Introduction to Rocks

Rocks-A-Palooza II
Overview of Rocks

The Rocks software stack
Cluster Software Stack

- Parallel Code / WebFarm / Grid / Computer Lab
- Message Passing / Communication Layer
- Job Scheduling and Launching
- Cluster Software Management
- Cluster State Management / Monitoring
- Linux Environment
- HPC Device Drivers (e.g., Interconnect and Storage)
- Linux Kernel
Common to Any Cluster

- Message Passing / Communication Layer
- Job Scheduling and Launching
- Linux Environment
- Linux Kernel

© 2006 UC Regents
Red Hat

- Enterprise Linux 4.0
  - Recompiled from public SRPMS, including errata updates (source code)
  - No license fee required, redistribution is also fine
  - Recompiled for all CPU types (x86, Opteron, Itanium)
  - Rocks 5.0 will be based on RHEL 5.0 (Centos, or RHEL)

- Standard Red Hat Linux kernel
  - No Rocks added kernel patches

- No support for other distributions
  - Red Hat is the market leader for Linux
    - In the US
    - And becoming so in Europe
  - Trivial to support any Anaconda-based system
  - Others would be harder, and require vendor support (SuSe ~ 12 months work)

- Excellent support for automated installation
  - Scriptable installation (Kickstart)
  - Very good hardware detection
Dell Invests in Red Hat

Michael Dell puts $99.5M in Red Hat
Billionaire chairman of No. 1 PC maker places big bet on Microsoft competitor.
May 10, 2005: 1:41 PM EDT

NEW YORK (CNN/Money) - Red Hat is getting a $99.5 million boost from Michael S. Dell, billionaire founder and chairman of Dell Inc., according a regulatory filing.

Through his private investment firm, MSD, Dell bought the largest share of $500 million in debentures offered by the software developer in January 2004, a Securities Exchange Commission filing showed.

Red Hat's main product, the Linux operating system for PCs, is a direct competitor to Microsoft's Windows. The Raleigh, N.C.-based company also provides support services for "open source" technology, which is software developed by communities of programmers for free use.

Dell (Research) is the nation's largest PC maker.

Debentures are similar to bonds in that the issuer promises a fixed return for a stated period of time on the investment.

In the case of a public company, a debenture can also be converted into shares or equity.
Batch Systems

- Portable Batch System and Maui
  - Long time standard for HPC queuing systems
  - Maui provides backfilling for high throughput
  - PBS/Maui system can be fragile and unstable
  - Multiple code bases:
    - PBS
    - OpenPBS
    - PBSPro
    - Scalable PBS

- Sun Grid Engine
  - Rapidly becoming the new standard
  - Integrated into Rocks by Scalable Systems
    - See Najib
  - Now the default scheduler for Rocks
  - Robust and dynamic
Communication Layer

- None
  - “Embarrassingly Parallel”
- Sockets
  - Client-Server model
  - Point-to-point communication
- MPI - Message Passing Interface
  - Message Passing
  - Static model of participants
- PVM - Parallel Virtual Machines
  - Message Passing
  - For Heterogeneous architectures
  - Resource Control and Fault Tolerance
Sockets are low level

- **Sockets**
  - Point-to-Point
  - N machines = \( \frac{n^2 - n}{2} \) connections
  - 1, 3, 6, 10, 15, ...

- **MPI/PVM**
  - Shared virtual channel
  - Implementation could be sockets
  - Easier to program
Sockets

- Open an endpoint
- Specify IP address and port
- Send / receive messages
  - If TCP, only point-to-point messages
  - If UDP, option of point-to-point or multicast (broadcast)
- Shutdown connection
High-level TCP Example

```c
/* SERVER CODE */

fd = socket();
.
.
saddr.s_addr = INADDR_ANY;
saddr.port = 1234;
bind(fd, &saddr);
listen(fd);
accept(fd);
.
read(fd, buffer, size);
.
close(fd);

/* CLIENT CODE */

fd = socket();
.
.
saddr.s_addr = gethostbyname("c0-0");
saddr.port = 1234;
.
write(fd, buffer, size);
.
close(fd);
```
Challenges with Sockets

- **TCP**
  - Reliable, but byte oriented
  - Need to write code to send and receive *packets* (at the application level)

- **UDP**
  - Unreliable
  - Need to write code to reliably send packets
MPI

- Message Passing Interface
- De facto standard for message passing
  - Runs over many CPU architectures and many communication substrates
- There are (and were) lots of good messaging libraries
  - But, MPI is the most pervasive
  - Developed a practical, portable, efficient and flexible standard
  - In development since 1992
MPI

