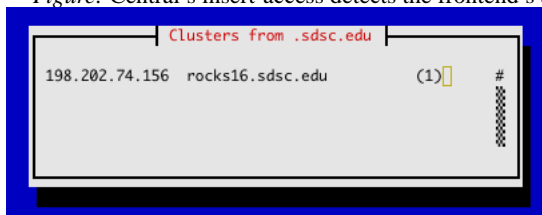
Figure: Central's insert-access shows the IP address, name and number of kickstart attempts this frontend client has made.

3. If this is your first time installing off the central server, you will see a screen describing the central's security credentials. Specifically you are seeing the contents of the mod_ssl apache server certificate on the central. At the same time, the insert-access program on the central is displaying your IP address indicating your kickstart attempt.
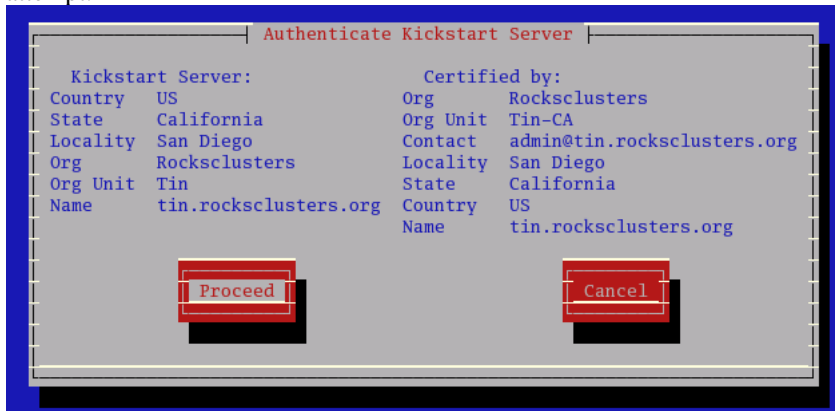
Figure: The WAN kickstart frontend client authentication screen. The left paragraph lists the credentials of the central server; the right paragraph shows the "Certificate Authority" that issued the central's certificate. Pressing "Proceed" will continue the process, choosing "Cancel" will abort the installation.

If the security credentials are to your liking, press the "Proceed" button on the authentication screen. If the central does not have insert-access running, you will see an *Could not get access to server* error. If you have kickstarted from this central before, no insert-access is required on central and you will immediately see the roll choice screen described below.

4. After gaining access to the central, your frontend client will download a kickstart file over HTTPS. You will presently see the roll choice screen:
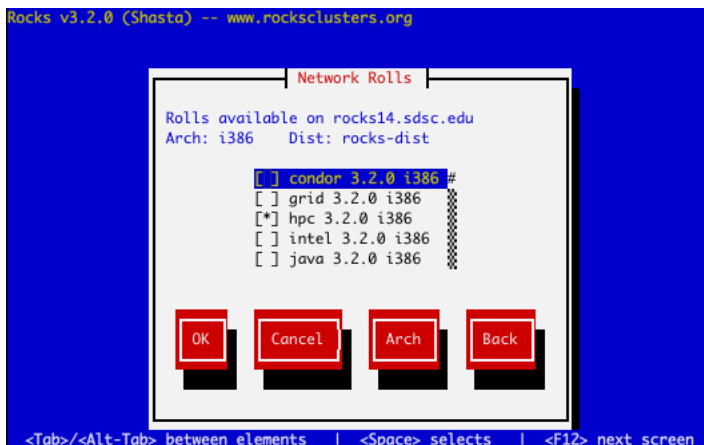
Choose your desired rolls, and press "Ok".

 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.

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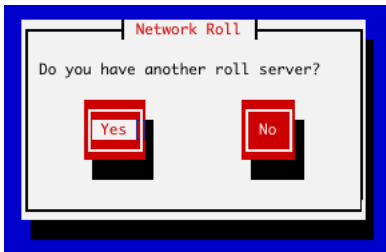
When you are finished adding new network rolls (ones obtained over the network), answer "No" to the new central server question.

6. 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.

7. 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.

8. The installer will then retrieve the chosen rolls, rebuild the distribution with all rolls included, then install the packages. Finally, the installer will proceed with the post-section and other elements of a standard frontend install.

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

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

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/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!

1.6.1. insert-access

The insert-access command controls access to a central server, and restricts which frontends may kickstart from it. Access is divided in two parts: (1) who gets access to *kickstart files*, and (2) who gets access to *RPM packages* (roll access). Kickstart files are the more sensitive of the two since they contain cryptographic keys.

insert-access is always run on a central server.

```
insert-access ... [--all] [--rolls] [--remove] [--stop] [--permanent] ... address
```

In normal form, insert-access grants kickstart access to any client coming from *address* while insert-access is actively running. RPM package access is granted (indefinitely) to any client that has successfully obtained a kickstart file.

```
# insert-access .sdsc.edu
```

> The above command will grant kickstart access to any client coming from the .sdsc.edu domain. The address argument may be any format legal in apache ACLs.

Some options available to insert-access

- *--permanent* Grants unlimited kickstart and roll access to clients from address. Insert-access is non-interactive when this flag is given.
- *--remove, --stop* Removes kickstart and roll access to address. If address is not specified, removes kickstart and roll access to the 'all' client. This is the compliment of --permanent.
- *--rolls* Allows unlimited RPM package access to clients from address. Kickstart access is not given.

> If you want to get rolls from multiple central servers, you must have at least --rolls access on all of them. We often allow all clients to obtain rolls from our centrals by running:
>
> ```
> # insert-access --rolls --all
> ```

- *--all* A synonym for the "all" address. Specifies all possible clients. If this flag is used, no address needs to be given.

1.7. Frontend Central Server

A Central Server is a Rocks Frontend node that can kickstart other frontends and provide rolls over the network, in a process called WAN kickstart. All Rocks frontends have the ability to act as central servers.

From Rocks 3.3.0 onwards, the standard distribution in /home/install/rocks-dist contains a distro suitable for WAN kickstart. The only steps you must take is to open "www" and "https" access on your frontend for the purpose of RPM package transfer. See Enable WWW access.

> 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.

1.7.1. Adding Rolls to serve on a Central

You may wish to serve rolls from your central that you have not installed during installation. All frontends will serve the rolls they were built with to client frontends, but often it is advantageous to serve other rolls as well.

