Linux.SYS-CON.com Cover Story: Rapid Cluster Deployment

After building a number of clusters from the ground up –including one that made it to the Top500 Supercomputer list – I decided to try a service that many vendors now offer – having a system racked and stacked at the factory then shipped to us. Such a service saves a huge amount of time, not to mention my back, not having to build the cluster and cable all the equipment together. I've been a fan of well-cabled systems and have found the quality control to be acceptable. The key component is the pre-build requirements and verification before the system is built. This will ensure the system shipped is what is expected when it arrives at your front door. There can still be a fair amount of cabling that has to be done once it arrives, if you have a multi-rack configuration, but it's usually limited to plugging in the system's power and public network.

Once this is done, the fun begins...
I've tried a few cluster distribution toolkits, and the one that works for me is the Rocks Cluster Distribution from the San Diego Supercomputing Center. I came across the package in a simple Google search in 2002 and was immediately sold on it. I use the term "sold" loosely since it's under an Open Source BSD-style license available for download and supported by a broad range of technical people who answer most questions on the Rocks user list. I've found support on the list to be better than most commercial distributions, but this may be because there are over 500 registered systems on the Rocks Register.

Here's how simple it is – insert the boot CD, complete a few screens worth of configuration data, and grab a coffee because it's a fairly simple base installation. The Rocks solution is extensible, with a mechanism for users and software vendors to ensure customizations are correctly installed on the system at setup. The mechanism is called a Roll.

The Roll typically consists of packages (RPMS/SRPMS/source) that have to be installed and scripts that are needed to ensure the packages are properly installed and distributed on the cluster. The Rocks team has extensive documentation for the Roll developer in the user manual.

Rocks 4.0.0 is a "cluster on a CD" set. That is it contains all the bits and configuration to build a cluster from "naked" hardware. The core OS bundled with Rocks is CentOS 4, which is a freely downloadable rebuild of Red Hat Enterprise Linux 4. As a side note, in Rocks CentOS 4 is encapsulated as the "OS Roll" and this OS Roll can be substituted with *any* Red Hat Enterprise Linux 4 rebuild (e.g., Scientific Linux) including the official bits from Red Hat. Rolls are used in Rocks to customize your cluster. For example, the HPC Roll contains cluster-specific packages, such as an MPI environment for developing and running parallel programs. Two other examples are the Ganglia Roll, which provides cluster-monitoring tools, and the Area51 Roll, which provides security tools such as Tripwire and chkrootkit.

The Software

The core OS we used for the cluster in this article is CentOS 4.0 and the rolls we used to customize the cluster to our needs were the Compute Roll and the PBS Roll from University of Tromso in Norway.

The Hardware

- **1** – Front-end node – a Dell PowerEdge 2850 with dual 3.6GHz Intel Xeon EM64T processors and 4GB RAM
- **48** – Compute nodes – Dell PowerEdge SC 1425s with dual 3.4GHz Intel Xeon EM64T processors, 2GB RAM and a Topspin PCI-X Infiniband HCA card
- **1** – Topspin 270 Infiniband chassis with modules
- **4** – Dell PowerConnect 5324 Gigabit Ethernet switches
- **1** – Panasas Storage Cluster with one DirectorBlade and 10 StorageBlades
- **2** – Dell 19-inch racks

Start the build process ***time 0:00:00***
Setting up the front-end:
– Insert Compute Roll and boot the system
– Select hpc, kernel, ganglia, base, java, and area51 as the rolls to install
– Select "Yes" for additional roll
– Insert CentOS disk 1
– Select "Yes" for additional roll
– Insert CentOS disk 2
– Select "Yes" for additional roll
– Insert PBS roll
– Select "No" for additional rolls
– Input data on the configuration screen (e.g., fully qualified domain name, root password, IP addresses)
– Select "Disk Druid" to create partitions
– Create/partition ext3 64GB
– Create swap partition 4GB
– Create/export partition 64GB
– Insert CDs as requested to merge them into the distribution

The most important step...grab a mocha and enjoy it while the install runs.

After the-front end installation completes, the site-specific customization of the front-end starts. The base installation of CentOS 4.0 x86_64 has the 2.6.9-5.0.5.ELsmp kernel and we need the 2.6.9-11.ELsmp for many of the packages that will be included with our cluster. Below we'll describe how we do this key upgrade then continue with many package and mount point customizations.

