NPACI Rocks Tutorial

NPACI All Hands Meeting
March 6, 2002

Outline

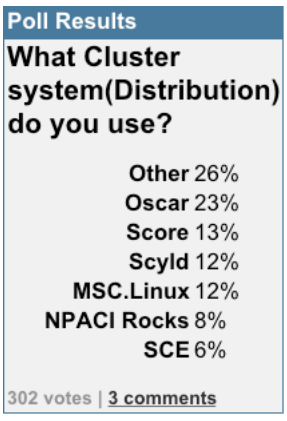
- Build a Rocks Cluster
  - **Start right away**
  - **Check in occasionally throughout this talk**
- Introduction
- Hardware Components
- Software Components
- Future Work
- Open Lab (Customization)

Who is NPACI Rocks?

- Cluster Computing Group at SDSC
- UC Berkeley Millennium Project
  - **Provide Ganglia Support**
- Linux Competency Centre in SCS Enterprise Systems Pte Ltd in Singapore
  - **Provide PVFS Support**
  - **Working on user documentation for Rocks 2.2**

Cluster distributions

- http://clusters.top500.org/
- We were honored to be seen as experts in the field
- We had a blast voting for Rocks and Oscar (this isn't a real poll)
- We encourage people to try these other distributions
- Why are 26% rolling there own clusters?
  - **This is the real message of the poll**



What sets us apart

- Fully automated cluster deployment
  1. **Get and burn ISO CD image from Rocks.npaci.edu**
  2. **Fill-out form to build initial kickstart file for your first front-end machine**
  3. **Kickstart "naked" frontend with CD and kickstart file**
  4. **Reboot frontend machine**
  5. **Integrate compute nodes with "Insert Ethers"**
  6. **Ready to go!**
- Complete out of the box solution with rational default settings

Who is Using It?

- Growing (and partial) list of users that we know about:
  - **SDSC, SIO, UCSD (8 Clusters, including CMS (GriPhyN) prototype)**
  - **Caltech**
  - **Burnham Cancer Institute**
  - **PNNL (several clusters, small, medium, large)**
  - **University of Texas**
  - **University of North Texas**
  - **Northwestern University**
  - **University of Hong Kong**
  - **Compaq (Working relationship with their Intel Standard Servers Group)**
  - **Singapore Bioinformatics Institute**
  - **Myricom (Their internal development cluster)**

Motivation

- Care and feeding for a system isn't fun.
- Enable non-cluster experts to run clusters
- Essential to track software updates
  - **Open source moves fast!**
  - **On the order of 3 updates a week for Red Hat**
- Essential to track Red Hat releases
  - **Feature rot**
  - **Unplugged security holes**
- Run on heterogeneous, standard high-volume components

Philosophy

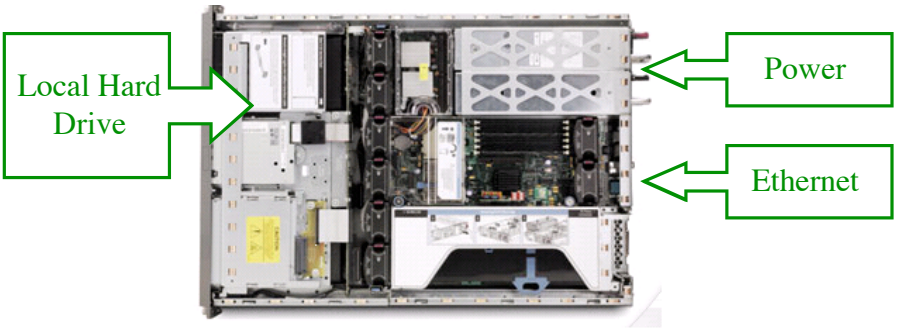
- All nodes are 100% automatically installed
  - **Zero "hand" configuration**
  - **Scales very well**
- NPACI Rocks is an entire cluster-aware distribution
  - **Included packages**
    - Full Red Hat release
    - De-facto standard cluster packages (MPI, PBS/Maui, etc.)
    - NPACI Rocks packages

More Philosophy

- Use installation as common mechanism to manage cluster
  - **Install when:**
    - Initial bring-up
    - Replacing a dead node
    - Adding new nodes
- Also use installation to keep software consistent
  - **If you catch yourself wondering if a node's software is up-to-date, reinstall!**
    - In 10 minutes, all doubt is erased.

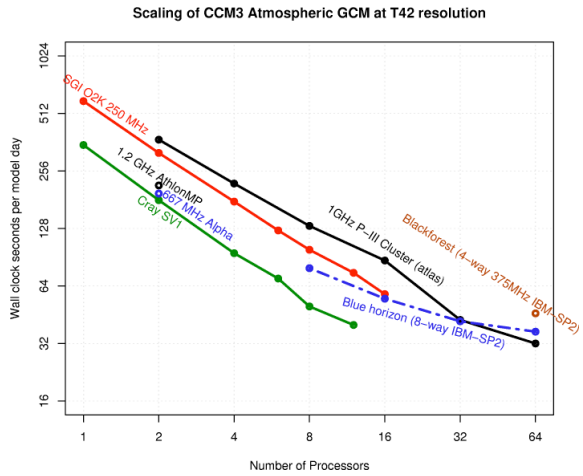
Hardware

Minimum Components



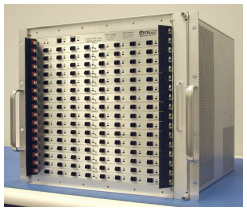
x86 server

Why x86 clusters?



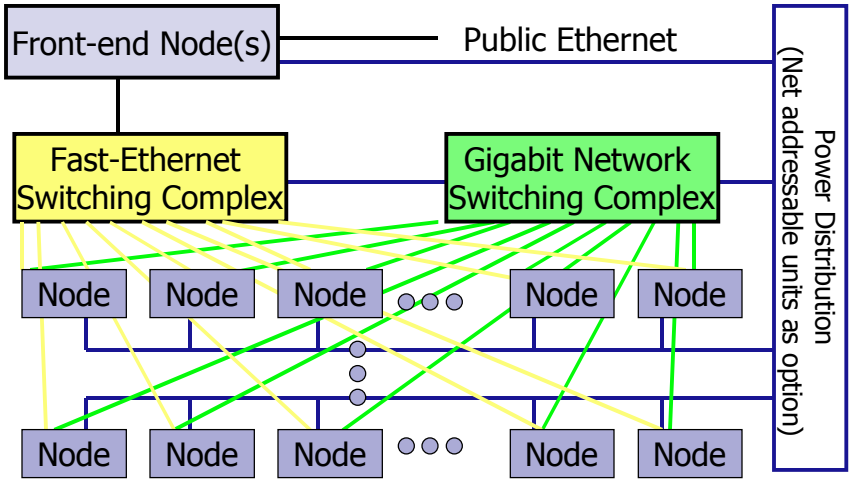
Source: Dave Pierce, SIO

Optional Components



- Myrinet high-performance network
- Network-addressable power distribution unit
- Evil keyboard/video/mouse network not required
  - **In fact, we believe it's not desirable – just another network to manage.**