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

2. # rocks-dist copyroll

3. Unmount the CD.

(repeat for each desired roll)

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

If you have a *.iso image of a roll, you can use the "mount -o loop <name>.iso /mnt/cdrom" command instead of burning the CD and mounting it as in step 1.

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:

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

2.1.2. Cluster-Fork

Cluster-Fork runs a command on compute nodes of your cluster.

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.

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[2] 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 2.1.2 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. Running Linpack

2.3.1. 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[3].

2.3.2. 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.

If you see in a error message in your output file that looks like:

```
p2_25612: p4_error: interrupt SIGSEGV: 11
p4_22913: p4_error: interrupt SIGSEGV: 11
Broken pipe
Broken pipe
```

or:

```
p2_25887: (6.780981) xx_shmalloc: returning NULL; requested 13914960 bytes
p2_25887: (6.781052) p4_shmalloc returning NULL; request = 13914960 bytes
You can increase the amount of memory by setting the environment variable
P4_GLOBMEMSIZE (in bytes);
```

Then you'll need to increase the size of P4_GLOBMEMSIZE. To do that, edit sge-qsub-test.sh and increase value in the line:

```
#$ -v P4_GLOBMEMSIZE=10000000
```

Then resubmit the job to SGE.

Notes

1. examples/HPL.dat

2. examples/sge-qsub-test.sh

3. 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.

```
# system-config-display
# startx
```

Once this is done, a standard RedHat desktop environment will appear. Point a 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 do 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 a browser on the cluster with the following command. The ssh process will setup an encrypted channel for the browser to operate through.

   $ firefox --no-remote &

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.

```
...
# Uncomment the lines below to activate web access to the cluster.
#-A INPUT -m state --state NEW -p tcp --dport https -j ACCEPT
#-A INPUT -m state --state NEW -p 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.
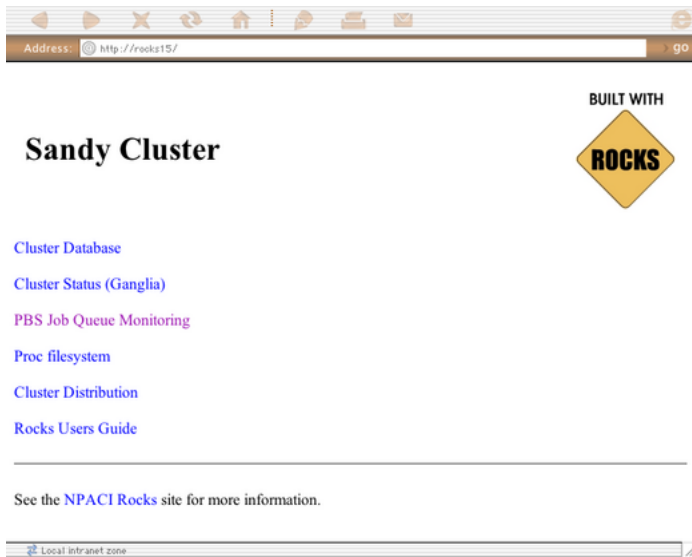
3.2. The Cluster Database

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].

3.3. Cluster Status (Ganglia)

The webpages available from this link provide a graphical interface to live cluster information provided by Ganglia monitors[3] 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.

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.
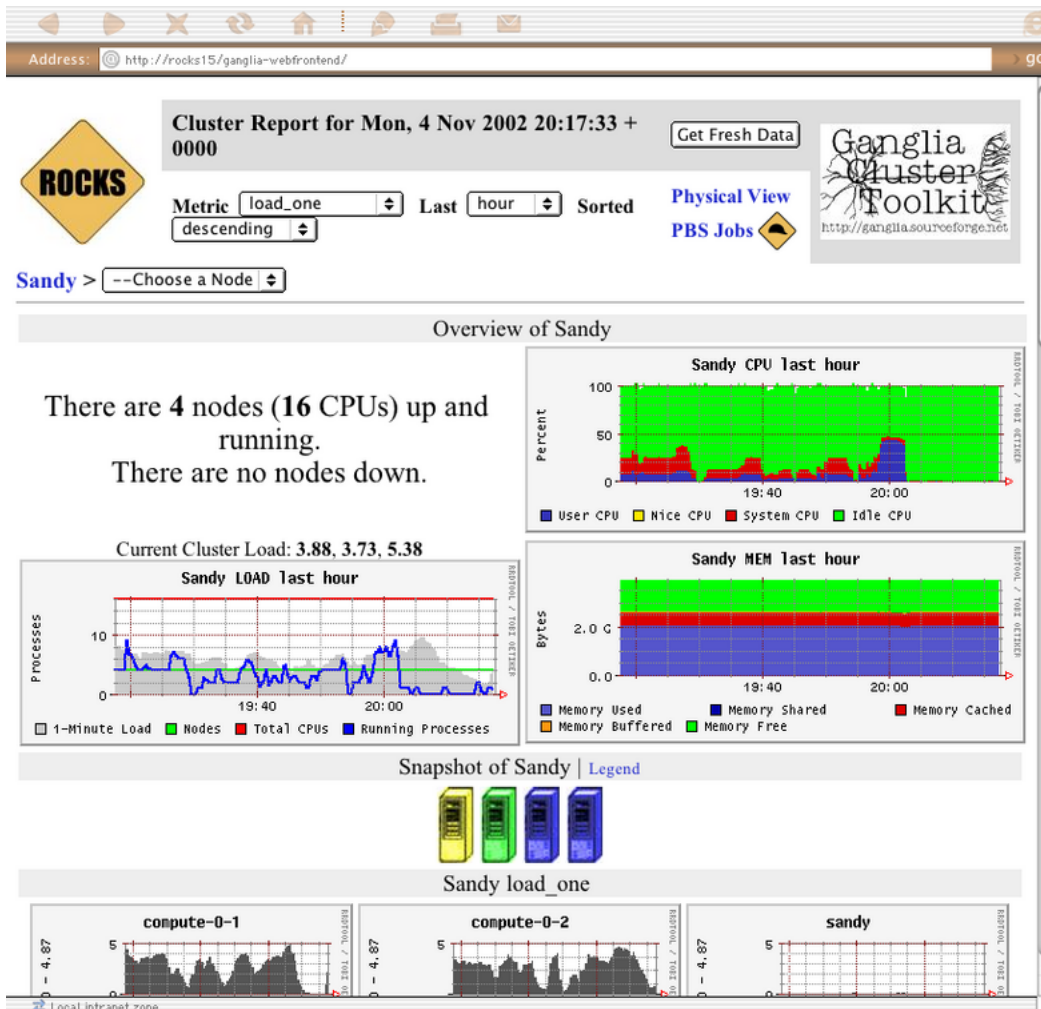
The Rocks Cluster Group maintains a similar web page called *Meta* that collects Ganglia information from many clusters built with Rocks software. It may give you a glimpse of the power and scalability of the Ganglia monitors. The meta page is available at http://meta.rocksclusters.org/.

Ganglia[5] 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.

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/. 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/4.1/database.html

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

3. http://ganglia.sourceforge.net/

4. http://meta.rocksclusters.org/

5. http://ganglia.sourceforge.net/

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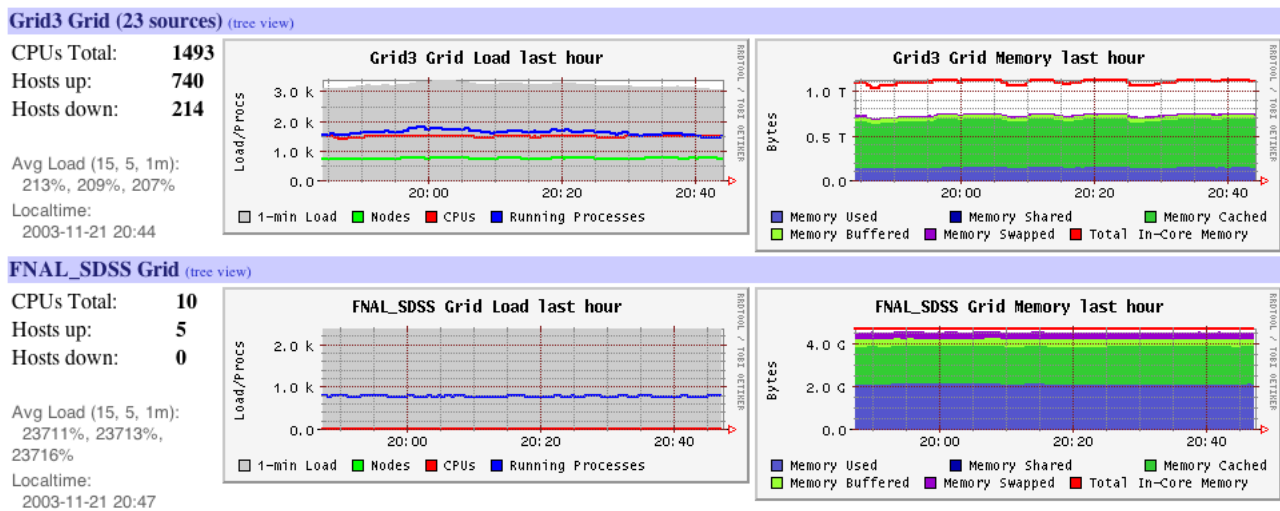
Chapter 4. Cluster Services

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 scalability, security, low-latency when changes occur, and resilience 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.

To force all 411 files to be re-encrypted and change alerts sent to all compute nodes, run this on the frontend

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

The 411 service uses IP broadcast messages on your cluster's private network to achieve optimal performance.

To force all compute nodes to retrieve the latest files from the frontend, execute:

