User Session 2
Introduction to Rocks

Rocks-A-Palooza III
Traditional Cluster Architecture

- Frontend Node
- Private Ethernet Network
- Application Network (Optional)
- Public Ethernet
- Power Distribution (Net addressable units as option)
Modern Cluster Architecture
User View of Cluster
www.camera.calit2.net

Sequence Alignment

Sequence: JCVI_READ_2056128  Score: 38,1576  Identities: 19/19 (100%)
Sequence Length: 1059  Expect: 0.789518  Positives: 0 / 19 (0%)
Alignment Length: 19  Query Begin/End: 25 - 43 (Plus)  Query Gaps: 0
Clear Range: 79 - 906  Subject Begin/End: 811 - 829 (Plus)  Subject Gaps: 0

Query: CGTTCACACACGTTCACACG 43
Subject: CGTTCACACACGTTCACACG 829
Visualization Clusters

- Cluster of GPUs
  - OpenGL machine
  - Not an MPI machine
- Massive Pixel Walls
  - 60 MegaPixels
  - Full rate HDTV
- Software
  - SAGE
  - DMX
  - Chromium
Diverse Grid End-Points

ganglia.sourceforge.net - UCB

Rocks Monitoring Grid (5 sources)
CPUs Total: 1712
Hosts up: 689
Hosts down: 37
Avg Load (15, 5, 1m):
29%, 29%, 29%
Localtime:
2007-03-15 18:23

Meta AKA rocks-135
CPUs Total: 4
Hosts up: 1
Hosts down: 0
Avg Load (15, 5, 1m):
74%, 67%, 69%
Localtime:
2007-03-15 18:22

Monitoring / Management
UCSD OptIPuter Grid
CPUs Total: 102
Hosts up: 34
Hosts down: 35
Avg Load (15, 5, 1m):
48%, 48%, 49%
Localtime:
2007-03-15 18:22

Visualization Cluster
CAMERA Grid
CPUs Total: 624
Hosts up: 172
Hosts down: 0
Avg Load (15, 5, 1m):
9%, 9%, 9%
Localtime:
2007-03-15 18:22
key point

Rocks builds more than just MPI machines
Basic 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

© 2007 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 to 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
  ➢ 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 = \((n^2 - n)/2\) 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

/*
 * 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
High-level MPI Example

```c
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

- Hard to manage deployment
  - network X compiler = mpi binaries
  - Result is several versions of MPI / cluster

© 2007 UC Regents
## 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>/opt/mpich/ethernet/ecc/mpicc</td>
</tr>
<tr>
<td>C++:</td>
<td>/opt/mpich/ethernet/ecc/mpiCC</td>
</tr>
<tr>
<td>F77:</td>
<td>/opt/mpich/ethernet/ecc/mpif77</td>
</tr>
<tr>
<td>F90:</td>
<td>/opt/mpich/ethernet/ecc/mpif90</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>/opt/mpich/myrinet/ecc/mpicc</td>
</tr>
<tr>
<td>C++:</td>
<td>/opt/mpich/myrinet/ecc/mpiCC</td>
</tr>
<tr>
<td>F77:</td>
<td>/opt/mpich/myrinet/efc/mpif77</td>
</tr>
<tr>
<td>F90:</td>
<td>/opt/mpich/myrinet/efc/mpif90</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
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
- Newer versions of Rocks (4.0+)
  use VNC
  - Graphical
  - Works on headless machines
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

- **Video**
  - nVidia (from Viz Roll)

- **Add your own**
  - Cluster Gigabit Ethernet driver
  - Infiniband driver

- **Kernel Modules are dynamically built**
- **No need to manage binary Kernel Modules**
- **Burn CPU time, not human time**
Ganglia

- Scalable cluster monitoring system
  - Based on ip multi-cast
  - Matt Massie, et al from UCB
  - [http://ganglia.sourceforge.net](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

Host Report for Tue, 18 Mar 2003 01:28:58 +0000

Last: hour: 2

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>2.4.18-0.12smp</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>1011588 KB</td>
</tr>
<tr>
<td>mtsu</td>
<td>1500 B</td>
</tr>
<tr>
<td>swap_total</td>
<td>1048544 KB</td>
</tr>
</tbody>
</table>
SCMSWeb Screenshot
Rocks Cluster Software

Cluster Software Management

Cluster State Management / Monitoring

HPC Device Drivers
(e.g., Interconnect and Storage)

Middleware

Kernel

Applications
Cluster State Management

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

- Dynamic Information
  - CPU utilization
  - Disk utilization
  - Which nodes are online
Cluster 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
```
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)

- RPMs
- Kickstart file
- Descriptive information to configure a node

Appliances

- 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
Nodes Main: Partitioning

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

```
part / --size 8000 --ondisk hda
part swap --size 1000 --ondisk hda
part /mydata --size 1 --grow --ondisk hda
```
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
What is a Kickstart File?