Basic Architecture

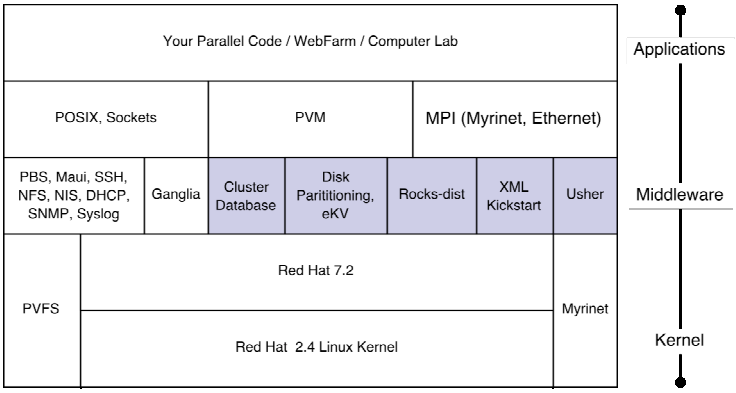


Meteor Cluster at SDSC

- Rocks v2.2
- 2 Frontends
- 4 NFS Servers
- 100 nodes
  - **Compaq**
    - 800, 933, IA-64
    - SCSI, IDA
  - **IBM**
    - 733, 1000
    - SCSI
- 50 GB RAM
- Ethernet
  - **For management**
- Myrinet 2000

Software

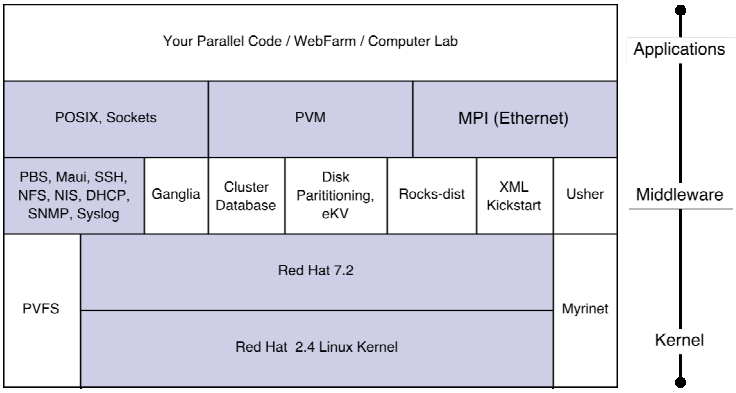
Major Components



Software

Standard Beowulf
(and then some)

Major Components



Red Hat

- Stock Red Hat 7.2 with updates (x86 and ia64)
- Linux 2.4 Kernel
- No support for other distributions
  - **We believe Red Hat is the market leader for Linux**
    - In the US
    - And becoming so in Europe
  - **Automated install (kickstart) is a requirement**
  - **Very good hardware detection**
  - **A Cluster is not a general purpose computer**
    - "But I'd rather run debian" doesn't mean anything

Parallel Application APIs

- POSIX, Sockets
  - **Simple parallel jobs**
  - **First level of parallel code**
- MPI, PVM
  - **Message passing**
  - **Advanced programming model**
  - **High performance parallelism**

MPI

- Message Passing Interface v1.1
  - **Standard for high performance message passing on parallel machines**
  - **http://www-unix.mcs.anl.gov/mpi/**
- Supports
  - **GNU C, Fortran 77**
  - **Intel C, Fortran 77, Fortran 90**
  - **Portland Group C, C++, Fortran 77, Fortran 90**
    - Requires site license

PVM

- Parallel Virtual Machines v3.4.3
  - **Message passing interface for heterogeneous architectures**
    - Supports over 60 variants of UNIX
    - Supports Windows NT
  - **Resource control and meta computing**
  - **Fault tolerance**
  - **http://www.csm.ornl.gov/pvm/**

Portable Batch System

- Three standard components to PBS
  - **MOM**
    - Daemon on every node
    - Used for job launching and health reporting
  - **Server**
    - On the frontend only
    - Queue definition, and aggregation of node information
  - **Scheduler**
    - Policies for what job to run out of which queue at what time
- We added a forth
  - **Configuration**
    - Get cluster environment from our SQL database
    - Get list of nodes from Ganglia

PBS RPM Packaging

- Repackaged PBS
  - **Added "chkconfig" support to start up scripts**
  - **Four packages**
    - pbs
    - pbs-mom
    - pbs-config-sql
      - **Python code to generate database report**
    - pbs-common
- Rocks 2.2 automatically defines a default queue with all compute nodes participating
- http://pbs.mrj.com is a good starting point for pbs

PBS Server Defaults

- Startup script: "/etc/rc.d/init.d/pbs-server start"
- /usr/apps/pbs/pbs.default
  - **Sourced every time PBS is started**

```
#
# Create and define queue default
#
create queue default
set queue default queue_type = Execution
set queue default resources_max.nodect = 256
set queue default resources_max.nodes = 999
set queue default resources_min.nodect = 1
set queue default resources_default.neednodes = 1
set queue default resources_default.nodect = 1
set queue default resources_default.nodes = 1
set queue default resources_default.walltime = 1000:00:00
set queue default enabled = True
set queue default started = True
#
# Set server attributes (assume maui scheduler will be installed)
#
set server managers = maui@frontend-0
set server operators = maui@frontend-0
set server default_queue = default
set server log_events = 511
set server mail_from = adm
set server scheduler_iteration = 600
set server scheduling = false
```

Maui Scheduler

- PBS using Maui 3.0 as the scheduler
  - **De facto standard HPC scheduling**
  - **http://supercluster.org/projects/maui**
- We are also looking into
  - **Sun's Grid Engine**
  - **SDSC's Catalina Scheduler**
  - **Maui Scheduler Molokini Edition**

NFS

- User account are served over NFS
  - **Works for small clusters (<= 128 nodes)**
  - **Will not work for large clusters (>1024 nodes)**
  - **NAS is better than Linux**
    - Rocks uses the Frontend machine to server NFS
    - We have deployed NAS on several clusters
- Applications are not served over NFS
  - **/usr/local/ does not exist**
  - **All software is installed locally from RPM**

Open SSH

- Replaces Telnet, Rsh
  - **Cryptographically strong authentication and encryption**
  - **Forwards X11 connections (no more $DISPLAY)**
- Rocks uses SSH
  - **Mpi-launch**
  - **Cluster-fork**
- Ssh-agent
  - **Manager for SSH keys**
  - **ssh-agent $SHELL**

NIS & DHCP

- Network Information System
  - **Manages user accounts and host information**
  - **Alternative to copying files around**
    - /etc/hosts
    - /etc/passwd
  - **Scalable**
    - Supports slave servers
    - Clients attach to closest server and load balance themselves
- Dynamic Host Configuration Protocol
  - **Manages network information**
  - **Alternative to static IP address**
  - **Foundation of insert-ethers**
- Critical to keep dynamic state dynamic

