Mendel at NERSC:
Multiple Workloads on a Single Linux Cluster

Larry Pezzaglia
NERSC Computational Systems Group
lmpezzaglia@lbl.gov
CUG 2013 (May 9, 2013)
Located at LBNL, NERSC is the production computing facility for the US DOE Office of Science.

- NERSC serves a large population of ~5000 users, ~400 projects, and ~500 codes.

Focus is on “unique” resources:

- Expert computing and other services
- 24x7 monitoring
- High-end computing and storage systems

NERSC is known for:

- Excellent services and user support
- Diverse workload
NERSC Systems

- **Hopper**: Cray XE6, 1.28 PFLOPS
- **Edison**: Cray XC30, > 2 PFLOPS once installation is complete
- Three x86_64 midrange computational systems:
  - **Carver**: ~1000 node iDataPlex; mixed parallel and serial workload; Scientific Linux (SL) 5.5; TORQUE+Moab
  - **Genepool**: ~400 node commodity cluster providing computational resources to the DOE JGI (Joint Genome Institute). Mixed parallel and serial workload; Debian 6; Univa Grid Engine (UGE)
  - **PDSF**: ~200 node commodity cluster for High Energy Physics and Nuclear Physics; exclusively serial workload; SL 6.2 and 5.3 environments; UGE
Midrange Expansion

- Each midrange system needed expanded computational capacity
- Instead of expanding each system individually, NERSC elected to deploy a single new hardware platform ("Mendel") to handle:
  - Jobs from the "parent systems" (PDSF, Genepool, and Carver)
  - Support services (NX and MongoDB)
- Groups of Mendel nodes are assigned to a parent system
  - These nodes run a batch execution daemon that integrates with the parent batch system
- Expansion experience must be seamless to users:
  - No required recompilation of code (recompilation can be recommended)
Approaches
Multi-image Approach

- One option: Boot Mendel nodes into modified parent system images.
- Advantage: simple boot process
- Disadvantage: Many images would be required:
  - Multiple images for each parent compute system (compute and login), plus images for NX, MongoDB, and Mendel service nodes
  - Must keep every image in sync with system policy (e.g., GPFS/OFED/kernel versions) and site policy (e.g., security updates):
    - Every change must be applied to every image
    - Every image is different (e.g., SL5 vs SL6 vs Debian)
    - All system scripts, practices, and operational procedures must support every image
- This approach does not scale sufficiently from a maintainability standpoint
NERSC Approach

- A layered model requiring only one unified boot image on top of a scalable and modular hardware platform
- Parent system policy is applied at boot time
- xCAT (eXtreme Cloud Management Toolkit) handles node provisioning and management
- Cfengine3 handles configuration management
- The key component is CHOS, a utility developed at NERSC in 2004 to support multiple Linux environments on a single Linux system
  - Rich computing environments for users separated from the base OS
  - PAM and batch system integration provide a seamless user experience
# The Layered Model

<table>
<thead>
<tr>
<th>User Applications</th>
<th>PDSF SL 6.2 Apps</th>
<th>PDSF SL 5.3 Apps</th>
<th>Genepool Debian 6 Apps</th>
<th>Genepool Debian 6 Logins</th>
<th>Carver SL 5.5 Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHOS</td>
<td>PDSF sl62 CHOS</td>
<td>PDSF sl53 CHOS</td>
<td>Genepool Compute CHOS</td>
<td>Genepool Login CHOS</td>
<td>Carver Compute CHOS</td>
</tr>
<tr>
<td>Boot-time</td>
<td>Genepool UGE</td>
<td>Genepool Cfengine Policy</td>
<td>Genepool xCAT Policy</td>
<td>Genepool Add-ons</td>
<td>Carver TORQUE</td>
</tr>
<tr>
<td>Differentiation</td>
<td>PDSF CF engine Policy</td>
<td>PDSF xCAT Policy</td>
<td>Genepool Add-ons</td>
<td>Carver xCAT Policy</td>
<td>Carver Add-ons</td>
</tr>
<tr>
<td>Base OS</td>
<td>PDSF Add-ons</td>
<td>Genepool Add-ons</td>
<td>Genepool Add-ons</td>
<td>Carver Add-ons</td>
<td></td>
</tr>
<tr>
<td>Hardware/Network</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unified Mendel Base OS</td>
<td>Unified Mendel Hardware Platform</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Implementation
Hardware