Setup & Packages (20%)

```
setup
zerombr yes
bootloader --location mbr --useLilo
skipx
auth --useshadow --enablemd5

clearpart --all
    part /boot --size 128
    part swap --size 128
    part / --size 4096
    part /export --size 1 --grow

lang en_US

langsupport --default en_US

keyboard us

mouse genericps/2

timezone --utc GMT

rootpw --iscrypted nrDq4Vb42jjQ.

text

install

reboot
```

Post Configuration (80%)

```
post

cat > /etc/nsswitch.conf << 'EOF'
    passwd:    files
    shadow:    files
    group:     files
    hosts:     files
dns
    bootparams:  files
    ethers:     files
    EOF

    cat > /etc/ntp.conf << 'EOF'
    server ntp.ucsd.edu
    server 127.127.1.1
    fudge 127.127.1.1 stratum 10
    authenticate no
    driftfile /etc/ntp/drift
    EOF

    /bin/mkdir -p /etc/ntp
    cat > /etc/ntp/step-tickers << 'EOF'
    ntp.ucsd.edu
    EOF

    /usr/sbin/ntpd /etc/ntp
    /usr/sbin/ntpd --ntpd
    /sbin/hwclock --systohc
```

© 2007 UC Regents
Issues

- High level description of software installation
  - List of packages (RPMs)
  - System configuration (network, disk, accounts, …)
  - Post installation scripts

- *De facto* standard for Linux

- Single ASCII file
  - Simple, clean, and portable
  - Installer can handle simple hardware differences

- Monolithic
  - No macro language
  - Differences require forking (and code replication)
  - Cut-and-Paste is not a code re-use model
XML Kickstart
It looks something like this
Implementation

◆ Nodes
  ➤ Single purpose modules
  ➤ Kickstart file snippets (XML tags map to kickstart commands)
  ➤ Approximately 200 node files in Rocks

◆ Graph
  ➤ Defines interconnections for nodes
  ➤ Think OOP or dependencies (class, #include)
  ➤ A single default graph file in Rocks

◆ Macros
  ➤ SQL Database holds site and node specific state
  ➤ Node files may contain <var name="state"/> tags
Composition

- Aggregate Functionality
- Scripting
  - IsA perl-development
  - IsA python-development
  - IsA tcl-development
Appliances

- Laptop / Desktop
  - Appliances
  - Final classes
  - Node types
- Desktop IsA
  - standalone
- Laptop IsA
  - standalone
  - pcmcia
- Specify only the differences
Architecture Differences

- Conditional inheritance
- Annotate edges with target architectures
- if i386
  - Base IsA grub
- if ia64
  - Base IsA elilo
- One Graph, Many CPU Architectures
  - Heterogeneity becomes easy
  - Not true for SSI or Imaging
Putting in all together
key point