SNMP

- Enabled on all compute nodes
- Great for point-to-point use
  - **Good for high detail on a single end-point**
  - **Does not scale to full cluster wide use**
- Supports Linux MIB
  - **Uptime, Load, Network statistics**
  - **Install Software**
  - **Running Processes**

snmp-status

- snmp-status compute-0-0

```
PID        TIME       MEM        PROCESS
1          424        264        init
2          0          0          keventd
3          3          0          ksoftirqd_CPU0
4          4          0          ksoftirqd_CPU1
5          17212      0          kswapd
6          0          0          kreclaimd
7          1          0          bdflush
8          5          0          kupdated
9          0          0          mdrecoveryd
18         1695       0          kjournald
137        0          0          kjournald
5769       0          3856       httpd
11685      0          144        pvfsd
12083      1          44         dhcpcd
11685      0          144        pvfsd
12083      1          44         dhcpcd
12224      2536       432        syslogd
12229      11         268        klogd
```

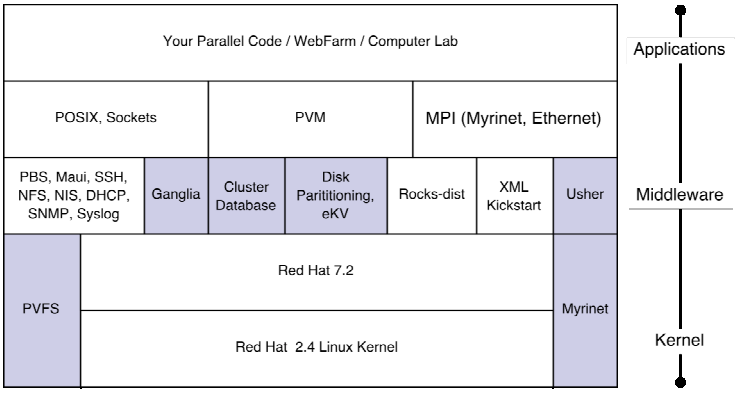
Syslog

- Native UNIX system event logger
  - **Logs events to local dist**
    - /var/log/message
    - Rotates logs daily, eventually historic data is lost
  - **Forwards all message to the frontend**
- Scalable
  - **Can add additional loghosts**
  - **Can throttle verbosity of loggers**
- Uses
  - **Predicting hardware and software failures**
  - **Post Mortem on crashed nodes**
  - **Debugging System startup**

Software

Beyond Beowulf

Major Components



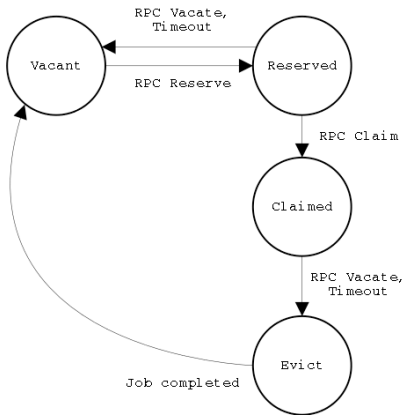
Optional Drivers

- PVFS
  - **Parallel Virtual File System**
  - **Kernel module built for all nodes**
  - **Initial support (full support in future version of Rocks)**
  - **User must decide to enable**
- Myrinet
  - **High Speed and Low Latency Interconnect**
  - **GM/MPI for user Applications**
  - **Kernel module built for all nodes with Myrinet cards**
  - **Usher daemon for dynamic port management**

Usher

- Distributed resource manager for GM ports
  - **RPC based**
  - **Reservation system**
  - **Simple state machine with timeouts**
  - **Allows multiple GM jobs per node**
  - **Transparent**
    - Integrated into “mpi-launch”, and “rexec” for job launching
- Stock GM
  - **All jobs run on port 3**
  - **No strategy for managing port allocations**

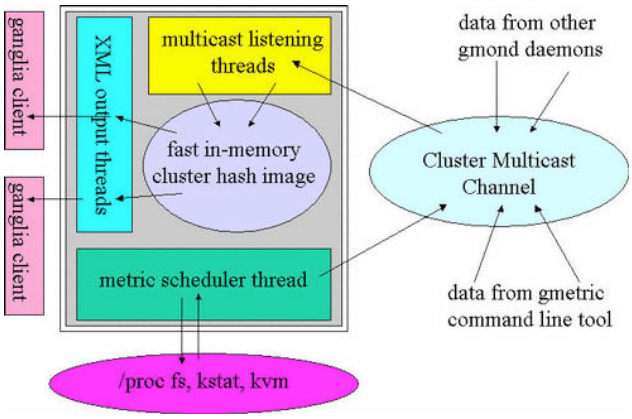
Usher State Machine



Ganglia

- Scalable cluster monitoring system
  - **Based on ip multi-cast**
  - **Matt Massie, et al from UCB**
  - **http://ganglia.sourceforge.net**
- Gmon daemon on every node
  - **Multicasts system state**
  - **Listens to other daemons**
  - **All data is represented in XML**
- Ganglia command line
  - **Python code to parse XML to English**
- Gmetric
  - **Extends Ganglia**
  - **Command line to multicast single metrics**

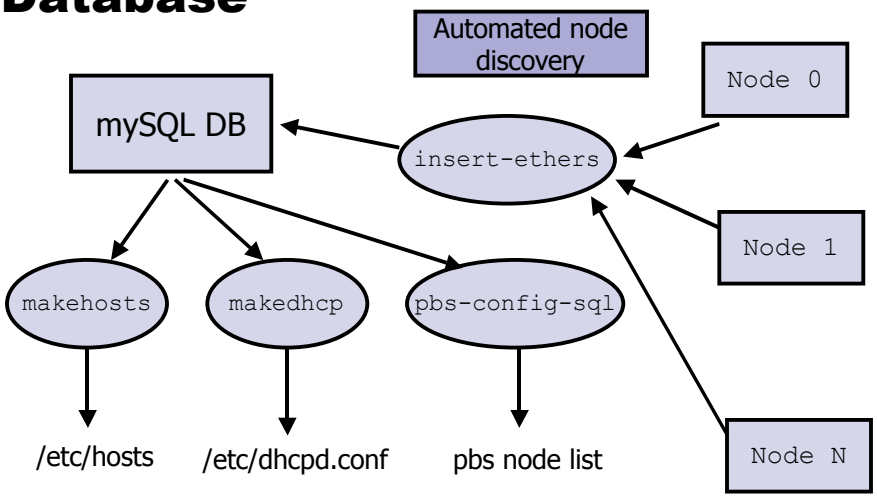
Ganglia



Cluster Database

- Goal: derive all cluster configuration via queries to an SQL database
  - **Rationale: SQL is an expressive, well-understood interface to data. Allows us to write scripts that deal with the evolving format of configuration files.**
- Goal: write utilities that dynamically extract information from the database
  - **insert-ethers**
  - **makehosts, makedhcp**
  - **Gmon + SQL = ganglia command line client**

