Cluster Management and Maintenance

Rocks-A-Palooza II
Lab Session
What We’ll Be Doing

- Adding content to frontend’s web site
- Discuss how to add new packages to compute nodes
- How to change configuration on compute node
- Adding an application to the compute nodes
- Discuss frontend and compute node partitioning
- Configuring additional ethernet interfaces on compute nodes
Add Content to the Frontend’s Web Site
Adding Content to the Frontend’s Web Site

◆ First, configure X
  ➜ # system-config-display

◆ Start the X window server
  ➜ # startx
Adding Content To Frontend’s Web Site

- Connect to web page
  - # firefox http://localhost/

- Click on link at bottom of page:
  - “Add content to this web site”

- Next screen you see ‘Login/Password’
  - Login = ‘admin’
  - Password = same as root password on frontend
Adding Content To Frontend’s Web Site

- Click ‘Write’ tab
Adding Content To Frontend’s Web Site

- Write your ‘post’, then ‘publish’
Adding Content To Frontend’s Web Site

- View your new web site at:
  - http://localhost/
Add A New Package
Adding a New Package to the Distribution

◆ All packages are found under ‘/home/install’
◆ Put the new package in /home/install/contrib/4.1/i386/RPMS
  ❍ Where <arch> is ‘i386’, ‘x86_64’ or ‘ia64’
◆ “Extend” an XML configuration file
◆ Rebind the distro:
  # cd /home/install
  # rocks-dist dist
◆ Apply the changes by reinstalling the compute nodes:
  ❍ “shoot-node compute-0-0”
Extend the “Compute” XML Configuration File

- To add the package named “strace”

  $ cd /home/install/site-profiles/4.1/nodes
  $ cp skeleton.xml extend-compute.xml

- In ‘extend-compute.xml’, change:

  <package> <!-- insert your 1st package name here --> </package>

- To:

  <package>strace</package>
Extend the “Compute” XML Configuration File

- **Rebind the distro**
  - This copies `extend-compute.xml` into `/home/install/rocks-dist/.../build/nodes`
    
    ```
    # cd /home/install
    # rocks-dist dist
    ```

- **Test the changes**
  - Generate a test kickstart file
    
    ```
    # dbreport kickstart compute-0-0 > /tmp/ks.cfg
    ```
  - You should see ‘strace’ under the ‘%packages’ section
Extend the “Compute” XML Configuration File

◆ When you are satisfied with the changes, reinstall a compute node

    # shoot-node compute-0-0
More on the Distro

- Rocks-dist looks for packages in:
  - “/home/install/ftp.rocksclusters.org”
    - RedHat and Rocks packages
  - “/home/install/contrib”
    - Pre-built 3rd party packages
  - “/usr/src/redhat/RPMS”
    - RedHat default location for ‘built’ packages
    - But, when building packages in Rocks source tree, packages are not placed here
      - The packages are placed local to the roll source code
More on the Distro

- Any time you add a package to the distro, you must re-run “rocks-dist dist dist”
- Rocks-dist binds all the found packages into a RedHat-compliant distribution
More on the Distro

Rocks-dist assembles a RedHat compliant distribution
Your Distro - Extending Rocks

- You can use “rocks-dist” to build and distribute your own distribution
  - Merges RPMS
  - Resolves versions

- Final distribution looks just like Rocks
  - And, Rocks looks just like Red Hat
Add an Application to the Compute Nodes
Default NFS Share

- By default, each node has access to NFS shared directory named ‘/share/apps’

- The actual location is on the frontend
  - ‘/export/apps’ on the frontend is mounted on all nodes as ‘/share/apps’

- Simply add directories and files to /export/apps on frontend
Default NFS Share - Example

- **On frontend:**
  
  ```
  # cd /export/apps
  # touch myapp
  ```

- **On compute node:**
  
  ```
  # ssh compute-0-0
  # cd /share/apps
  # ls
  # ls
  myapp
  ```
Default NFS Share
Adding ‘bonnie’