```
# cluster-fork 411get --all
```

4.2.2. Structure

4.2.2.1. Listener

Client nodes listen on the IP broadcast channel for "411alert" messages from the master (the frontend). 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 client 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. 411 alerts are resent periodically to compensate for lost messages.
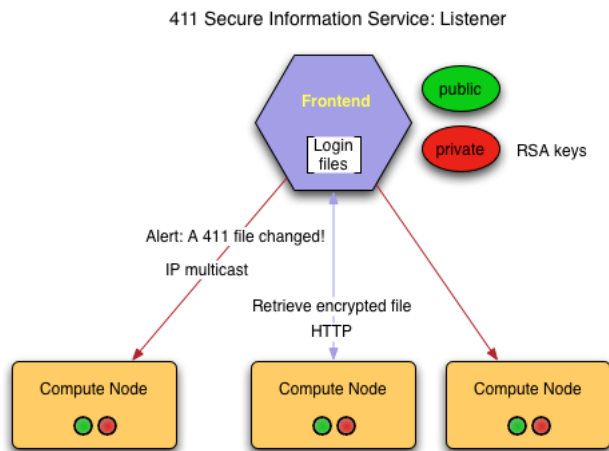
Figure: The 411 listener architecture. When the frontend changes a login file, the 411 makefile sends out a broadcast alert to the cluster nodes notifying of the event. Nodes then pull the file from the frontend via HTTP.

To prevent flooding the master server with requests, the listeners keep an estimate of the cluster size, and use it calibrate a random backoff for their requests. A client does not immediately request the changed file, but waits some amount of time before asking for it.

411 is akin to a distributed database, and is not a centralized lookup service like NIS. While scalable, 411 does not provide instantaneous distribution of new files. The delay between running the 411 makefile and all nodes receiving the changed file depends on cluster size. A large password file on a cluster with many nodes can take up to a minute to fully synchronize on all nodes.

4.2.2.2. Poller

In addition to the 411 listener agent on nodes, another agent retrieves all messages from the frontend at a regular interval, regardless of whether the files have changed or not.

The polling interval at nodes is set to 5 hours by default. To change it, set the "interval" option in the 411 configuration file on nodes.