Configuration Derived from Database



Key Tables - Nodes

ID	MAC	Name	Membership	Hardware	Rack	Rank	IP	C
1		frontend-0	1	0	0	0	10.1.1.1	
2	00:30:c1:d8:59:00	network-1-0	6	0	1	0	10.255.255.254	
3	00:01:e7:1a:be:00	network-0-0	6	0	0	0	10.255.255.253	
4	00:30:c1:d8:ac:80	network-3-0	6	0	3	0	10.255.255.252	
5	00:50:8b:a5:4d:b1	nfs-0-0	12	0	0	0	10.255.255.251	
6	00:50:8b:c5:c3:72	nfs-0-1	12	0	0	1	10.255.255.250	
7	00:50:8b:a5:57:ff	nfs-1-0	12	0	1	0	10.255.255.249	
8	00:50:8b:a5:4c:f4	nfs-1-1	12	0	1	1	10.255.255.248	
9	00:50:8b:e0:3a:a7	compute-0-0	2	0	0	0	10.255.255.247	
10	00:50:8b:e0:44:5e	compute-0-1	2	0	0	1	10.255.255.246	
11	00:50:8b:e0:40:95	compute-0-2	2	0	0	2	10.255.255.245	
12	00:50:8b:e0:40:93	compute-0-3	2	0	0	3	10.255.255.244	
13	00:50:8b:e0:42:df	compute-0-4	2	0	0	4	10.255.255.243	

Key Tables - Nodes

- Nodes table associates MAC addresses to IP addresses and physical location
- Nodes table is populated with ‘insert-ethers’

Key Tables - Memberships

ID	Name	Appliance	Distribution	Compute
1	Frontend	1	1	no
2	Compute	2	1	yes
3	PVFS I/O Node	3	1	no
4	Compute with PVFS	4	1	yes
5	Laptop	5	1	no
6	Ethernet Switches	6		no
7	Myrinet Switches	6		no
8	Power Units	7		no
9	Remote Management	8		no
10	DTF Compute	9	1	yes
11	Web Portal	10	1	no
12	NFS Server	11	1	no

- Defines “appliance” types

Key Tables - Appliances

ID	Name	ShortName	Graph	Node
1	frontend	f	default	frontend
2	compute	c	default	compute
3	pvfs	pv	default	pvfs-io
4	comp-pvfs	cp	default	compute-pvfs
5	laptop		default	laptop
6	network	n		
7	power	p		
8	manager			
9	dtf	d	default	dtf-compute
10	portal	pl	default	portal
11	nfs	n	default	nfs

- Used to associate graph “starting points” for different appliances

Key Tables – App_Globals

ID	Membership	Service	Component	Value
1	0	Kickstart	PublicNTPHost	ntp.ucsd.edu
2	0	Kickstart	ZeroMBR	yes
3	0	Kickstart	PrivateKickstartCGI	kickstart.cgi
4	0	Kickstart	PublicNetmask	255.255.255.0
5	0	Kickstart	PublicNetwork	192.31.21.0
6	0	Kickstart	PrivateNISMaster	frontend-0
7	0	Kickstart	PrivateHostname	frontend-0
8	0	Kickstart	PrivateIPForwarding	yes
9	0	Kickstart	PrivateGateway	10.1.1.1
10	0	Kickstart	PublicKickstartBasedir	install
11	0	Kickstart	Lang	en_US

- Used to configure DHCP and to customize "appliance" kickstart files

Table Relationships

ID	MAC	Name	Membership	Hardware	Rack	Rank	IP	C
1		frontend-0	1	0	0	0	10.1.1.1	
2	00:30:c1:d8:59:00	network-1-0	6	0	1	0	10.255.255.254	
3	00:01:e7:1a:be:00	network-0-0	6	0	0	0	10.255.255.253	
4	00:30:c1:d8:ac:80	network-3-0	6					
5	00:50:8b:a5:4d:b1	nfs-0-0	12					
6	00:50:8b:c5:c3:72	nfs-0-1	12					
7	00:50:8b:a5:57:ff	nfs-1-0	12					
8	00:50:8b:a5:4c:f4	nfs-1-1	12					
9	00:50:8b:e0:3a:a7	compute-0-0	2					
10	00:50:8b:e0:44:5e	compute-0-1	2					
11	00:50:8b:e0:40:95	compute-0-2	2					
12	00:50:8b:e0:40:93	compute-0-3	2					
13	00:50:8b:e0:42:df	compute-0-4	2					

Nodes Table

ID	Name	Appliance	Distribution	Compute
1	Frontend	1	1	no
2	Compute	2	1	yes
3	PVFS I/O Node	3	1	no
4	Compute with PVFS	4	1	yes
5	Laptop	5	1	no
6	Ethernet Switches	6		no
7	Myrinet Switches	6		no
8	Power Units	7		no
9	Remote Management	8		no
10	DTF Compute	9	1	yes
11	Web Portal	10	1	no
12	NFS Server	11	1	no

Memberships Table

Table Relationships

ID	Name	Appliance	Distribution	Compute
1	Frontend	1	1	no
2	Compute	2	1	yes
3	PVFS I/O Node	3		no
4	Compute with PVFS	4		yes
5	Laptop	5		no
6	Ethernet Switches	6		no
7	Myrinet Switches	6		no
8	Power Units	7		no
9	Remote Management	8		no
10	DTF Compute	9		yes
11	Web Portal	10		no
12	NFS Server	11	1	no

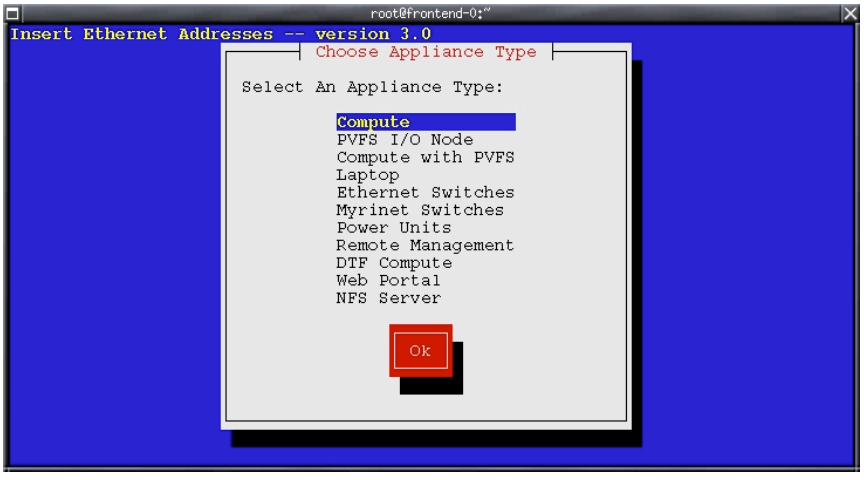
Memberships Table

ID	Name	ShortName	Graph	Node
1	frontend	f	default	frontend
2	compute	c	default	compute
3	pvfs	pv	default	pvfs-io
4	comp-pvfs	cp	default	compute-pvfs
5	laptop		default	laptop
6	network	n		
7	power	p		
8	manager			
9	dtf	d	default	dtf-compute
10	portal	pl	default	portal
11	nfs	n	default	nfs