The XML Graph is the DNA of the cluster
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
  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>
```
Cluster SQL Database
# Nodes and Groups

## Nodes Table

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

## Memberships Table

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Appliance</th>
<th>Distribution</th>
<th>Compute</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Frontend</td>
<td>1</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>Compute</td>
<td>2</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>PVFS I/O Node</td>
<td>3</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>Compute with PVFS</td>
<td>4</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>5</td>
<td>Laptop</td>
<td>5</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>6</td>
<td>Ethernet Switches</td>
<td>6</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>7</td>
<td>Myrinet Switches</td>
<td>6</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>8</td>
<td>Power Units</td>
<td>7</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>9</td>
<td>Remote Management</td>
<td>8</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>10</td>
<td>DTF Compute</td>
<td>9</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>11</td>
<td>Web Portal</td>
<td>10</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>12</td>
<td>NFS Server</td>
<td>11</td>
<td>1</td>
<td>no</td>
</tr>
</tbody>
</table>
# Groups and Appliances

## Appliances Table

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

## Memberships Table

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>ShortName</th>
<th>Graph</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>frontend</td>
<td>f</td>
<td>default</td>
<td>frontend</td>
</tr>
<tr>
<td>2</td>
<td>compute</td>
<td>c</td>
<td>default</td>
<td>compute</td>
</tr>
<tr>
<td>3</td>
<td>pvfs</td>
<td>pv</td>
<td>default</td>
<td>pvfs-io</td>
</tr>
<tr>
<td>4</td>
<td>comp-pvfs</td>
<td>cp</td>
<td>default</td>
<td>compute-pvfs</td>
</tr>
<tr>
<td>5</td>
<td>laptop</td>
<td></td>
<td>default</td>
<td>laptop</td>
</tr>
<tr>
<td>6</td>
<td>network</td>
<td>n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>power</td>
<td>p</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>manager</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>df</td>
<td>d</td>
<td>default</td>
<td>df-compute</td>
</tr>
<tr>
<td>10</td>
<td>portal</td>
<td>pl</td>
<td>default</td>
<td>portal</td>
</tr>
<tr>
<td>11</td>
<td>nfs</td>
<td>n</td>
<td>default</td>
<td>nfs</td>
</tr>
</tbody>
</table>
Simple key - value pairs

- Used to configure DHCP and to customize appliance kickstart files

<table>
<thead>
<tr>
<th>ID</th>
<th>Membership</th>
<th>Service</th>
<th>Component</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Kickstart</td>
<td>PublicNTPHost</td>
<td>ntp.ucsd.edu</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>Kickstart</td>
<td>ZeroMBR</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>Kickstart</td>
<td>PrivateKickstartCGI</td>
<td>kickstart.cgi</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>Kickstart</td>
<td>PublicNetmask</td>
<td>255.255.255.0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>Kickstart</td>
<td>PublicNetwork</td>
<td>192.31.21.0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>Kickstart</td>
<td>PrivateNISMaster</td>
<td>frontend-0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>Kickstart</td>
<td>PrivateHostname</td>
<td>frontend-0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>Kickstart</td>
<td>PrivateIPForwarding</td>
<td>yes</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>Kickstart</td>
<td>PrivateGateway</td>
<td>10.1.1.1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>Kickstart</td>
<td>PublicKickstartBasedir</td>
<td>install</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>Kickstart</td>
<td>Lang</td>
<td>en_US</td>
</tr>
</tbody>
</table>
Nodes XML Tools: <var>

◆ Get Variables from Database
  ✱ <var name="Kickstart_PrivateGateway"/>
  ✱ <var name="Node_Hostname"/>

  10.1.1.1
  compute-0-0

◆ Can grab any value from the app_globals database table
Nodes XML Tools: \texttt{<eval>}

- Do processing on the frontend:
  \[ \texttt{<eval shell="bash">} \]

- To insert a fortune in the kickstart file:

\begin{verbatim}
<eval shell="bash"> \\
/usr/games/fortune \\
</eval>
\end{verbatim}

"Been through Hell? Whaddya bring back for me?"
-- A. Brilliant
Nodes XML Tools: `<eval>`

- Inside `<eval>` variables are not accessed with `<var>`: use the environment instead.

```xml
<eval shell="sh">
  echo "My NTP time server is $Kickstart_PublicNTPHost"
  echo "Got it?"
</eval>

My NTP time server is time.apple.com
Got it?

<eval shell="python">
  import os
  print "My NTP time server is",
  os.environ['Kickstart_PublicNTPHost']
  print "Got it?"
</eval>

My NTP time server is time.apple.com
Got it?
Nodes XML Tools <include>

- Auto-quote XML characters in a file
  - `<include file="foo.py"/>

- Quotes and includes file
  - `sweetroll/include/foo.py`

- `foo.py` (native) → `foo.py` (quoted xml):

```python
#!/usr/bin/python
import sys
def hi(s):
    print >> sys.stderr, s
```

```python
#!/usr/bin/python
import sys
def hi(s):
    print >&2; sys.stderr, s
```

© 2007 UC Regents
Nodes XML Tools `<file>`

- Create a file on the system:
  - `<file name="/etc/hi-mom" mode="append">
    · How are you today?
  </file>`

- Used extensively throughout Rocks post sections
  - Keeps track of alterations automatically via RCS.

```xml
<file name="/etc/hi" perms="444">
  How are you today?
  I am fine.
</file>
```

```bash
...RCS checkin commands...
cat > /etc/hi << 'EOF'
How are you today?
I am fine.
EOF
chmod 444 /etc/hi-mom
...RCS cleanup commands...
```
Fancy <file>: nested tags

<file name="/etc/hi">

Here is your fortune for today:
<eval>
date +"%d-%b-%Y"
echo ""
/usr/games/fortune
</eval>

</file>

…RCS checkin commands...
cat > /etc/hi << ‘EOF’