- Vendor: Cray Cluster Solutions (formerly Appro)
  - Scalable Unit expansion model
- FDR InfiniBand interconnect with Mellanox SX6518 and SX6036 switches
- Compute nodes are half-width Intel servers
  - S2600JF or S2600WP boards with on-board FDR IB
  - Dual 8-core Sandy Bridge Xeon E5-2670
  - Multiple 3.5” SAS disk bays
- Power and airflow: ~26kW and ~450 CFM per compute rack
- Dedicated 1GbE management network
  - Provisioning and administration
  - Sideband IPMI (on separate tagged VLAN)
Base OS

- Need a Linux platform that will support IBM GPFS and Mellanox OFED
  - This necessitates a “full-featured” glibc-based distribution
  - Scientific Linux 6 was chosen for its quality, ubiquity, flexibility, and long support lifecycle
- Boot image is managed with NERSC’s `image_mgr`, which integrates existing open-source tools to provide a disciplined image building interface
  - Wraps xCAT genimage and packimage utilities
  - add-on framework for adding software at boot-time
  - Automated versioning with FSVS
    - Like SVN, but handles special files (e.g., device nodes)
    - Easy to revert changes and determine what changed between any two revisions
    - http://fsvs.tigris.org/
Cfengine rules are preferred
  - They apply and *maintain* policy (promises)
  - Easier than shell scripts for multiple sysadmins to understand and maintain

xCAT postscripts
  - Mounting local and remote filesystems
  - Changing IP configuration
  - Checking that BIOS/firmware settings and disk partitioning match parent system policy

image_mgr add-ons add software packages at boot time
  - Essentially, each add-on is a cpio.gz file, {pre-,post-}install scripts, and a MANIFEST file
CHOS

- CHOS provides the simplicity of a “chroot” environment, but adds important features.
  - Users can manually change environments
  - PAM and Batch system integration
    - PAM integration CHOSes a user into the right environment upon login
    - Batch system integration: SGE/UGE (starter_method) and TORQUE+Moab/Maui (preexec or job_starter)
  - All user logins and jobs are chroot’ed into /chos/, a special directory managed by sysadmins
  - Enabling feature is a /proc/chos/link contextual symlink managed by the CHOS kernel module
- Proven piece of software: in production use on PDSF (exclusively serial workload) since 2004.
/chos/

/chos/ when CHOS is not set:

/chos/bin → /proc/chos/link/bin → /bin/
/chos/etc → /proc/chos/link/etc → /etc/
/chos/lib → /proc/chos/link/lib → /lib/
/chos/usr → /proc/chos/link/usr → /usr/
/chos/proc → /local/proc/
/chos/tmp → /local/tmp/
/chos/var → /local/var/
/chos/dev/ # Common device nodes
/chos/gpfs/ # Mountpoint for a shared filesystem
/chos/local/ # Mountpoint for the real root tree
/chos/

/chos/ when CHOS is sl5:

/chos/bin → /proc/chos/link/bin → /os/sl5/bin/
/chos/etc → /proc/chos/link/etc → /os/sl5/etc/
/chos/lib → /proc/chos/link/lib → /os/sl5/lib/
/chos/usr → /proc/chos/link/usr → /os/sl5/usr/
/chos/proc → /local/proc/
/chos/tmp → /local/tmp/
/chos/var → /local/var/
/chos/dev/  # Common device nodes
/chos/gpfs/  # Mountpoint for a shared filesystem
/chos/local/ # Mountpoint for the real root tree
/chos/

/chos/ when CHOS is deb6:

/chos/bin → /proc/chos/link/bin → /os/deb6/bin/
/chos/etc → /proc/chos/link/etc → /os/deb6/etc/
/chos/lib → /proc/chos/link/lib → /os/deb6/lib/
/chos/usr → /proc/chos/link/usr → /os/deb6/usr/
/chos/proc → /local/proc/
/chos/tmp → /local/tmp/
/chos/var → /local/var/
/chos/dev/ # Common device nodes
/chos/gpfs/ # Mountpoint for a shared filesystem
/chos/local/ # Mountpoint for the real root tree
CHOS Challenges

- CHOS starter_method for UGE enhanced to handle complex qsub invocations with extensive command-line arguments (e.g., shell redirection characters)
- UGE qlogin does not use the starter_method. Reimplemented qlogin in terms of qrsh
- TORQUE job_starter was only used for the launch of the first process of a job, not for subsequent processes through Task Manager (TM)
  - All processes need to run inside the CHOS environment
  - NERSC developed a patch to pbs_mom to use the job_starter for processes spawned through TM
  - Patch accepted upstream and is in 4.1-dev branch
Base OS Image Management
We needed an alternative to “traditional” image management:

1. genimage (xCAT image generation)
2. chroot...vi...yum
3. packimage (xCAT boot preparation)
4. Repeat steps 2 and 3 as needed

The traditional approach leaves sysadmins without a good understanding of how the image has changed over time.