Appliances Table

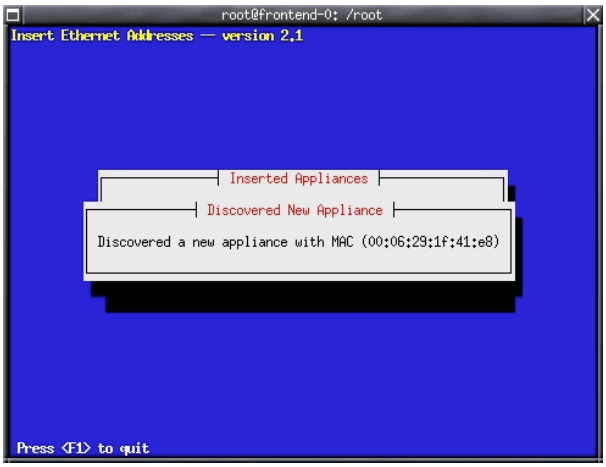
Example – Insert-ethers

- Insert-ethers gets ‘appliance list’ from database



Example – Insert-ethers

- Then it ‘inserts’ discovered appliances into nodes table



More Examples

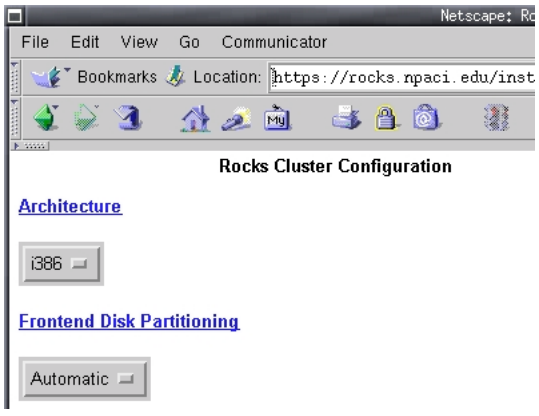
- The command line tool for ganglia reads the nodes list to determine all the compute nodes in the system.
- PBS ‘nodes’ file is dynamically built by extracting node names from the nodes table.
- Host entries in dhcpd.conf is built by extracting the names of the nodes from the nodes list
  - **The “global” configuration is built by extracting rows from the “app_globals” table**

Disk Partitioning and eKV

- Enhancements to standard Red Hat Kickstart
- Disk Partitioning
  - **Automatic or Manual for frontend**
  - **"Greedy" partitioning for compute nodes**
- eKV
  - **Ethernet Keyboard and Video**
  - **Watch and Interact with Kickstart over Ethernet**
    - No serial port concentrators
    - No lights out management card

Frontend Partitioning

- Setup from the configuration web page
- Two choices
  - **Automatic**
  - **Manual**

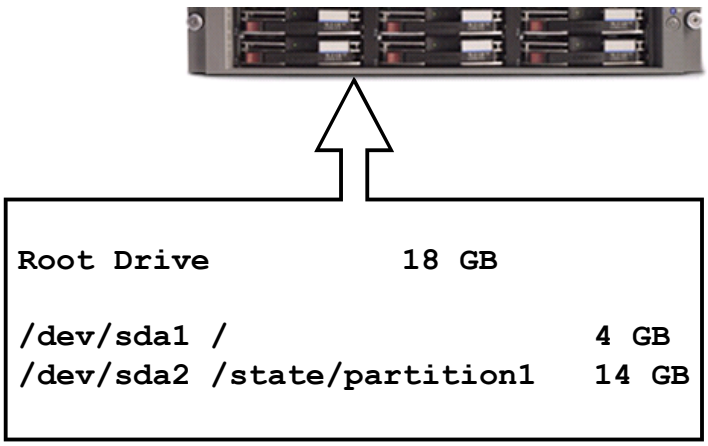


- Automatic
  - 4 GB root partition
- Manual
  - **You're on your own!**

Compute Node Partitioning

- Creates 4 GB root partition on first drive
  - **This partition is volatile, that is, when the node is reinstalled, this partition is reformatted**
- Remainder of first drive is put into a partition called "/state/partition1"
- For each remaining drives, one partition is created per drive and named "/state/partition2", "/state/partition3", etc.
- All partitions labeled "/state/partition[n]" are not reformatted on reboots.

Example



Example

```
Second Drive    36 GB

/dev/sdb1 /state/partition2    36 GB
```

Example

```
Third Drive     18 GB

/dev/sdc1 /state/partition3    18 GB
```

Example

```
Fourth Drive    72 GB

/dev/sdd1 /state/partition4    72 GB
```

Example

```
Fifth Drive     36 GB

/dev/sde1 /state/partition5    36 GB
```

Example

```
Sixth Drive     181 GB

/dev/sdf1 /state/partition6    181 GB
```

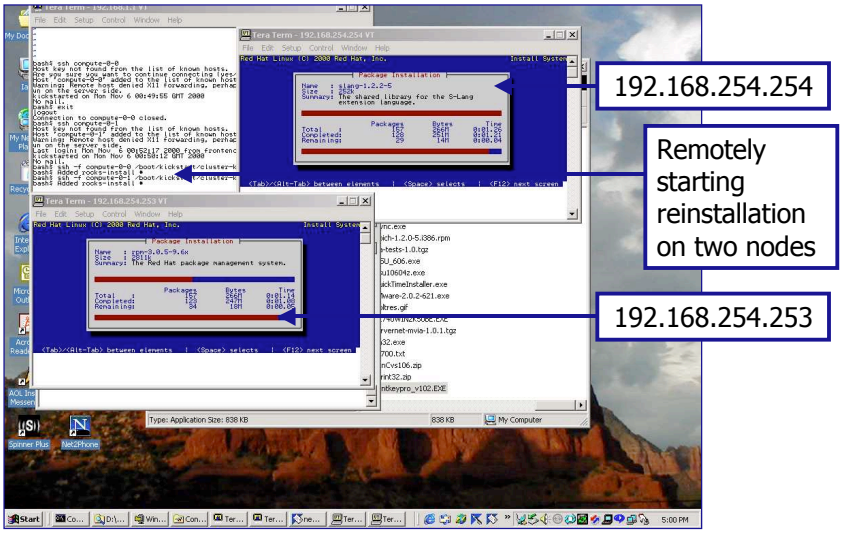
Example

Device	Name	Size
/dev/sda1	/	4 GB
/dev/sda2	/state/partition1	14 GB
/dev/sdb1	/state/partition2	36 GB
/dev/sdc1	/state/partition3	18 GB
/dev/sdd1	/state/partition4	72 GB
/dev/sde1	/state/partition5	36 GB
/dev/sdf1	/state/partition6	181 GB

eKV

- Remotely Interact with Installation
  - **Initial kickstart**
  - **Re-Installation**
- Shoot-node
  - **Reinstall OS and brings up eKV**
- eKV
  - **telnet compute-0-0 8000**
  - **Only on during installation**