Here is your fortune for today:
13-May-2005

"Been through Hell? Whaddya bring back for me?"
-- A. Brilliant

EOF
…RCS cleanup commands…
Nodes Main

- Used to specify basic configuration:
  - timezone
  - mouse, keyboard types
  - install language
- Used more rarely than other tags
- Rocks main tags are usually a straight translation:

```xml
<main>
  <timezone>America/Mission_Beach</timezone>
  rootpw --iscrypted sndk48shdlwis
  mouse genericps/2
  url --url http://10.1.1.1/install/rocks-dist/..
</main>
```
Nodes Packages

- `<package>java</package>`
  - Specifies an RPM package. Version is automatically determined: take the *newest* rpm on the system with the name ‘java’.
- `<package arch="x86_64">java</package>`
  - Only install this package on x86_64 architectures
- `<package arch="i386,x86_64">java</package>`

```
<package>newcastle</package>
<package>stone-pale</package>
<package>guinness</package>

%packages
newcastle
stone-pale
guinness
```
Nodes Packages

- **RPM name is a basename (not full name of RPM)**
  - For example, RPM name of package below is ‘kernel’

```bash
# rpm -qip /home/install/rocks-dist/lan/i386/RedHat/RPMS/kernel-2.6.9-22.EL.i686.rpm
```

<table>
<thead>
<tr>
<th>Name</th>
<th>kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>2.6.9</td>
</tr>
<tr>
<td>Release</td>
<td>22.EL</td>
</tr>
<tr>
<td>Group</td>
<td>System Environment/Kernel</td>
</tr>
<tr>
<td>Size</td>
<td>25589794</td>
</tr>
<tr>
<td>Signature</td>
<td>DSA/SHA1, Sun 09 Oct 2005 10:44:40 AM WET, Key ID a53d0bab443e1821</td>
</tr>
<tr>
<td>Packager</td>
<td>Johnny Hughes <a href="mailto:johnny@centos.org">johnny@centos.org</a></td>
</tr>
<tr>
<td>Summary</td>
<td>the linux kernel (the core of the linux operating system)</td>
</tr>
<tr>
<td>Description</td>
<td>The kernel package contains the Linux kernel (vmlinuz), the core of any Linux operating system</td>
</tr>
</tbody>
</table>

License: GPLv2

Vendor: CentOS
Build Date: Sun 09 Oct 2005 03:01:51 AM WET
Build Host: louisa.home.local
Install Date: (not installed)
Nodes Post

◆ `<post>` for *Post-Install* configuration scripts

◆ Configuration scripts in `<post>` section run after all RPMs have been installed.
  ✐ Useful: you have all your software available
  ✐ Scripts run in “target” environment: /etc in `<post>` will be /etc on the final installed system

◆ Scripts are always non-interactive
  ✐ No Human is driving
Nodes Post

ntp-client.xml

```xml
<post>
/bin/mkdir -p /etc/ntp
/usr/sbin/ntpdate <var name="Kickstart_PrivateNTPHost"/>
/sbin/hwclock --systohc
</post>

%post

/bin/mkdir -p /etc/ntp
/usr/sbin/ntpdate 10.1.1.1
/sbin/hwclock --systohc
```
Putting it together
Space-Time and HTTP

Node Appliances

- DHCP
- Kickstart RQST
  - Request Package
  - Install Package
  - Post Config
  - Reboot

Frontends/Servers

- IP + Kickstart URL
- Generate File
- Serve Packages
  - kpp
- SQL DB
  - kgen

HTTP:
- Kickstart URL (Generator) can be anywhere
- Package Server can be (a different) anywhere
Coalescing Node Files

- Traverse a graph to build up a kickstart file
- Makes kickstart file building flexible
- Easy to share functionality between disparate node types
Another Look at XML

<graph>

<edge from="client">
<to arch="i386,x86_64">grub-client</to>
<to>autofs-client</to>
<to>installclass-client</to>
</edge>

<graph>
Partial Ordering

```xml
<graph>
  <order head="autofs-client" tail="client"/>
  <edge from="client">
    <to arch="i386,x86_64">grub-client</to>
    <to>autofs-client</to>
    <to>installclass-client</to>
  </edge>
</graph>
```

- Forces `autofs-client <post>` section to run before client’s `<post>` section
- In order graph traversal enforces a partial ordering
- Applying standard graph theory to system installation
Application Layer

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

- **Example**
  - Bio (BLAST)
  - Chem (GAMESS)
  - Visualization Clusters