- Explicitly move data like sockets, but virtualizes the endpoints
  - Remote endpoints addressed by integer 0, 1, …, n
- Primitives to support point-to-point and broadcast

```
Process 0
Process 1
Process 2
Process 3
```
MPI

- Single interface to pass messages over many communication substrates

- MPI API
  - Ethernet
  - Myrinet
  - IB
  - Dolphin
High-level MPI Example

MPI_Init();

MPI_Comm_rank(&my_mpi_id);

Remote_mpi_id = 1
MPI_Send(send_buffer, buf_size, remote_mpi_id)

MPI_Recv(recv_buffer, buf_size, remote_mpi_id)

MPI_Finalize()
Challenges with MPI

- If a node fails, no easy way to reconfigure and route around the problem
  - Basically, your program stops
Compile

◆ MPICH with GNU Compilers and Ethernet

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:</td>
<td>/opt/mpich/ethernet/gcc/bin/mpicc</td>
</tr>
<tr>
<td>C++:</td>
<td>/opt/mpich/ethernet/gcc/bin/mpiCC</td>
</tr>
<tr>
<td>F77:</td>
<td>/opt/mpich/ethernet/gcc/bin/mpif77</td>
</tr>
</tbody>
</table>

◆ MPICH with GNU Compilers and Myrinet

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:</td>
<td>/opt/mpich/myrinet/gcc/bin/mpicc</td>
</tr>
<tr>
<td>C++:</td>
<td>/opt/mpich/myrinet/gcc/bin/mpiCC</td>
</tr>
<tr>
<td>F77:</td>
<td>/opt/mpich/myrinet/g77/bin/mpif77</td>
</tr>
</tbody>
</table>
Compile

**MPICH with Intel Compilers and Ethernet**

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:</td>
<td><code>/opt/mpich/ethernet/ecc/mpicc</code></td>
</tr>
<tr>
<td>C++:</td>
<td><code>/opt/mpich/ethernet/ecc/mpiCC</code></td>
</tr>
<tr>
<td>F77:</td>
<td><code>/opt/mpich/ethernet/ecc/mpif77</code></td>
</tr>
<tr>
<td>F90:</td>
<td><code>/opt/mpich/ethernet/ecc/mpif90</code></td>
</tr>
</tbody>
</table>

**MPICH with Intel Compilers and Myrinet**

<table>
<thead>
<tr>
<th>Compiler</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>C:</td>
<td><code>/opt/mpich/myrinet/ecc/mpicc</code></td>
</tr>
<tr>
<td>C++:</td>
<td><code>/opt/mpich/myrinet/ecc/mpiCC</code></td>
</tr>
<tr>
<td>F77:</td>
<td><code>/opt/mpich/myrinet/efc/mpif77</code></td>
</tr>
<tr>
<td>F90:</td>
<td><code>/opt/mpich/myrinet/efc/mpif90</code></td>
</tr>
</tbody>
</table>
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/
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
  - Mpirun
  - Cluster-fork

- Ssh-agent
  - Manager for SSH keys
  - ssh-agent $SHELL
Rocks Cluster Software
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
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
eKV

- Remotely Interact with Installation
  - Initial kickstart
  - Re-Installation
- Shoot-node
  - Reinstall OS and brings up eKV
- eKV
  - Ssh to node while it is installing
  - See the console output over Ethernet
Cluster State Management

- **Static Information**
  - Node addresses
  - Node types
  - Site-specific configuration

- **Dynamic Information**
  - CPU utilization
  - Disk utilization
  - Which nodes are online
Cluster Database
Node Info Stored In A MySQL Database

- If you know SQL, you can execute powerful commands
  - Rocks-supplied command line utilities are tied into the database

  - E.g., get the hostname for the bottom 8 nodes of each cabinet:

    `# cluster-fork --query="select name from nodes where rank<8" hostname`
Ganglia (or SCMSWeb / SCE Roll)

- 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 Screenshot

Our Cluster > britannic

britannic Overview

This node is up and running

Time and String Metrics

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>boottime</td>
<td>Tue, 18 Mar 2003 00:23:20 +0000</td>
</tr>
<tr>
<td>gexec</td>
<td>OFF</td>
</tr>
<tr>
<td>machine_type</td>
<td>ia64</td>
</tr>
<tr>
<td>os_name</td>
<td>Linux</td>
</tr>
<tr>
<td>os_release</td>
<td>2.4.18-1.12smp</td>
</tr>
<tr>
<td>sys.clock</td>
<td>Tue, 18 Mar 2003 00:25:34 +0000</td>
</tr>
<tr>
<td>uptime</td>
<td>0 day, 1:5</td>
</tr>
</tbody>
</table>