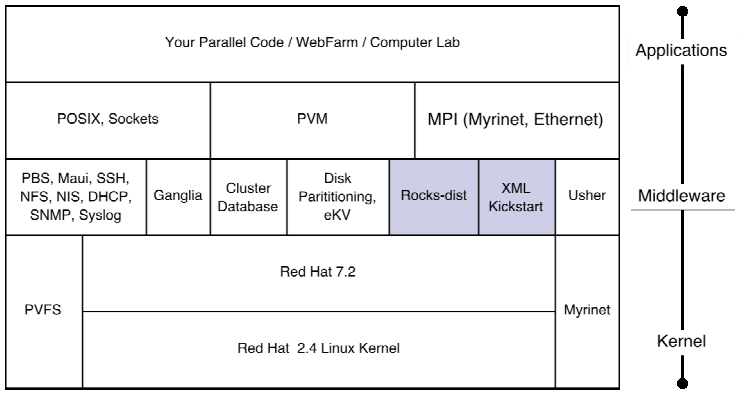
eKV



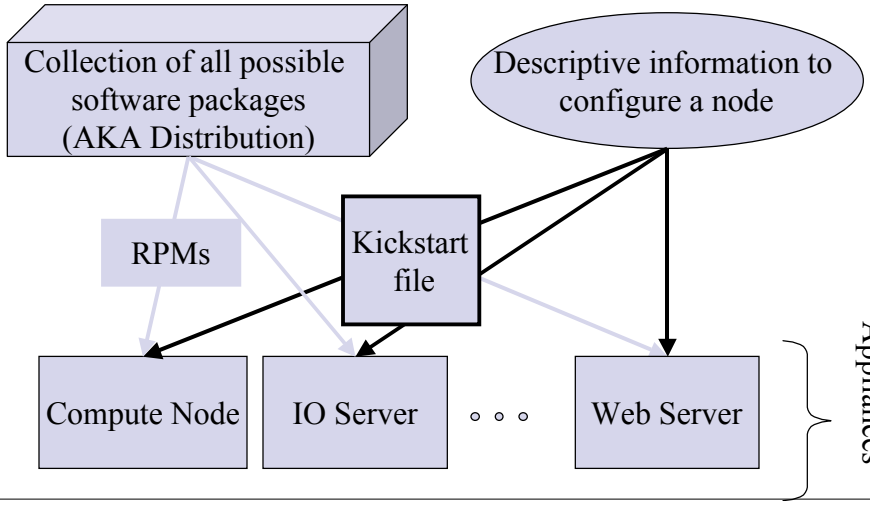
Software

Distribution Building and Installation

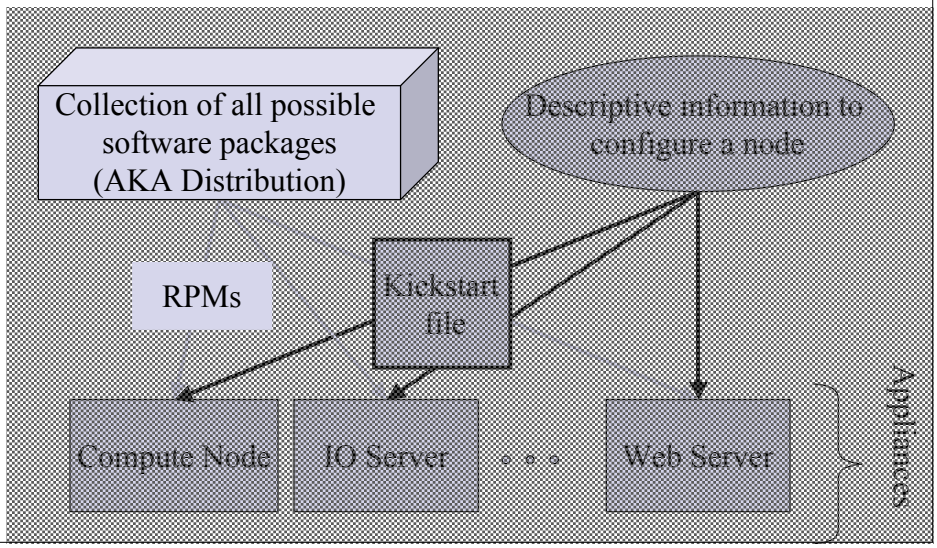
Major Components



Software Installation



Software Repository



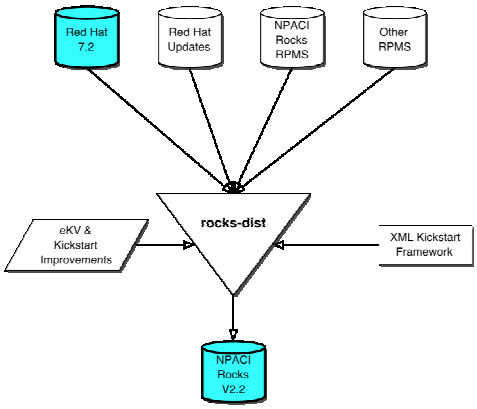
Rocks-dist

- Distribution builder
  - **Rocks**
  - **Red Hat**
  - **Same thing**
- Version Manager
  - **Resolves software updates**
  - **Default to the most recent software**
  - **Can force package versions as needed**
- Distribution versioning
  - **Allows multiple distributions at once**
- CDROM building
  - **Build your own bootable Rocks CD**

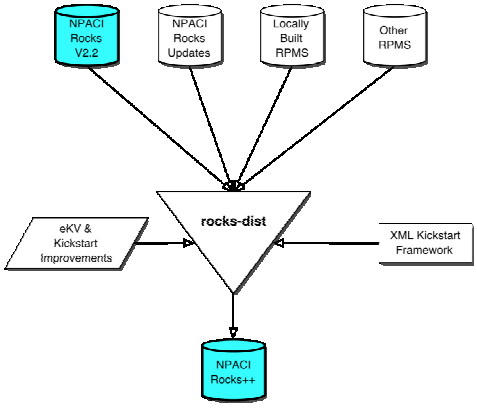
Command line

- # rocks-dist mirror
  - **Build (or update) a mirror of rocks.npaci.edu**
  - **All Rocks software plus constant Red Hat updates**
- # rocks-dist dist
  - **Build the distribution for compute nodes**
  - **Gathers packages from various places**
    - Mirror (/home/install/rocks.npaci.edu)
    - Local Contrib (/home/install/contrib)
    - Locally built packages (/usr/src/redhat/RPMS/*)
- Mirror and distribution are setup out of the box
  - **Need to "rocks-dist mirror" to get all the unused RPMs**
  - **Need to "rocks-dist dist" to build updated distributions**