Customization of the front-end:
The first step is to apply the updated kernel packages:

- **# rpm** -ivh kernel-smp.2.6.9-11.EL.x86_64.rpm
- **# rpm** -ivh kernel-smp-devel-2.6.9-11.EL.x86_64.rpm
- **# rpm** -ivh kernel-sourcecode-2.6.9-11.EL.x86_64.rpm

I always check /boot/grub/grub.conf to be sure the system is booting from the proper kernel after an update.

Then apply an RPM to resolve a known (to us) library issue:

- **# rpm** -ivh compat-libstdc++-33-3.2.3-47.3.i386.rpm

Prepare for Panasas Storage Cluster and Topspin integration on the front-end:

- **# rpm** -ivh panfs-2.6.9-11.EM-mp-2.2.3-166499.27.rhel_4_amd64.rpm
- **# rpm** -ivh topspin-ib-rhel4-3.1.0-113. x86_64.rpm

- **# rpm** -ivh topspin-ib-mod-rhel4-2.6.9-11.ELsmp-3.1.0-113.x86_64.rpm

  Time for a break. ***time 1:05:00***

I need to complete the setup of the disks on the front-end because there are two RAID volumes and Rocks only configures the first disk (the boot disk) on the front-end leaving the other disks untouched.

Create a second partition for applications:

fdisk /dev/sdb
(400GB single partition on our system)

Create the file system and mount point:

mkfs -t ext3 /dev/sdb1
mkdir /space

Modify /etc/fstab to include the mount point then mount it:

mount /space

Now let's start adding some of the goods...

- Install Portland Group compilers in /space/apps/pgi/
- Install Intel 9.0 compilers in /space/apps/pgi
- Install OSU MVAPICH 0.95 in /space/apps/mvapich
- Build version of MVAPICH for intel/gnu/pgi
- Install our own version of Python in /space/apps/python64
- Install f2py, Numeric, pyMPI built against our vanilla version of python64

The Rocks solution uses a simple XML framework to provide a programmatic way in which to apply site-specific customizations to compute nodes. While this requires a small learning curve regarding the XML syntax, once you make the transition from "administering" your cluster to "programming" your cluster, you'll find that writing programs (in the form of scripts) are a powerful way in which to ensure your site customizations are consistent across all compute nodes. The following describes how we used the XML framework to apply our customizations.

Put the package you want to add in:

/home/install/contrib./4.0.0/arch/RPMS

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

/home/install/site-profiles/4.0.0/nodes
cp skeleton.xml extend-compute.xml

Inside the extend-compute.xml, add the package name with a <package> tag.

Insert any post installation scripting in the <post> section.

This is a truncated version of an extend-compute.xml script we use:

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

```
<!-- There may be as many packages as needed here. -->
<package> kernel-smp </package>
<package> kernel-smp-devel </package>
<package> kernel-sourcecode </package>
<package> topspin-ib-rhel4 </package>
<package> topspin-ib-mod-rhel4 </package>
<package> compat-libstdc++-33 </package>
<package> panfs </package>
<package> F2PY-fix </package>
<post>

<file name="/etc/profile.d/topspinvars.sh">
if ! echo ${LD_LIBRARY_PATH} | grep -q /usr/local/topspin/mpi/mpich/lib64;

then

LD_LIBRARY_PATH=/usr/local/topspin/mpi/mpich/lib64:${LD_LIBRARY_PATH}
fi
if ! echo ${LD_LIBRARY_PATH} | grep -q /usr/local/topspin/lib64 ; then
LD_LIBRARY_PATH=/usr/local/topspin/lib64:${LD_LIBRARY_PATH}
fi
</file>

<file name="/etc/profile.d/topspinvars.csh">
if ( /usr/local/topspin/mpi/mpich/lib64 !~ "${LD_LIBRARY_PATH}" ) then
set LD_LIBRARY_PATH = ( /usr/local/topspin/mpi/mpich/lib64$LD_LIBRARY_PATH )
endif
if ( /usr/local/topspin/lib64 !~ "${LD_LIBRARY_PATH}" ) then
set LD_LIBRARY_PATH = ( /usr/local/topspin/lib64 $LD_LIBRARY_PATH )
endif
</file>

<!-- Setup for Portland Group Compilers -->
<file name="/etc/profile.d/pgi-path.sh">
LM_LICENSE_FILE=/space/apps/pgi/license.dat
export LM_LICENSE_FILE
export PGI=/space/apps/pgi
if ! echo ${PATH} | grep -q /space/apps/pgi/linux86-64/6.0/bin ;

then

PATH=/space/apps/pgi/linux86-64/6.0/bin:${PATH}
fi
</file>

<file name="/etc/profile.d/pgi-path.csh">
setenv PGI /space/apps/pgi
setenv LM_LICENSE_FILE $PGI/license.dat
if ( /space/apps/pgi/linux86-64/6.0/bin !~ "${path}" ) then
set path = ( /space/apps/pgi/linux86-64/6.0/bin $path )
endif
</file>
```