```
/etc/411.conf:

...
<interval sec="300">
...
```

There is automatically some randomness introduced to the polling interval to avoid storms on the network.

The poller is implemented as a `greceptor` 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.

Because the 411 polling daemon runs as root on client nodes, it is essential that the 411 http directory on a master server be writable only by root to avoid any chance of *privilege elevation*. This is the default in Rocks.

4.2.3. Security

See the 411 paper in the Bibliography section for details of 411 security mechanisms.

4.2.4. 411 Groups

Beginning in Rocks 3.3.0, 411 has the ability to send messages to subsets of the cluster. This facility, called 411 groups, allows us to distribute different files to nodes depending on their type. The group mechanism depends on the client nodes specifying group names in their local 411 configuration file; these are called the client's "registered" groups.

There is no per-group key in 411. The groups mechanism is only a convenience feature, without strong security to enforce it. Specifically, a node can eavesdrop on messages for a foreign group that it is not a member of.

Group names are multi-level, and resemble file paths. By default, every node is a member of the '/' group (corresponding to the traditional top-level 411 group), and the '/Membership' group, where *membership* is the node membership in the frontend database, such as "Compute" or "NAS".

A special Makefile called /var/411/Group.mk is available to help you setup and maintain 411 groups. After editing this file to specify which files go to which group, run

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

To activate the 411 group makefile actions.

By default, nodes are members of a group with the same name as their *Membership*. For example compute nodes are automatically a member of the group "Compute". A sample `411.conf` file with several groups looks like:

```
<!-- Configuration file for the 411 Information Service -->
<config>
<master url="http://10.1.1.1/411.d/" score="0"/>
<group>/light/blue</group>
<group>Compute</group>
</config>
```

Multi-element group names have a simple inheritance model: specific groups imply more general ones. For example, if you are a member of the group `/compute/light`, you will automatically be interested in messages in group "/compute/light" and "/compute". You will not be interested in messages from group "/compute/heavy". In this case "/compute/light" is the specific group, and "/compute" is the more general one.

411 Groups

Offered 411 msgs from master:
- /etc.passwd
- /blue/etc.sudoers
- /Compute/heavy/etc.crontab
- /NAS/etc.exports
- /Compute/etc.shadow
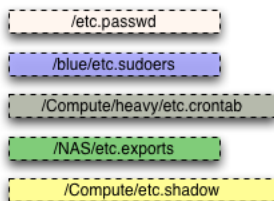
Registered Groups on client:
- /
- /blue
- /Compute/light

Written on client:
- /etc.passwd
- /blue/etc.sudoers
- /Compute/etc.shadow
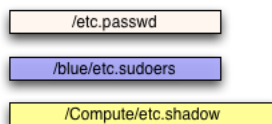
Figure: 411 groups. The client uses registered groups from its local configuration file to filter a stream of offered messages from the master. The messages with the dashed border represent newly changed 411 files on the master, the solid messages at the bottom have been chosen by the client. Note that group "/compute/light" implies "/compute".

4.2.5. Commands

4.2.5.1. 411get

```
411get [--all] [--master=url] [--conf] [--pub] [--shared] [--local] [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:

- *--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.

- *--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.

- *--file, --local* Assume the file is local, ie present in the current directory. Does not use http to retrieve the file. Decrypts and prints the file contents.

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

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

- *--shared* The location of the cluster shared key. Defaults to "/etc/411-security/shared.key"

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

4.2.5.2. 411put

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

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

The following options are available:

- *--chroot=dir* Turn "dir" into the root directory of the destination file. This allows files to be located in a different place on the master and clients.

  ```
  Example:
  411put --chroot=/var/411/groups/compute /var/411/groups/compute/etc/passwd
  ```

  Will put "/var/411/groups/compute/etc/passwd" on compute nodes as "/etc/passwd".

- *--chroot-here* A convenience option, equivalent to --chroot=$PWD.

- *--group=name* A 411 group for this file. Clients will ignore 411 messages in groups which they are not a part of. Allows 411 files to be published to a subset of the cluster. Name is path-like: "Compute/green", or "/Compute/green". Spaces are ok: "a space/yellow" is a valid group name as well.

- *--comment* The comment character for this file. Used to place a descriptive header without disrupting normal operations. Often set to "#". Default is none.

- *--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 IP broadcast channel (255.255.255.255).

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

- *--pub* The location of the 411 master public RSA key. Defaults to a 1024 bit key in "/etc/411-security/master.pub". This file should have permissions 0444 (read by all) and be owned by root.

- *--priv* The location of the 411 master private RSA key. Defaults to a 1024 bit key in "/etc/411-security/master.key". This file should exist only on the master node and be owned by root and have permissions 0400 (read only by root).

- *--make-shared-key* Generate a new random shared key. The key is a 256 random number encoded in base64.

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.

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/4.1/arch/RPMS
```

Where *arch* is your architecture ("i386", "x86_64" or "ia64").

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

```
# /home/install/site-profiles/4.1/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/4.1/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. Adding Applications to Compute Nodes

If you have code you'd like to share among the compute nodes, but your code isn't in an RPM (or in a roll), then this procedure describes how you can share it with NFS.

On the frontend, go to the directory `/export/apps`.

```
# cd /export/apps
```

Then add the files you'd like to share within this directory.

All files will be available on the compute nodes under: `/share/apps`. For example:

```
# cd /export/apps
# touch myapp
# ssh compute-0-0
# cd /share/apps
# ls
myapp
```

5.4. 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]).

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 per node.

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

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

  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=
  ```

- To apply your changes, reinstall the nodes that you have defined an additional interface (use `shoot-node`).

5.5. Compute Node Disk Partitioning

5.5.1. Default Disk Partitioning

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

Only the root disk (the first discovered disk) is partitioned by default. To partition all disks connected to a compute node, see the section Forcing the Default Partitioning Scheme for All Disks on a Compute Node.

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

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

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

5.5.2. 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/4.1/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 compute node `compute-0-0` to your specification above, you'll need to first remove the partition info for `compute-0-0` from the database:

```
# rocks-partition --list --delete --nodename compute-0-0
```

Then you'll need to remove the file `.rocks-release` from the first partition of *each disk* on the compute node. Here's an example script:

```
for i in `df | awk '{print $6}'`
do
 if [ -f $i/.rocks-release ]
 then
  rm -f $i/.rocks-release
 fi
done
```

Save the above script as `/home/install/sbin/nukeit.sh` and then execute:

```
# ssh compute-0-0 'sh /home/install/sbin/nukeit.sh'
```

Then, reinstall the node:

```
# ssh compute-0-0 '/boot/kickstart/cluster-kickstart'
```

5.5.3. 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/4.1/nodes/
# cp skeleton.xml replace-auto-partition.xml
```

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

```
<main>
 <part> / --size 8000 --ondisk hda </part>
 <part> swap --size 1000 --ondisk hda </part>
 <part> /mydata --size 1 --grow --ondisk hda </part>
</main>
```

This will set up an 8 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` keyword, see Red Hat Enterprise Linux 4: 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 change the partitioning scheme, *all* partitions will be removed and reformatted.