How we use rocks-dist

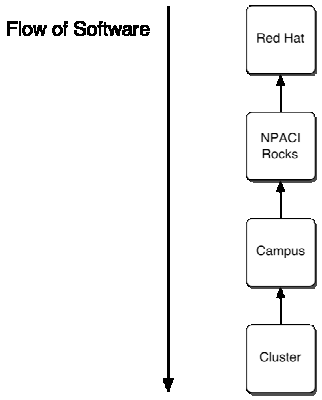


How you use rocks-dist

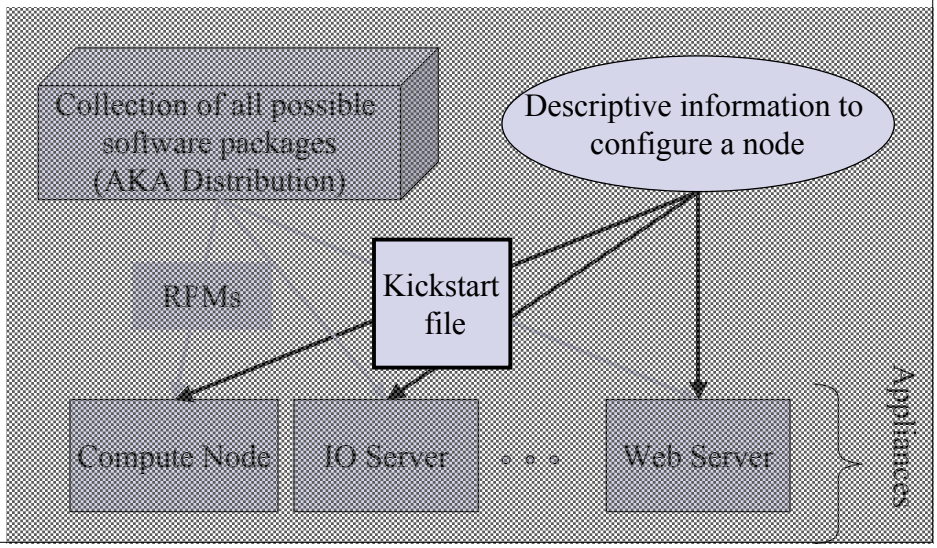


Inheritance

- Rocks
  - **Red Hat plus updates**
  - **Rocks software**
- Campus
  - **Rocks software**
  - **Campus changes**
- Cluster
  - **Campus Rocks**



Installation Instructions



Kickstart

- Description based installation
  - **Manage software components not the bits on the disk**
  - **Only way to deal with heterogeneous hardware**
    - System imaging (aka bit blasting) relies on homogeneity
    - Homogenous clusters do not exist
- Red Hat's Kickstart
  - **Flat ASCII file**
  - **No macro language**
  - **Requires forking based on site information and node type**
- Rocks' XML Kickstart
  - **Decompose a kickstart file into nodes and graphs**
  - **Macros and SQL for site configuration**
  - **Driven from web cgi script**

Kickstart File Sections

- Main
  - **Disk partitioning**
  - **Root password**
  - **RPM repository URL**
  - **...**
- Packages
  - **List of RPMs (w/o version numbers)**
  - **The repository determines the RPM versions**
  - **The kickstart file determines the set of RPMs**
- Pre
  - **Shell scripts run before RPMs are installed**
  - **Rarely used (Rocks uses it to enhance kickstart)**
- Post
  - **Shell scripts to cleanup RPM installation**
  - **Fixes bugs in packages**
  - **Adds local information**

XML Kickstart

- Nodes
  - **Describe a single set of functionality**
    - Ssh
    - Apache
  - **Kickstart file snippets (XML tags map to kickstart commands)**
  - **Pull site configuration from SQL Database**
  - **Over 80 node files in Rocks**
- Graph
  - **Defines interconnections for nodes**
  - **Think OOP or dependencies**
  - **A single graph file in Rocks**
- Graph + Nodes + SQL => Node specific kickstart file

Sample Node File

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE kickstart SYSTEM "@KICKSTART_DTD@" [<!ENTITY ssh "openssh">]>
<kickstart>
            <description>
            Enable SSH
            </description>

            <package>&ssh;</package>
            <package>&ssh;-clients</package>
            <package>&ssh;-server</package>
            <package>&ssh;-askpass</package>
<post>

cat &gt; /etc/ssh/ssh_config &lt;&lt; 'EOF' <!-- default client setup -->
Host *
        ForwardX11 yes
        ForwardAgent yes
EOF

chmod o+rx /root
mkdir /root/.ssh
chmod o+rx /root/.ssh

</post>
</kickstart>>
```

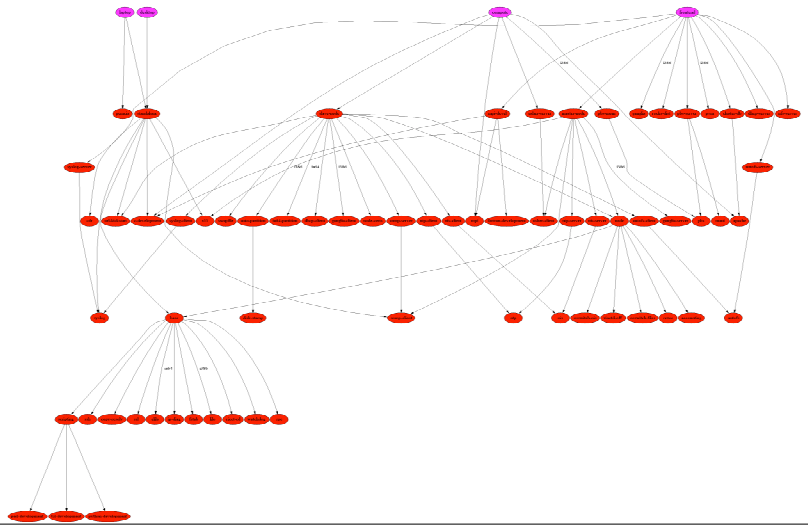
Sample Graph File

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

<graph>
            <description>
            Default Graph for NPACI Rocks.
            </description>

            <edge from="base" to="scripting"/>
            <edge from="base" to="ssh"/>
            <edge from="base" to="ssl"/>
            <edge from="base" to="lilo"  arch="i386"/>
            <edge from="base" to="elilo" arch="ia64"/>
                  …
            <edge from="node" to="base" weight="80"/>
            <edge from="node" to="accounting"/>
            <edge from="slave-node" to="node"/>
            <edge from="slave-node" to="nis-client"/>
            <edge from="slave-node" to="autofs-client"/>
            <edge from="slave-node" to="dhcp-client"/>
            <edge from="slave-node" to="snmp-server"/>
            <edge from="slave-node" to="node-certs"/>
            <edge from="compute" to="slave-node"/>
            <edge from="compute" to="usher-server"/>
            <edge from="master-node" to="node"/>
            <edge from="master-node" to="x11"/>
            <edge from="master-node" to="usher-client"/>
</graph>
```

Macros

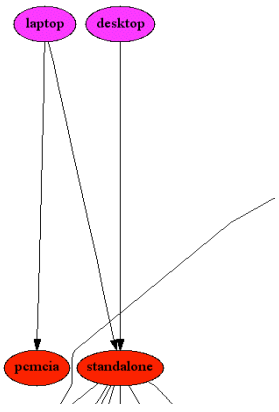
- SQL Database
  - **App_globals table defines over 20 variables based on the node's group membership**
- Creating new macros
  - **By value**
    - <var name="Foo" val="Bar"/>
  - **By reference**
    - <var name="Foo2" ref="Foo"/>
    - This is invalid: <var name="Foo2" val=<var name="Foo"/>/>
- Referencing
  - **<var name="Foo"/>**