</kickstart>

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

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

Time for a break. ***time 2:30:00

Switch Configurations
- Setup IP information
- Enable "fast link" on all ports
- Link aggregation (four 1GB aggregated links between switches, four 1GB aggregated link for the Panasas Installation)
- Setup configuration on Topspin 270 switch

Insert the Compute Nodes
- Invoke 'insert-ethers' on the front-nd node and select 'compute node' for the appliance type (insert-ethers is used to discover compute nodes and ensure they get the proper configuration)

I chose to make modifications to each compute node for remote management access for remote boot/power cycle, etc. This added two minutes to each node's install time. I also like to make sure each node is discovered in the order in which I power them on (left to right, bottom to top in the racks). This process takes more time, but it saves a huge amount of time when tracking down hardware failures. It also makes labeling them following the compute node installation easy, with the utility provided by Rocks for creating labels with identification information. (This little tool really shows that the distribution was developed by people who manage clusters.) 48 nodes x 3.5 minutes per node = 2 hours 48 minutes for this process due to my requirements. It could be far less time depending on your needs.

Time for a break. ***time 5:40:00***

Time for Panasas Integration
The Panasas Storage Cluster arrived in three boxes on one pallet. From the time I clipped the first band on the pallet to having the system fully operational was only 1 hour 55 minutes. Here's how the process went.

First off, we unpacked the equipment and identified all components and parts. I found a location in the rack, set up the mounting hardware then mounted the chassis. I loaded the chassis with one DirectorBlade and 10 StorageBlades, network, battery and power supply modules, and connected four 1GB Ethernet links to the link aggregation point on the switch.

Now it's time to configure the software side of it.

- Connect a console cable from my laptop to the serial port on the DirectorBlade
- Log in as admin
- Configure the system via the pancli (a modified Panasas command line interface on the Unix-based operating system) with IP information

I was then able to connect to the system via the Web GUI to complete some additional configuration:

- Customer ID
- System name (required for global file system configuration)
- IP range for DHCP pool for Director Blade and StorageBlades

– Timezone

This minimum configuration is all that's required for the system to function as a basic network attached storage (NAS) system. Additional modifications can include DNS, NIS, NTP, NFS, CIFS, and SMTP servers (notifications). You can also modify volume and quota information. There are some modifications to make to the useradd scripts based on setting choices and auto-mount options if you want user home directories located on the Panasas Storage Cluster, one way in which we use the unit.

At this point, we can run the following on the front-end:

```
# config_panfs -r [ip_address_of_shelf]
```

Restart the PanFS service on all compute nodes:

```
# cluster-fork 'service panfs restart'
```

(We included an IP in the extend-compute.xml for the shelf in advance.)

The cluster is ready to run jobs and you can read/write data to the shelf.

End of the basic installation for the cluster ***time 7:35:00***

So, you've seen the basics of setting up a cluster. The thing to remember is that Rocks is the middleware component that gives you the tools to do anything you want with the system, within reason. We have MANY customizations we will do on the cluster to support a variety of user requests. One example is enabling MPI-based applications to leverage Panasas-specific parallel I/O extensions.

Panasas offers an SDK that includes modifications for the MPI component implementing parallel I/O called ROMIO. ROMIO implements the MPI-IO layer in MPICH, one of the more popular MPI implementations. In the Panasas SDK is a patch to apply to MPICH (we run MVAPICH from OSU, based on MPICH and MVICH) that lets Panasas-specific features function.

Required items
– Panasas DirectFLOW client software that got installed during the initial cluster configuration
– Panasas SDK
– Source code for a ROMIO-based MPI implementation

Unpack the source and apply the patch:

```
# tar zxvf mvapich.tgz
# cd mpich
# patch NP1 < ~/romio.patch
```

Configure, make, and install MVAPICH including the following options (only PanFS flags shown)

```
# export CFLAGS=" -D_FILE_OFFSET_BITS=64 -D_LARGEFILE_SOURCE -I ¥
# /opt/panfs/include/ -D_linux_=1"
# ./configure --with-romio --file-system=ufs+nfs+panfs
# make
# make install
```

The PanFS patch implements several MPI hints to specify data storage layout and/or concurrent write access when opening a file on a Panasas Storage Cluster.