Constant Metrics

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu_idle</td>
<td>97.1%</td>
</tr>
<tr>
<td>cpu_num</td>
<td>2</td>
</tr>
<tr>
<td>cpu_speed</td>
<td>900 MHz</td>
</tr>
<tr>
<td>mem_total</td>
<td>1011568 KB</td>
</tr>
<tr>
<td>mtu</td>
<td>1500 B</td>
</tr>
<tr>
<td>swap_total</td>
<td>1048544 KB</td>
</tr>
</tbody>
</table>
SCMSWeb Screenshot
Software Installation

Collection of all possible software packages (AKA Distribution)

Descriptive information to configure a node

RPMs

Kickstart file

Compute Node

IO Server

Web Server

Appliances
Software Repository

Collection of all possible software packages (AKA Distribution)

RPMs

Kickstart file

Compute Node

IO Server

Web Server

Descriptive information to configure a node

Appliances
Installation Instructions

Collection of all possible software packages (AKA Distribution)

RFMs

Kickstart file

Descriptive information to configure a node

Compute Node

IO Server

Web Server
Cluster Software Management

Software Packages

◆ RPMs
  ➡ Standard Red Hat (desktop) packaged software
  ➡ Or your own addons

◆ Rocks-dist
  ➡ Manages the RPM repository
  ➡ This is the distribution

Software Configuration

◆ Tuning RPMs
  ➡ For clusters
  ➡ For your site
  ➡ Other customization

◆ XML Kickstart
  ➡ Programmatic System Building
  ➡ Scalable
Building a Rocks Distribution

- Start with Red Hat
- Add updates, Rocks (and optional other) software
- Add Kickstart profiles
- Modify Red Hat installation boot image
- Resulting in a Red Hat compatible Rocks distribution

© 2006 UC Regents
Kickstart

- **Red Hat’s Kickstart**
  - Monolithic 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 a graph
    - Graph specifies OO framework
    - Each node specifies a service and its configuration
  - 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
Sample Node File

```xml
<?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
<?xml version="1.0" standalone="no"?>
<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="grub" arch="i386"/>
  <edge from="base" to="elilo" arch="ia64"/>
  ...  
  <edge from="node" to="base"/>
  <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>
```
Kickstart Framework
Appliances

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

- Conditional inheritance
- Annotate edges with target architectures
  - if i386
    - Base IsA grub
  - if ia64
    - Base IsA elilo
- One Graph, Many CPUs
  - Heterogeneity is easy
  - Not true for SSI or Imaging
Optional Drivers

◆ PVFS
  ✉ Parallel Virtual File System
  ✉ Kernel module built for all nodes
  ✉ 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

◆ Add your own
  ✉ Cluster Gigabit Ethernet driver
  ✉ Infiniband driver
Application Layer

- **Rocks Rolls**
  - Optional component
  - Created by SDSC
  - Created by others

- **Example**
  - Bio (BLAST)
  - Chem (GAMESS)
  - Visualization Clusters
Building on Top of Rocks

Inheritance and Rolls
How Rocks in built

🔹 Rocks-dist
  ✅ Merges all RPMs
    • Red Hat
    • Rocks
  ✅ Resolves versions
  ✅ Creates Rocks

🔹 Rocks distribution
  ✅ Looks just like Red Hat
  ✅ Cluster optimized Red Hat
How You Create Your Own Rocks

- Rocks-dist
  - Merges all RPMs
    - Rocks
    - Yours
  - Resolves versions
  - Creates Rocks++

- Your distribution
  - Looks just like Rocks
  - Application optimized Rocks
Extension Through Inheritance

- UCSD/SDSC Rocks
  - BIRN
  - GAMESS Portal
  - GEON
  - GriPhyN
  - Camera
  - Optiputer

- Commercial
  - Scalable Systems
  - Platform Computing

- Can also override existing functionality
  - Rocks without NFS?
  - Rocks for the desktop?
Rolls

Think of a roll as a “package” for a car
Rolls Break Apart Rocks

- Parallel Code / WebFarm / Grid / Computer Lab
- Message Passing / Communication Layer
- Job Scheduling and Launching
- Cluster Software Management
- Cluster State Management / Monitoring
- Linux Environment
- HPC Device Drivers (e.g., Interconnect and Storage)
- Linux Kernel
Rocks is What You Make it

**Motivation**
- "I’m concerned Rocks is becoming everything for everyone" - rocks mailing list
- “Building a cluster should be like ordering a car. I want the sports package, but not the leather seats, …” - z4 owning rocks developer
- We need to let go of Rocks but hold onto the core
  - Recruit more external open-source developers
  - Only trust ourselves with fundamental architecture and implementation
- We wanted to move the SGE but need to still support PBS

**Rolls**
- Optional configuration and software
- Just another CD for installed (think application pack)
- SGE and PBS are different Rolls
  - User chooses scheduler
  - PBS Roll supported by Norway
  - SGE Roll supported by Singapore (and us)
- Rolls give us more flexibility and less work