Kickstart framework



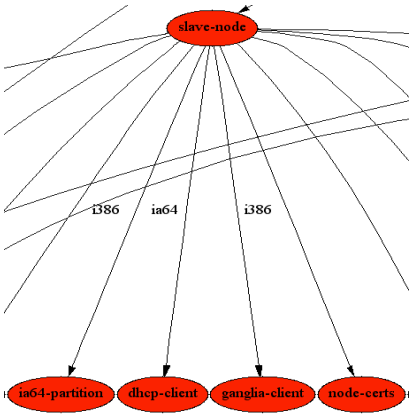
Appliances

- Laptop / Desktop
  - **Appliances**
  - **Final classes**
  - **Node types**
- Desktop IsA
  - **standalone**
- Laptop IsA
  - **standalone**
  - **pcmcia**
- Code re-use is good



Architecture Differences

- Annotate edges with the architecture of the node
- ia64-partition
  - **EFI (DOS) partition**
  - **Only for Itanium**
- dhcp-client / ganglia-client
  - **Not ported to Itanium**



Creating the Kickstart file

1. Node makes HTTP request to get configuration
   - **Can be online or captured to a file**
   - **Node reports architecture type, IP address, [*appliance type*], [*options*]**
2. Kpp – preprocessor
   - **Start at appliance type (node) and make a single large XML file by traversing the graph**
   - **Node-specific configuration looked up in SQL database**
3. Kgen – generation
   - **Translation to kickstart format**
   - **Other formats could be supported**

- Graph visualization using dot (AT&T)

Software

The Payoff of description based installation

Payoff #1

- Dual -Athlon white-box, 20GB IDE, 3COM Ethernet
  - **3:00 in cardboard box**
  - **Shook out loose screws and dropped in Myrinet card**
  - **Screwed into the rack and cable it up**
  - **3:25 inserted Rocks CD**
  - **3:40 ran Linpack**
- Completely foreign hardware
  - **Never done AMD**
  - **Never done IDE**
  - **Never done 3COM**

Payoff #2

- Two dual-Itanium, 18GB SCSI, Intel Ethernet
  - **2:00 in cardboard box**
  - **3:40 debugged 2.4.6 ia64 kernel bug**
  - **Downloaded Rawhide 2.4.9 kernel**
  - **Rebuilt Rocks distribution**
  - **4:30 both nodes integrated into the cluster**
- Second IA64 box we touched
- This one was hard!

Future Work

Real Job Control (and cleanup)

- Adequate job launching
  - **Mpirun**
    - From MPI software
  - **Mpi-launch**
    - SSH-based job launcher
  - **Rexec**
    - SSL-based job launcher from UCB
- Looking for better solutions

Storage

- We know NFS is wrong
  - **Linux NFS is really wrong**
  - **NAS is better**
  - **SAN with NFS metadata server isn't right either**
- Need flexible parallel file system
  - **Tunable performance**
  - **Tunable redundancy**
  - **Leverage existing cluster components**

System Monitoring

- Want lots of detail
  - **Snmp**
  - **Netsaint**
  - **http access to proc file system**
- Need scalability
  - **Ganglia**
- Two concepts
  - **Cluster wide status**
  - **Single node status**
- Want multiple views of cluster status

Itanium

- Rocks 2.1
  - **ia64 compute node support**
  - **Required i386 frontend machine**
- Rocks 2.2 ia64 support coming this month
  - **Still compute node only support**
- Next major release of Rocks will have full ia64 support

Grid

- Globus integration is too hard
  - **Local administrator (with 2 years Globus experience) can install Globus in 30 minutes**
  - **We can install a cluster in 30 minutes**
- Tracking NMI (National Middleware Initiative)
  - **Packaging Grid tools in target OS's native package format**
  - **Primary target is Linux**
- Does Grid = Globus?

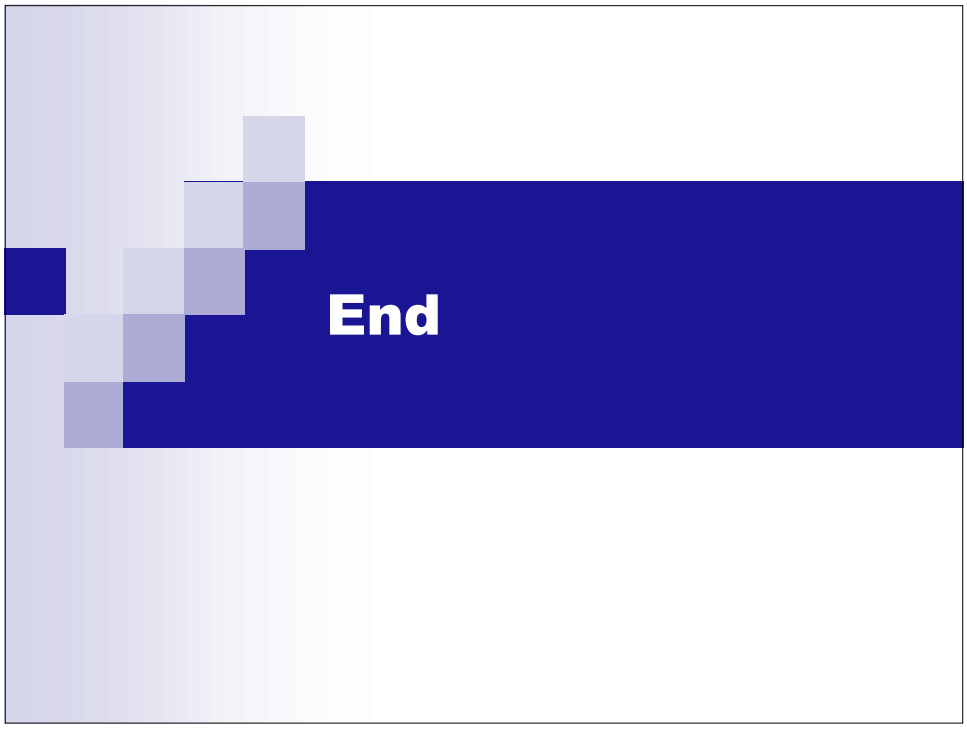
Lots of other stuff

- Track Red Hat faster than we have been
- Fix the frontend upgrade problem
  - **Compute node are easy**
  - **A lot more state on the frontend**
  - **Push all state into SQL database**
- Documentation
  - **Top priority now that 2.2 is out**
- Any one want a job?

Open Lab

Ideas

- Tour of rocks-dist
  - **Building personal distributions**
  - **Changing XML profiles**
- Database
  - **Change “compute-0-0” to “node-0-0”**
  - **Adding personal distributions**
- PBS
  - **Launch some jobs (linpack)**
- Build your own cluster


End