Here's a sample PanFS-specific hint – panfs_concurrent_write – If the value of this hint is "1" open the file in concurrent write mode. If the value is "0" or the hint is missing, open the file in standard (non-concurrent write) mode.

So, now you have a much better idea of what's possible with the configuration of the system. Let's try a simple compile and run.

We use a simple program named "bounce" as the first benchmark for every system we build. This program times blocking send/received and reports the latency and bandwidth of the communication system. It's a great tool for us as it's small, portable, and tests our MPI F90 capability, something important to us. Here's how simple it is to use.

Compile:

```
$ mpif90 -o bounce bounce.f
```

Run:

```
$ qsub -I -lnodes=4:ppn=2
waiting for job.x to start
$ mpirun -ssh -np 8 -hostfile $PBS_NODEFILE ./bounce >& ¥
$PBS_WORKDIR/log.bounce
$ tail log.bounce
```

Start adding users:

```
# useradd alice
# passwd alice
```

What I typically do at this point is create an e-mail distribution list for the cluster for announcements of additions, changes, and maintenance windows. I'll include simple instructions on the compiling, location, and naming conventions used for the compilers and a sample PBS script. This usually answers most of the questions we'll have when first logging onto the cluster.

Here is a sample PBS script:

```
#!/bin/bash
#PBS -N BOUNCE
#PBS -e BOUNCE.err
#PBS -o BOUNCE.out
#PBS -m aeb
#PBS -M alice@cluster.com
#PBS -l nodes=8:ppn=1
#PBS -l walltime=30:00:00

PBS_O_WORKDIR='/home/alice/bounce'
export PBS_O_WORKDIR

### ---------------------------------------
### BEGINNING OF EXECUTION
### ---------------------------------------

echo The master node of this job is `hostname`
echo The working directory is `echo $PBS_O_WORKDIR`
echo This job runs on the following nodes:
```

```
echo `cat $PBS_NODEFILE`

### End of information preamble
cd $PBS_O_WORKDIR
cmd="/apps/bin/mpirun -ssh -np 8 -hostfile $PBS_NODEFILE
$PBS_O_WORKDIR/bounce"
echo "running mpirun with: $cmd in directory "`pwd`
$cmd >& $PBS_O_WORKDIR/log.bounce
```

Why did I select the solutions above? The Rocks Cluster Distribution is widely adopted with over 500 locations running the cluster distribution package. Many companies are offering fee-based support for the package as part of their offering. Dell even ships clusters with Rocks pre-installed.

Panasas was a blessing. Our researchers were finding it difficult to function on one system with 164 processors and a RAID SATA Disk Linux NFS server solution. I followed all the recommended solutions for optimizing NFS on Linux servers, but it was still easy to saturate. The first thing I tried was splitting users across multiple NFS servers, but this quickly became an issue since the users just scaled up their I/O traffic to run faster, which eventually created the same bottleneck – the NFS server.

I found Panasas through a contact at the AMD Developer Center who manages their clusters and decided to give it a try. I was impressed at the ease of installation on our cluster and the results I was able to achieve. I decided to run a benchmark on the Panasas system against our NFS servers. "bonnie++" was the tool I decided to use, and I also decided to put the executable in the shared location and run the tests through the queue.

Here's what I did:

Create the PBS scripts to run the tests–(following the example script above – truncated version)

```
#!/bin/bash
#PBS -l nodes=1:ppn=2
cmd="/home/tools/bonnie++/sbin/bonnie++ -s 8000 -n 0 -f -d
/home/alice-nfs-directory/bonnie"
$cmd >& $PBS_O_WORKDIR/Log.d/run.nfs/log.bonnie.nfs.$PBS_JOBID

#!/bin/bash
#PBS -l nodes=1:ppn=2
cmd="/home/tools/bonnie++/sbin/bonnie++ -s 8000 -n 0 -f -d
/home/alice-panfs-directory/bonnie"
$cmd >& $PBS_O_WORKDIR/Log.d/run.panfs/log.bonnie.panfs.$PBS_JOBID
Run the tests-
Results from eight instances of the NFS script-
17.80MB/sec for concurrent write using NFS with eight dual-processor jobs
36.97MB/sec during read process

Results from eight instances of the PanFS script-
154MB/sec for concurrent write using PanFS with eight dual-processor jobs
190MB/sec during read process
```

The numbers were excellent, so I decided to scale it up:

Results from 16 instances of the NFS script-

20.59MB/sec for concurrent write using NFS with 16 dual-processor jobs
27.41MB/sec for during read process

Results from 16 instances of the PanFS script-

187MB/sec for concurrent write using PanFS with 16 dual-processor jobs
405MB/sec during read process

It's pretty much a no-brainer here. We moved to the new system. Having the Rocks Clusters platform let us integrate the Panasas solution in just under two hours.