Then apply this configuration to the distribution by executing:

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

To reformat compute node `compute-0-0` to your specification above, you'll need to first remove the partition info for `compute-0-0` from the database:

```
# rocks-partition --list --delete --nodename compute-0-0
```

Then you'll need to remove the file `.rocks-release` from the first partition of *each disk* on the compute node. Here's an example script:

```
for i in `df | awk '{print $6}'`
do
 if [ -f $i/.rocks-release ]
 then
  rm -f $i/.rocks-release
 fi
done
```

Save the above script as `/home/install/sbin/nukeit.sh` and then execute:

```
# ssh compute-0-0 'sh /home/install/sbin/nukeit.sh'
```

Then, reinstall the node:

```
# ssh compute-0-0 '/boot/kickstart/cluster-kickstart'
```

5.5.4. Forcing the Default Partitioning Scheme for All Disks on a Compute Node

This procedure describes how to force all the disks connected to a compute node back to the default Rocks partitioning scheme regardless of the current state of the disk drive on the compute node. the Rocks compute node default partitioning scheme.

The root disk will be partitioned as described in Default Partitioning and 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*

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

```
# cd /home/install/site-profiles/4.1/nodes/
# cp skeleton.xml replace-auto-partition.xml
```

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

```
<main>
 <part> force-default </part>
</main>
```

Then apply this configuration to the distribution by executing:

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

To reformat compute node `compute-0-0` to your specification above, you'll need to first remove the partition info for `compute-0-0` from the database:

```
# rocks-partition --list --delete --nodename compute-0-0
```

Then you'll need to remove the file `.rocks-release` from the first partition of *each disk* on the compute node. Here's an example script:

```
for i in `df | awk '{print $6}'`
do
 if [ -f $i/.rocks-release ]
 then
  rm -f $i/.rocks-release
```

```
 fi
done
```

Save the above script as /home/install/sbin/nukeit.sh and then execute:

```
# ssh compute-0-0 'sh /home/install/sbin/nukeit.sh'
```

Then, reinstall the node:

```
# ssh compute-0-0 '/boot/kickstart/cluster-kickstart'
```

After you have returned all the compute nodes to the default partitioning scheme, then you'll want to remove replace-auto-partition.xml in order to allow Rocks to preserve all non-root partition data.

```
# rm /home/install/site-profiles/4.1/nodes/replace-auto-partition.xml
```

Then apply this update to the distribution by executing:

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

5.5.5. Forcing Manual Partitioning Scheme on a Compute Node

This procedure describes how to force a compute node to always display the manual partitioning screen during install. This is useful when you want full and explicit control over a node's partitioning.

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

```
# cd /home/install/site-profiles/4.1/nodes/
# cp skeleton.xml replace-auto-partition.xml
```

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

```
<main>
 <part> manual </part>
</main>
```

Then apply this configuration to the distribution by executing:

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

The next time you install a compute node, you will see the screen:

```
Rocks v4.0.0 (Whitney) -- 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 interact with the above screen, from the frontend execute the command:

```
# ssh compute-0-0 -p 2200
```

Then, to manually partition the node, select `Disk Druid`.

5.6. Creating a Custom Kernel RPM

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

- On the frontend, check out the Rocks source code. See Read-Only Access to CVS for details.

- Change into the directory:

  ```
  # cd rocks/src/roll/kernel/src/kernel.org
  ```

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

  ```
  # wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.9.tar.gz
  ```

- Create a kernel "config" file and put it in `config-<version>`

     The *<version>* number must match the version number of the kernel source. For example, if you downloaded `linux-2.6.9.tar.gz`, the name of the config file must be `config-2.6.9`.

You'll want to ensure that the following line is in your `config-<version>` file:

```
CONFIG_DEBUG_KERNEL=y
```

Otherwise, the kernel modules will contain debug info and be quite large.

- Update `version.mk`.

  The file `version.mk` has the following contents:

  ```
  NAME            = kernel
  RELEASE         = 1

  VERSION         = 2.6.9
  SMP             = 1
  ```

  The `VERSION` value must match that of the linux kernel tarball you downloaded (e.g., 2.6.9).

  If you are building a uni-processor kernel, then set the following:

  ```
  SMP             = 0
  ```

  Or, if you are building a multi-processor kernel (i.e., SMP), then set the following:

  ```
  SMP             = 1
  ```

- Build the kernel:

  ```
  # make rpm
  ```

- Copy the resulting RPMs into the current distribution:

  ```
  # cp ../../RPMS/<arch>/kernel*rpm /home/install/contrib/4.1/<arch>/RPMS/
  ```

  Where *<arch>* is i386, x86_64 or ia64.

- *This next step is for Rocks 4.0.0 only.*

  For Rocks version 4.0.0 you need to do the following:

  ```
  # cd /home/install/site-profiles/4.1/nodes
  # wget http://www.rocksclusters.org/ftp-site/beta/4.1.0/kernel.org/extend-compute.xml
  ```

• Rebuild the distribution:

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

• 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.

5.7. 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.

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/lan/arch/build/graphs/default/base-rsh.xml \
 /home/install/site-profiles/4.1/graphs/default/
```

Where *arch* is your architecture ("i386", "x86_64" or "ia64").

Now edit `/home/install/site-profiles/4.1/graphs/default/base-rsh.xml` and change the following:

```
        <!-- Uncomment to enable RSH on your cluster

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

        -->
```

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</to>
                <to>rsh</to>
```

```
          </edge>
```

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

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

Then, reinstall your compute nodes.

5.8. Customizing Ganglia Monitors

5.8.1. 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 XML node file called `replace-ganglia-client.xml`.

  Put the following contents in the new file:

  ```
  <?xml version="1.0" standalone="no"?>

  <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.

5.9. Adding a New Appliance Type to the Cluster

This procedure describes how to add a new appliance type to your cluster. This is useful when you want a subset of compute nodes to have specific behavior that is different from the rest of the compute nodes. For example, if you want all the nodes in cabinet 1 to be configured differently from the rest of the compute nodes.

Before you begin, you'll want to be comfortable with the Rocks XML framework that is used to produce a configuration graph. Details on this framework are found in the Reference Guide[3].