- Burden is on sysadmin to log all changes
- No way to exhaustively track or roll back changes
- No programmatic way to reproduce image from scratch
New approach: rebuild the image from scratch every time it is changed
  - `image_mgr` makes this feasible
  - We modify the `image_mgr` script, not the image

Standardized interface for image creation, manipulation, analysis, and rollback.

Automates image rebuilds from original RPMs

Images are versioned in a FSVS repository

“release tag” model for switching the production image
FSVS layout

The root directory of the repository

```
/                      ...
netboot/
  SL6.3/                OS version
  x86_64/               Architecture
  mendel-core.prod/     Image name

tags/
  2013-03-01-14-13-45-RELEASE-by-user/
    rootimg/
    add-ons/
      kvm/
      build-info/
        image_mgr.sh       The build script
        stats             Build statistics
  2013-02-27-11-10-02-RELEASE-by-user2/

trunk/
```
image_mngr supports several subcommands: **create**, **tag**, **list-tags**, and **pack**

- **create**: Build a new image and commit it to trunk/ (uses xCAT genimage and FSVS):

  ```
  # image_mngr create -p mendel-core.prod -o SL6.3 -a x86_64 -m "Test build" -u user1
  ```

- **tag**: Create a new SVN tag of trunk/ at the current revision, marking it as a potential production release

  ```
  # image_mngr tag -p mendel-core.prod -o SL6.3 -a x86_64 -u user1
  ```
list-tags: List all tags

```
# image_mgr list-tags -p mendel-core.prod -o SL6.3 -a x86_64
2013-03-01-14-13-45-RELEASE-by-user1
2013-02-27-11-10-02-RELEASE-by-user2
...
```

pack: Pack a tag as the production image (uses xCAT packimage)

```
# image_mgr pack -p mendel-core.prod -o SL6.3 -a x86_64 -t 2013-03-01-14-13-45-RELEASE-by-user1
```
Feedback for CCS
Several areas for improvement. CCS is actively working with NERSC to improve.

- Hardware supply chain issues
  - Delays getting parts from upstream vendor
- Proper cabling is **essential** when hundreds of cables are involved. We need to be able to service all equipment.
- **24x7 really means 24x7**
  - NERSC users work around the clock, weekends, and holidays
  - The system is never “down for the weekend”
  - **Any** outage, planned or unplanned, is severely disruptive to our users
  - We need detailed timelines for all work requiring downtimes
Conclusion
Acknowledgements

- **Doug Jacobsen**
  - Extensive Genepool starter_method and qlogin changes

- **Nick Cardo and Iwona Sakrejda**
  - Constructive feedback on the image_mgr utility

- **Shane Canon**
  - Original CHOS developer. Provided significant guidance for the Mendel CHOS deployment

- **Zhengji Zhao**
  - Early software tests on the Mendel platform.

- **Brent Draney, Damian Hazen, Jason Lee**
  - Integration of Mendel into the NERSC network

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.
Performance Data

Per-node HEPSPEC06 scores on SL6 on dual Xeon E5-2670 servers with 64 GB RAM.

NAMD STMV benchmark (1,066,628 atoms, periodic, PME) on dual Xeon E5-2670, 128 GB RAM.
Data provided by Zhengji Zhao, NERSC User Services Group
Additional Resources

- FSVS: http://fsvs.sf.net/
- xCAT: http://xcat.sf.net/
- Original CHOS paper:
  - http://indico.cern.ch/getFile.py/access?contribId=476&sessionId=10&resId=1&materialId=paper&confId=0
- 2012 HEPiX presentation about CHOS on PDSF:
  - http://www.nersc.gov/assets/pubs_presos/cho.pdf
- CHOS GitHub repository: https://github.com/scanon/chos/
- PDSF CHOS User documentation:
Conclusion

- The layered Mendel combined cluster model integrates a scalable hardware platform, xCAT, Cfengine, CHOS, and image_mgr to seamlessly support diverse workloads from multiple “parent” computational systems and support servers.
- Nodes can be easily reassigned to different parent systems.
- Separation between the user and sysadmin environments, which can each be architected exclusively for their intended uses.
- While this approach introduces additional complexity, it results in an incredibly flexible and maintainable system.