Final Thoughts

First off, this may sound as if it's very simple to set up and manage a cluster. While the description above looks to be straightforward, the real work starts once your first user logs onto the front-end. I have a group of power users that I've unleashed on systems in the past and I'm comfortable with their ability to compile and run known good code and provide me with clear information related to the issues they encounter on the system. I wouldn't advise setting up a system and turning everyone loose on it since you might spend the next few weeks trying to resolve issues that may or may not be cluster-related.

This is where the Rocks discussion list comes in handy. Although you may have certain configuration and code-related differences with other Rocks installations, you share the same common thread with everyone, the same tools to troubleshoot and apply changes.

You'll also find your cluster may not be as different as you think it is from other clusters. I've been able to manage a large number of clusters without having the overhead of many system administrators primarily because of interactions with people on the Rocks list. By far the easiest system I've installed was a Penguin AMD Opteron cluster running Infinicon (now Silverstorm) Infiniband equipment. The team at the AMD Developer Center runs Rocks on the multiple clusters they use to provide service for developers running on their test systems and then rebuild their clusters each time a new client requests system time. They have been an excellent resource through the Rocks list with information about cluster building and day-to-day management.

About the Science

Researchers in the mechanical engineering and aeronautics & astronautics departments at Stanford University are involved in several research programs that require large-scale massively parallel computing resources to carry out first-of-a-kind simulations. The cluster is being used to compute the details of the flow and acoustic fields created by helicopters in forward flight. For this purpose two major simulation codes, SUmb and CDP, are run simultaneously to compute the flow in separate areas of the domain: SUmb computes the flow in the region near the surface of the blades where compressibility and viscous effects are dominant, and CDP resolves the wake portion where the identification of the strength and location of the trailing vortices is of fundamental importance.

SUmb (Stanford University multi-block) and CDP (named for the late Charles David Pierce) are both mas-|sively parallel flow solvers developed at Stanford under the sponsorship of the Department of Energy's ASC program. SUmb can be used to solve for the compressible flow in many applications including, but are not limited to, jet engines, subsonic and supersonic aircraft, helicopters, launch vehicles, space and re-entry vehicles, and a host of other research applications.

SUmb uses a multi-block structured meshing approach. The mesh is decomposed into a number of pieces that are distributed to each of the processors in a calculation and the Message Passing Interface (MPI) standard is used to communicate between processors using a high-bandwidth, low-latency network (InfiniBand in the case of our Rocks Cluster).

CDP uses a fully unstructured grid approach to allow more flexibility in concentrating the grid points

in regions of interest. The code was developed to simulate the multiphase reacting flow in jet engine combustors, but can be applied more generally to simulate a variety of flows where important flow structure persists for a relatively long time, such as the trailing vortices generated by the helicopter's blade tips.

For coupled simulations where SUmb needs to interact with CDP, the codes use a Python-based interface to simplify access to the data structures, while allowing the core portions of the solution to be carried out using highly optimized compiled languages. Both codes have been routinely run on thousands of processors and have been readied for computations on large parallel computers such as BlueGene/L, which is expected to reach a total of about 130,000 processors.

Resources

- www.rocksclusters.org/rocks-register
- http://npaci-rocks-discuss@sdsc.edu
- www.centos.org
- www.scientificlinux.org
- www-unix.mcs.anl.gov/mpi
- http://ganglia.sourceforge.net
- http://sourceforge.net/projects/tripwire
- www.chkrootkit.org
- http://uit.no/itavd/HPC-Rocks-PBS-Roll/
- www.panasas.com
- www-unix.mcs.anl.gov/romio
- http://nowlab.cse.ohio-state.edu/projects/mpi-iba
- www-unix.mcs.anl.gov/mpi/mpich
- http://old-www.nersc.gov/research/FTG/via/download_info.html
- www.penguincomputing.com
- www.silverstorm.com
- http://developer.amd.com/
- http://icme.stanford.edu
- http://sourceforge.net/projects/subounce