First, you'll need to create a new node XML file. This file will contain the configuration scripts and/or packages that will be applied to each of your appliances. Let's call it `my-compute.xml`. This file should be created in the directory `/home/install/site-profiles/4.1/nodes`. Below is the contents of the file:

```
<?xml version="1.0" standalone="no"?>

<kickstart>

<description>
My specialized compute node
</description>


<changelog>
</changelog>

<post>

<file name="/etc/motd" mode="append">
My Compute Appliance
</file>

</post>

</kickstart>
```

Now, we'll link the above file into the existing XML configuration graph. We'll simply point the above XML node to the existing `compute.xml` node. In object-oriented terms, we are inheriting all the functionality of the compute appliance and then extending it.

To link `my-compute.xml` to `compute.xml`, in the directory `/home/install/site-profiles/4.1/graphs/default`, create the file `my-appliance.xml` and have it contain:

```
<?xml version="1.0" standalone="no"?>

<graph>

<description>
</description>

<changelog>
</changelog>

<edge from="my-compute">
```

```
        <to>compute</to>
</edge>

<order gen="kgen" head="TAIL">
        <tail>my-compute</tail>
</order>

</graph>
```

To apply the changes above to the current distribution, execute:

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

Now we need to add an entry into the Rocks MySQL database. This is accomplished with `add-new-appliance`:

```
# add-new-appliance --appliance-name "My Compute" --xml-config-file-name my-compute
```

Now let's retarget an existing compute node. We'll use `insert-ethers` to accomplish this task. First, ask `insert-ethers` to replace `compute-0-0`:

```
# insert-ethers --replace compute-0-0
```

This displays the screen:
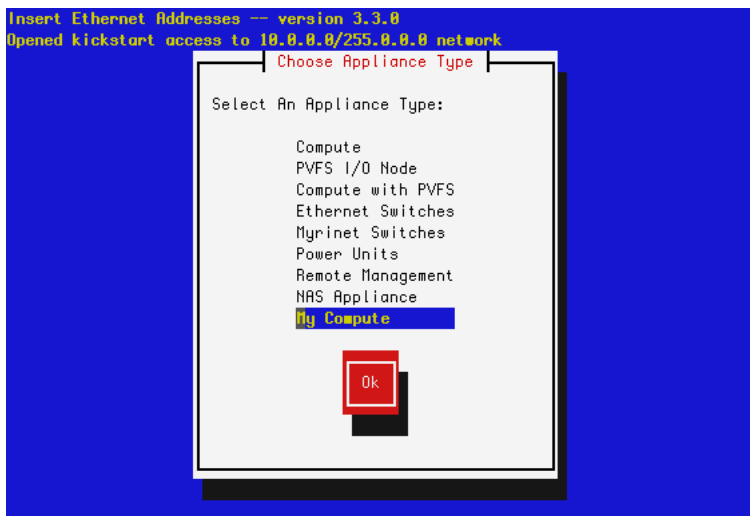
Select *My Compute* then hit *Ok*. This removes `compute-0-0` from the database and the next node that asks to be configured (that is, the next node that sends out a DHCP request) will be assigned the name `my-compute-0-0`. To see this in action, now instruct `compute-0-0` to reinstall itself:

```
# ssh-add
# shoot-node compute-0-0
```

Eventually, you'll see `insert-ethers` report that it discovered `my-compute-0-0`. After the node installs, it will be configured as a *my-appliance*. You can login to the node by executing:

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

Your custom appliance can be applied to any new node in your system by starting `insert-ethers` as instructed above, then by booting a new node in configuration mode (by forcing it to PXE boot or by booting the node with the a Rocks CD that contains the Kernel Roll).

5.10. Adding a Device Driver

This section describes how to add a device driver to the installation environment (*initrd.img*). This enables the installation environment to use the new driver as well as installing the device driver into the running environment (that is, after the node has installed).

This feature is enabled by the ddiskit[4] package created by John W. Linville at RedHat.

1. Set up a build environment:

   ```
   # cd /tmp
   # 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.

2. Now get the Rocks core development environment and the Kernel Roll development environment:

   ```
   # cvs -d:pserver:anonymous@cvs.rocksclusters.org:/home/cvs/CVSROOT checkout -r ROCKS_4_1 rocks-de
   # cvs -d:pserver:anonymous@cvs.rocksclusters.org:/home/cvs/CVSROOT checkout -r ROCKS_4_1_DD rocks
   ```

3. Go to the directory which holds the device driver code:

   ```
   # cd /tmp/rocks/src/roll/kernel/src/rocks-boot/enterprise/4/images/drivers
   ```

4. In this directory, you'll see some example drivers. Let's look at the *e1000* driver:

   ```
   # cd e1000
   ```

5. If you want to supply a new version of the driver, you'll have to download the e1000 source tarball and copy the *.c and *.h files from the tarball to this directory. Make sure all the *.c and *.h files are listed at the top of the *Makefile*:

   ```
   MODULES := e1000

   SOURCES := \
           e1000_ethtool.c \
           e1000_hw.c \
           e1000_main.c \
           e1000_param.c \
           kcompat.c \
           kcompat_ethtool.c

   HEADERS := \
           e1000.h \
           e1000_hw.h \
   ```

```
        e1000_osdep.h \
        kcompat.h
```

6. You'll need to make sure the proper PCI ids are in the file *pcitable*. For example, to test on one of our Dell SC1425's, we added the line:

```
0x8086  0x1076  "e1000" "Intel|82541GI/PI Gigabit Ethernet Controller (rev 05)"
```

7. Now we'll need to specify to the device driver building code that the e1000 driver should be built. To do this, edit the file *subdirs*:

```
# cd ..
# vi subdirs
```

8. Change the section from:

```
#
# put a list of all the driver directories that you'd like to build.
#
# for example, to build the 'e1000' driver, uncomment the line below:
#e1000
```

to:

```
#
# put a list of all the driver directories that you'd like to build.
#
# for example, to build the 'e1000' driver, uncomment the line below:
e1000
```

9. Build the *rocks-boot* package:

```
# cd /tmp/rocks/src/roll/kernel/src/rocks-boot
# make rpm
```

10. When this completes, copy the binary RPMs into a directory where the distribution building utility (*rocks-dist*) will find and include them:

```
# cp /tmp/rocks/src/roll/kernel/RPMS/x86_64/rocks-boot* \
 /home/install/contrib/4.1/x86_64/RPMS/
```

If you are building on an i386 system, change the above x86_64 references to i386.

11. Rebuild the distro:

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

12. Install the newly created *initrd.img* and its matching kernel *vmlinuz* so PXE booted nodes will get the new device drivers:

```
# cd /home/install
# rpm -Uvh --force rocks-dist/lan/x86_64/RedHat/RPMS/rocks-boot-4*.rpm
# cp /boot/kickstart/default/initrd.img /tftpboot/pxelinux/
# cp /boot/kickstart/default/vmlinuz /tftpboot/pxelinux/
```

13. Now PXE boot a node. This node will load your new driver and will install this driver into the running environment.

5.10.1. Adding a New Device Driver (That Isn't One of the Example Drivers)

If the name of your device driver you wish to add is not one of the example device drivers (e.g., ata_piix, cciss, e1000, sk98lin, or tg3), then you'll need to create a new directory and populate it with the appropriate files.

For example, let's say your new device driver is called *fa* and it is a network device driver.

```
# cd /tmp/rocks/src/roll/kernel/src/rocks-boot/enterprise/4/images/drivers
# mkdir fa
# cd fa
# cp ../e1000/modinfo .
# cp ../e1000/Makefile* .
# cp ../e1000/modules.dep .
# cp ../e1000/pcitable .
```

You'll need to edit *modinfo*, *modules.dep* and *pcitable* to match your driver. See ddiskit[5] for details on how to properly format the files.

Then you'll need to edit *Makefile* to ensure all the *.c and *.h files are listed. Also, if the driver requires special flags, make sure the are appended to the last line of *Makefile*. For example, to add the flag *-DFA_DEBUG*, change the line from:

```
EXTRA_CFLAGS += -I$(PWD)
```

to:

```
EXTRA_CFLAGS += -I$(PWD) -DFA_DEBUG
```

The rest of the build process follows the same procedure as above starting at step 7 where you'll have to add *fa* to the file *subdirs*.

Notes

1. http://www.rocksclusters.org/rocks-documentation/reference-guide/4.1/database.html

2. https://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/sysadmin-guide/s1-kickstart2-options.html

3. http://www.rocksclusters.org/rocks-documentation/reference-guide/4.1/

4. http://people.redhat.com/linville/ddiskit/

5. http://people.redhat.com/linville/ddiskit/INSTALL

Chapter 6. Downloads

6.1. ISO images and RPMS

Rocks software can be downloaded here[1].

6.2. CVS Access to the Rocks Source Tree

Anonymous read-only access is provided in two forms:

- Browsing[2] using ViewCVS.
- Using the CVS `pserver`.

6.2.1. Read-Only Access to CVS

To checkout the source, 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 the source tree:

```
$ cvs -d:pserver:anonymous@cvs.rocksclusters.org:/home/cvs/CVSROOT checkout -r ROCKS_4_1 rocks
```

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://www.rocksclusters.org/wordpress/?page_id=3

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

Chapter 7. Frequently Asked Questions

7.1. Installation

1. Insert-ethers never sees new compute nodes. I also don't see any DHCP messages from compute nodes on the frontend. What is wrong?

Try bypassing the network switch connecting your nodes to the frontend. The swich may be configured to squash broadcast messages from unknown IP addresses, which drops DHCP messages from nodes. To verify your switch is indeed the problem:

1. Connect a crossover cable (or a normal cable if you use Gigabit Ethernet) between a single compute node and the frontend's "eth0" interface.

2. Install the compute node normally (install compute nodes). You should see the DHCP messages from the node at the frontend.

2. While trying to bring up a compute node, I boot it from the Rocks Boot 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
# ./sbin/kickstart.cgi --client="compute-0-0"
```

This should return a kickstart file.

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

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

3. 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 Boot Roll 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.

4. 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.

5. 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 Production Release* (as of this writing, this is version 5.4.0).

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-5.4.0-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 `system-config-display` 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. 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).

5. 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 Boot CD. If you select the boot order to place PXE before hard disk to node will repeatedly re-install itself.

6. 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.

7. 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/lan/arch/build/nodes/auto-kickstart.xml \
site-profiles/4.1/nodes/replace-auto-kickstart.xml
```

  Where *arch* is "i386", "x86_64" or "ia64".

- Edit the file `site-profiles/4.1/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. 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.

2. 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.

3. How do I use user accounts from an external NIS server on my cluster?

While there is no certain method to do this correctly, if necessary we recommend you use "ypcat" to periodically gather external NIS user accounts on the frontend, and let the default 411 system distribute the information inside the cluster.

The following cron script will collect NIS information from your external network onto the frontend. The login files created here will be automatically distributed to cluster nodes via 411. This code courtesy of Chris Dwan at the University of Minnesota.

```
(in /etc/cron.hourly/get-NIS on frontend)

#!/bin/sh
ypcat -k auto.master > /etc/auto.master
ypcat -k auto.home   > /etc/auto.home
ypcat -k auto.net    > /etc/auto.net
ypcat -k auto.web    > /etc/auto.web

ypcat passwd        > /etc/passwd.nis
cat   /etc/passwd.local /etc/passwd.nis > /etc/passwd.combined
cp    /etc/passwd.combined /etc/passwd

ypcat group         > /etc/group.nis
cat   /etc/group.local /etc/group.nis > /etc/group.combined
cp    /etc/group.combined /etc/group
```

There is no way to insure that UIDs GIDs from NIS will not conflict with those already present in the cluster. You must always be careful that such collisions do not occur, as unpredicatble and undefined behavior will result.

4. How can I use a different DNS server for my compute nodes?

While rocks enforces that the compute nodes use the DNS server on the frontend, you may have the compute nodes use specific DNS servers from your organization to resolve names.

Instruct the frontend DNS server to use your desired servers as designated forwarders. This will "expand" the rocks server's knowledge to include names served by your DNS servers. (Submitted by Matt Wise)

1. Edit "/etc/named.conf" on your frontend.

   In the "options" paragraph, add the following lines:

   ```
   forward first;
   forwarders { (local named server); };
   ```

2. With an example forwarder 198.202.75.26, the file would look like this:

   ```
   //
   // named.conf for Red Hat caching-nameserver
   //

   options {
    directory "/var/named";
    dump-file "/var/named/data/cache_dump.db";
          statistics-file "/var/named/data/named_stats.txt";
    /*
     * If there is a firewall between you and nameservers you want
     * to talk to, you might need to uncomment the query-source
     * directive below.  Previous versions of BIND always asked
     * questions using port 53, but BIND 8.1 uses an unprivileged
     * port by default.
     */
    // query-source address * port 53;
    forward first;
    forwarders { 198.202.75.26; };
   };
   ```

3. Restart the named service:

   ```
   service named restart
   ```

Your compute nodes will be able to resolve names served by your nameserver.

5. How do I add a user account on a NAS (Network Attached Storage) appliance?

Assume your NAS node is called "bigbird" and users directories are exported from "/export/users". To add the user "dave" with home area as bigbird:/export/users/dave

Use "useradd" to make an account for dave.