**Rocks is done**
- The core is basically stable and needs continued support
- Rolls allow us to develop new ideas
- Application Domain specific

Extensible Rocks

- Over a dozen Rolls already created (e.g.):
  - SGE, PBS
  - Grid (NMI stack)
  - Java
  - Condor
  - SCE

- Several third party Rolls have started:
  - Quadrics (rumored)
  - PGI (just Released)
  - NIMROD
  - BIRN
  - DB2

- Rocks is done:
  - The core is basically stable and needs continued support
  - Rolls allow us to develop new ideas
  - Application Domain specific
  - For example: Visualization…
Viz Roll

Rocks becomes more than just compute clusters
Early Work: NCSA

- **LCD Cluster**
  - Custom framing
  - One PC / tile
  - Portable (luggable)
  - SC 2001 Demo

- **NCSA Software**
  - Pixel Blaster
  - Display Wall In-A-Box
  - OSCAR based
  - Never fully released
NCMIR

- Using Rocks
- Hand configured a visualization cluster
- “Administered the machine to the point of instability” - David Lee
- Automation is needed
COTS Vis: GeoWall

- LCD Clusters
  - One PC / tile
  - Gigabit Ethernet
  - Optional Stereo Glasses
  - Portable
  - Commercial Frame (Reason)

- Applications
  - Large remote sensing
  - Volume Rendering
  - Seismic Interpretation
  - Brain mapping (NCMIR)

- Electronic Visualization Lab
  - Jason Leigh (UIC)
Eye Candy (NCMIR)
Rocks Installation

Step by step instruction for building your cluster
Frontend Installation

- Turn on node
- Insert CDROM
- Type
  - `frontend`

```bash
# frontend  
For a new installation.

# frontend upgrade  
For an upgrade installation.

# frontend central=NAME  
For a new network based installation. Where name is "rocks" or the FQDN of your central server.

# frontend rescue  
To boot into rescue mode.
```

Client

do nothing (default)
Rolls

- Anaconda Starts
- Asks for Rolls
- Select “Yes”
- Insert
  - base
  - hpc+kernel
  - area51+java+grid+sge
Cluster Information

- Specific to Rocks
- Used for Certificates
  - SSL/HTTPS
  - Globus
- Hostname
  - Must be FQDN
  - Must be in DNS
  - Must not be an Alias
Partitioning

- **Automatic**
  - 6GB /
  - 1GB swap
  - Remainder for /export

- **Manual**
  - You choose
  - Must create a /export

- **Select Wisely**
Networks

- Private Network
  - eth0
  - Cluster-side only
- Public Network
  - eth1
  - Internet/LAN side
- You must configure both and have 2 NICs
Gateway

- Gateway / DNS
  - Same as any other device on your network
- All traffic for compute nodes is NATed through the frontend.
- DNS is only for the frontend, compute nodes use the frontend as their DNS.
Network Time Protocol

◆ Choose timezone
  ➔ UTC is a good choice
  ➔ Or localize

◆ Default server is
  ➔ time.apple.com
  ➔ Change it if you wish
Root Password

- Password is secure
  - Not stored in clear text form anywhere (not in DB)
- Also used for mysql password
Installing Packages
Integrate Compute Nodes

- Log into Frontend (as root)
- Run `insert-ethers`
  - Can choose appliance type
  - Rolls add new appliance types
  - For now we will use Compute
- Turn on first node
  - Nodes are integrated serially
  - Need to map machine name to machine location
  - After we integrate machines can be re-installed in parallel
- Remote Terminal (ekv)
  - `ssh compute-0-0 -p2200`
Discovering Compute-0-0

Retrieved kickstart file
useradd
user login

```
$ ssh concave.rockschusters.org
mjk@concaev.rockschusters.org's password:
Last login: Mon May 16 19:50:09 2005 from client64-84.sdsc.edu
Rocks Frontend Node - Rocks-39 Cluster
Rocks 4.0.0 (Whitney)
Profile built 13:03 26-Apr-2005

Kickstarted 13:03 26-Apr-2005

It doesn't appear that you have set up your ssh key.
This process will make the files:
/home/mjk/.ssh/id_rsa.pub
/home/mjk/.ssh/id_rsa
/home/mjk/.ssh/authorized_keys

Generating public/private rsa key pair.
Enter file in which to save the key (/home/mjk/.ssh/id_rsa):
Enter directory '/home/mjk/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/mjk/.ssh/id_rsa.
Your public key has been saved in /home/mjk/.ssh/id_rsa.pub.
The key fingerprint is:
[mjk@rocks-39 ~]$ 
```
End