- Bonnie is a file system benchmark
  - See ‘Introduction to Benchmarking’ Lab
- We’ll download the source and build it
  - On frontend:

  ```bash
  # cd /share/apps
  # mkdir benchmarks
  # mkdir benchmarks/bonnie++
  # cd benchmarks/bonnie++
  # mkdir bin src
  # cd src
  ```
Adding bonnie

◆ Build and install it:

```bash
# tar -zxvf bonnie++-1.03a.tgz
# cd bonnie++-1.03a
# ./configure --prefix=/share/apps/benchmarks/bonnie++
# make ; make install
```

◆ You can now run it on a compute node:

```bash
# ssh compute-0-0
# mkdir ~/output_files
# cd /share/apps/benchmarks/bonnie++/sbin/
# ./bonnie++ -s 100 -r 50 -u root -n 0 -f -d ~/output_files
```
Package bonnie as an RPM

- Checkout the Rocks development source tree

```
# cd /export
# mkdir src
# cd src
# cvs -d:pserver:anonymous@cvs.rocksclusters.org:/home/cvs/CVSROOT login
# cvs -d:pserver:anonymous@cvs.rocksclusters.org:/home/cvs/CVSROOT checkout rocks-devel
```
Create a Benchmark Roll

- Use the ‘template’ roll to populate a skeleton ‘benchmark’ roll

  # cd rocks/src/roll/
  # bin/make-roll-dir.py -n benchmark

- Create directory for bonnie

  # cd benchmark/src
  # mkdir bonnie
Create a Bonnie RPM

- **Get build files**
  
  ```
  # cd bonnie
  # cp ../benchmark/Makefile .
  # cp ../benchmark/version.mk .
  ```

- **Get the source**
  
  ```
  ```
Create a Bonnie RPM

- Update version.mk to match source
  - Change:
    
    NAME            = benchmark
    VERSION         = 1
    RELEASE         = 1
    TARBALL_POSTFIX = tgz

  - To:
    
    NAME            = bonnie++
    VERSION         = 1.03a
    RELEASE         = 1
    TARBALL_POSTFIX = tgz
Create a Bonnie RPM

- Build the RPM
  
  # make rpm

- You see lots of output
  
  The last line shows you where the resulting binary RPM is:

  Wrote: /export/src/rocks/src/roll/benchmark/RPMS/i386/bonnie++-1.03a-1.i386.rpm
Create a Bonnie RPM

◆ View the RPM contents

```
# rpm -qlp /export/src/rocks/src/roll/benchmark/RPMS/i386/bonnie++-1.03a-1.i386.rpm
```

◆ Which outputs:

```
/
/opt
/opt/benchmark
/opt/benchmark/bonnie++
/opt/benchmark/bonnie++/bin
/opt/benchmark/bonnie++/bin/bon_csv2html
/opt/benchmark/bonnie++/bin/bon_csv2txt
/opt/benchmark/bonnie++/man
/opt/benchmark/bonnie++/man/man1
/opt/benchmark/bonnie++/man/man1/bon_csv2html.1
/opt/benchmark/bonnie++/man/man1/bon_csv2txt.1
/opt/benchmark/bonnie++/man/man8
/opt/benchmark/bonnie++/man/man8/bonnie++.8
/opt/benchmark/bonnie++/man/man8/zcav.8
/opt/benchmark/bonnie++/sbin
/opt/benchmark/bonnie++/sbin/bonnie++
/opt/benchmark/bonnie++/sbin/zcav
```
Copy the bonnie++ RPM so rocks-dist Can Find It

- All packages are found under ‘/home/install’
- Put bonnie++ RPM package in /home/install/ contrib/4.1/<arch>/RPMS
  - Where <arch> is ‘i386’, ‘x86_64’ or ‘ia64’

  # cd /home/install/contrib/4.1/i386/RPMS
  # cp /export/src/rocks/src/roll/benchmark/RPMS/i386/bonnie++-1.03a-1.i386.rpm .
Extend the “Compute” XML Configuration File

- To add the package named “bonnie++”
  