```
# useradd  --netserver bigbird -d /export/users/dave dave
```

In the example above, the directory bigbird:/export/users must be both mountable and writable by root on the frontend. If the directory is not writable by root then the users directory must be created by other means. See the "-M" option to useradd.

You can use the -d flag to use a different directory than the example above

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

411 on Scalable Password Service , Federico D Sacerdoti, Mason J. Katz, and Philip M. Papadopoulos, July 2005, IEEE High Performance Distributed Computing Conference, North Carolina. , (PDF)[1] .

Rolls: Modifying a Standard System Installer to Support User-Customizable Cluster Frontend Appliances , Greg Bruno, Mason J. Katz, Federico D. Sacerdoti, and Philip M. Papadopoulos, September 2004, IEEE International Conference on Cluster Computing, San Diego , (PDF)[2] .

Grid Systems Deployment & Management Using Rocks , Federico D. Sacerdoti, Sandeep Chandra, and Karan Bhatia, September 2004, IEEE International Conference on Cluster Computing, San Diego , (PDF)[3] .

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)[4] .

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)[5] .

Leveraging Standard Core Technologies to Programmatically Build Linux Cluster Appliances , Mason J. Katz, Philip M. Papadopoulos, and Greg Bruno, April 2002 , CLUSTER 2002:[6] IEEE International Conference on Cluster Computing , (PDF)[7] .

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[8] Special Issue: Cluster 2001 , (PDF)[9] (PostScript)[10] .

NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters , Philip M. Papadopoulos, Mason J. Katz, and Greg Bruno, October 2001 , Cluster 2001[11] , (PDF)[12] (PostScript)[13] .

Talks

Rocks Basics, SUN HPC Consortium. , November 2004 , (Powerpoint)[14] .

High Performance Linux Clusters, Guru Session, Usenix. , June 2004 , (Powerpoint)[15] .

NPACI All Hands Meeting, Rocks v2.3.2 Tutorial Session , March 2003 , (PDF)[16] (Powerpoint)[17] .

Managing Configuration of Computing Clusters with Kickstart and XML , March 2002 , (Powerpoint)[18] .

NPACI All Hands Meeting, Rocks v2.2 Tutorial Session , March 2002 , (PDF)[19] (Powerpoint)[20] .

NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters , IEEE Cluster 2001, Newport Beach, CA. , October 2001 , (PDF)[21] .

Introduction to the NPACI Rocks Clustering Toolkit: Building Manageable COTS Clusters , NPACI All Hands Meeting, Rocks v2.0 Tutorial Session. , February 2001 , (Powerpoint)[22] .

Press

Itanium Gets Supercomputing Software , 10 April 2003, C|Net ,   (HTML) [23] .

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) [24] .

Linux, Intel and Dell - Supercomputing at a Major University : Intel e-Business Center Case Study , August 2002, Intel Corporation ,   (PDF) [25] .

NPACI Rocks Simplifies Deployment of Intel Itanium Clusters , 12 June 2002, NPACI & SDSC Online,   (HTML)[26] .

The Beowulf State of Mind, 01 May 2002, Glen Otero, Linux Journal, (HTML)[28] .

Linux on Big Iron, 25 March 2002, eWeek,   (HTML)[29] .

Keck-funded SDSC Satellite Site Proves a Boon to Campus Scientists , 6 March 2002, NPACI & SDSC Online,   (HTML)[30] .

Cal-(IT)2, IBM, SDSC & Scripps Inst. of Oceanography Announce COMPAS, December 2001, Supercomputing Online,   (HTML)[31] .

Texas Advanced Computing Center Adds Two New Clusters, December 2001, IT @ UT,   (HTML)[32] .

Cluster-Management Software Developers Preparing for the TeraGrid, July-September, 2001, NPACI Envision, (HTML)[33] .

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)[34] .

High Performace Computing for Proliant, 22 June 2001, Compaq Corporation Web Site,   (HTML)[35] .

NPACI Rocks Open-source Toolkit Improves Speed and Ease of Use in Cluster Configuration, 21 March 2001, NPACI & SDSC Online,   (HTML)[36] .

NPACI Releases Rocks Open-source Toolkit for Installing and Managing High-performance Clusters, 1 November 2000, NPACI & SDSC Online,   (HTML)[37] .

Research

A Cache-Friendly Liquid Load Balancer , Federico D. Sacerdoti, Masters Thesis: June 2002 , UCSD Technical Report , (PDF)[38] .

Notes

1. papers/hpdc2005-411.pdf

2. papers/cluster2004-roll.pdf

3. papers/cluster2004-rocks-wan.pdf

4. papers/cluster2003-ganglia.pdf

5. papers/lyon-2002.pdf

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

7. papers/clusters2002-rocks.pdf

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

9. papers/concomp2001-rocks.pdf

10. papers/concomp2001-rocks.ps

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

12. papers/clusters2001-rocks.pdf

13. papers/clusters2001-rocks.ps

14. talks/sun-hpc-2004.ppt

15. talks/guru.ppt

16. talks/npaci-ahm-2003.pdf

17. talks/npaci-ahm-2003.ppt

18. talks/Rocks-MSR.ppt

19. talks/npaci-ahm-2002.pdf

20. talks/npaci-ahm-2002.ppt

21. talks/cluster2001.pdf

22. talks/npaci-ahm-2001.ppt

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

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

25. papers/NUSCaseStudy.pdf

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

27. http://linuxprophet.com/

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

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

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

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

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

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

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

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

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

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

38. 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

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