  ```
  $ cd /home/install/site-profiles/4.1/nodes
  $ vi extend-compute.xml
  ```

- In ‘extend-compute.xml’, change the section:
  
  ```xml
  <package>strace</package>
  ```
  
  To:
  
  ```xml
  <package>strace</package>
  <package>bonnie++</package>
  ```
Extend the “Compute” XML Configuration File

- **Rebind the distro**
  - This copies ‘extend-compute.xml’ into /home/install/rocks-dist/…/build/nodes
    
    ```
    # cd /home/install
    # rocks-dist dist
    ```

- **Test the changes**
  - Generate a test kickstart file
    
    ```
    # dbreport kickstart compute-0-0 > /tmp/ks.cfg
    ```
  - You should see ‘bonnie++’ under the ‘%packages’ section
Extend the “Compute” XML Configuration File

- When you are satisfied with the changes, reinstall a compute node

  # shoot-node compute-0-0
Custom Partitioning
Default Frontend Partitioning

- 6 GB for /
  - Applications
  - Configuration files
  - Log files
- 1 GB swap
- Rest of first drive is /export
  - Home directories
  - Rocks distribution
Modifying Frontend Partitioning

- Can only change during frontend installation

- Note: must have ‘/export’
  - /export is automatically mounted by all compute nodes
Default Compute Node Partitioning

- 6 GB for / on first disk
- 1 GB for swap on first disk
- Remainder of first disk
  - Partitioned as “/state/partition1”
- All non-root partitions are saved over reinstalls
Changing Size of Root and Swap on a Compute Node

- If just want to change size of root and swap, only need to change two variables
- Create the file “extend-auto-partition.xml”

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

- Above the “<main>” section, add the two variables

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

- Above XML variables will create a 10 GB root partition and a 2 GB swap partition

- Rebind the distro (rocks-dist dist) and reinstall a compute node (shoot-node compute-0-0)
Specifying a New Partition Layout

- Only requirement is that ‘/’ is “big enough”
- Create the file “extend-auto-partition.xml”
  ```
  # cd /home/install/site-profiles/4.1/site-nodes/
  # cp skeleton.xml extend-auto-partition.xml
  ```
- In the “<main>” section, add (assumes disk name is ‘hda’):
  ```
  <main>
      <part> / --size 9000 --ondisk hda </part>
      <part> swap --size 1000 --ondisk hda </part>
      <part> /mydata --size 1 --grow --ondisk hda </part>
  </main>
  ```
- Rebind the distro (rocks-dist dist) and reinstall a compute node (shoot-node compute-0-0)
Specifying Software RAID

− Create the file “extend-auto-partition.xml”
  
  # cd /home/install/site-profiles/4.1/site-nodes/
  # cp skeleton.xml extend-auto-partition.xml

− In the “<main>” section, add:

  <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>

− Rebind the distro (rocks-dist dist) and reinstall a compute node (shoot-node compute-0-0)
Testing Changes

- **Use:**
  - “dbreport kickstart compute-0-0 > /tmp/ks.cfg”

- **Should not see any output**
  - That is, no error output

- **The file “/tmp/ks.cfg” should contain your changes**
  - Look for ‘part’ definitions towards the top of /tmp/ks.cfg
Configuring Additional Ethernet Interfaces
Configuring ‘eth1’

- If a compute node has a second ethernet NIC, use the command ‘add-extra-nic’ to assign it an IP address

```bash
# add-extra-nic --if=<interface> --ip=<ip address> --netmask=<netmask>\n    --gateway=<gateway> --name=<host name> <compute node>
```
Configuring ‘eth1’ - Example

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

◆ For compute-0-0, the above line sets the following values for ‘eth1’:

   ✍ IP address: 192.168.1.1
   ✍ Name for above IP address: fast-0-0
   ✍ Netmask: 255.255.255.0
   ✍ Gateway: 192.168.1.254
Configuring ‘eth1’

- To apply the change, reinstall the compute node

  # shoot-node compute-0-0