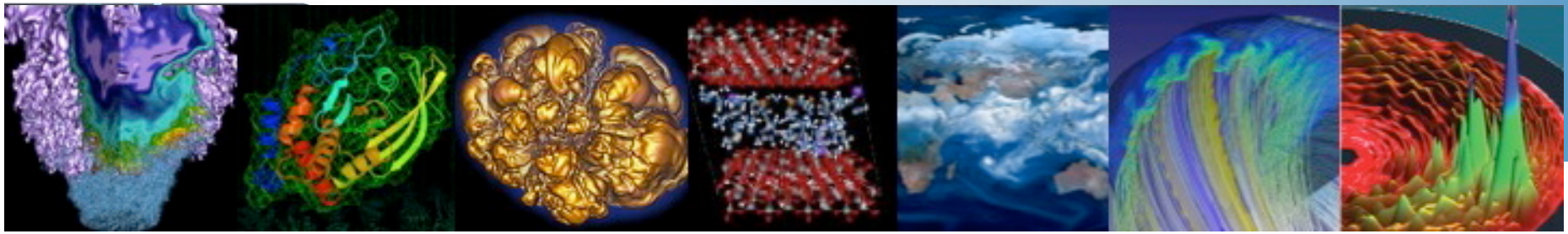

Crossing the Rhine – Moving to CLE 6.0 System Management

Tina Butler

NERSC

05/10/2016

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory

Setting the Stage

- NERSC is DOE's largest open science computing center
  - 5000+ users
  - 700+ applications
- Currently 3 Cray systems on the floor
  - Mendel CS300
  - Edison XC-30
  - Cori XC-40
- NERSC Global Filesystem (NGF)
  - Center-wide GPFS-based filesystem instances
  - Provide persistent home and project space

CLE 6.0 - Why upgrade now?

- CLE 6.0 UP00 is limited availability (LA)
  - Only supported for new installs
  - Required for KNL
  - UP01 GA promised for 3/16
- NERSC's Cori Phase 1
  - Delivered mid 2015
  - 12 cabinets of Haswell and DataWarp
  - Brought up under CLE 5.2 UP04
    - DataWarp software not ready for CLE 6.0 at time of Cori P1 delivery
    - Cori P1 focused on data intensive workload

Why upgrade now, cont.

- Cori Phase 2
  - 52 cabinets of KNL and DataWarp
  - KNL requires CLE 6.0
  - Delivery mid 2016
- Decision to upgrade Cori P1 before delivery of next phase to minimize downtime
  - Upgrade Cori TDS first to develop knowledge and skills to minimize downtime
- Early exposure to CLE 6.0 UP00 through ACES collaboration with Sandia and Los Alamos
- Increased pressure to start with UP00 when UP01 release slipped to June

The trouble with UP00

- **Limited availability, so not supported at usual level**
  - **Not allowed to open bugs**
  - **No released documentation**
  - **Many features/improvements delayed until UP01**
- **Bare-metal install**
  - **No migration path from CLE 5.2**
  - **SMW and bootraid completely reformatted**
  - **No tools for gathering configuration from existing system**

Cray Management System Basics

- Separate installation and configuration of images for service, login and compute nodes
  - Allows prescriptive definition of nodes configurations
  - Allows local custom node types
  - No shared root
- Update to newer, more widely used and known tools
  - Ansible
  - Open Stack
- Centralize configuration and installation for both internal and external node types
- Very different point-of-view from previous CLE management tools

NERSC Local Customizations

- **Network configuration method not standard**
  - Interfaces not configured through specialization
  - Use route and host input files to set interface configs and routes
  - Bonded interfaces
- **GPFS-based NERSC Global Filesystems**
  - Center-wide filesystems served to compute and service nodes through DVS

Network Configuration

- Cori has 32 RSIP nodes, 32 DVS nodes, 2 network nodes, 130 LNET nodes
- Not yet well supported in CMS
  - Laborious and error-prone using cfgset
  - Doesn't support bonded interfaces
- Local Cray staff wrote a configuration scraper to gather network and service information
  - Used to generate 6.0 config
  - Bonding accomplished through ifcfg files uploaded via simple_sync, and scripts run by ansible plays
- Still a work in progress

GPFS at NERSC

- **NERSC Global Filesystem (NGF)**
  - **Actually 8 different GPFS-based filesystem instances that are mounted on all NERSC production systems**
  - **Supported through GPFS remote clustering with direct client mounts on DVS server and eLogin nodes**
  - **DVS server nodes serve the NGF filesystems to compute and selected service nodes**
- **Only the two latest releases of GPFS are supported with SLES12**
  - **4.1.1 and 4.2.0**
  - **NERSC GPFS owning cluster being upgraded to 4.1.1**

How to install GPFS with R/R?

- GPFS installation and upgrades need to be maintainable and sustainable
  - Able to do manual workarounds, but not sustainable
  - Expect changes in UP01 that will improve the process
- GPFS install model not a natural fit with CLE 6.0 install philosophy
  - Requires base RPM initial install, update install for any fix levels (PTFs)
  - Generation of a personality layer RPM on a booted client node whenever the kernel changes
  - Remote cluster configuration must persist

Installing GPFS

- Create local repos for GPFS RPMs
  - gpfs-4.1-base - base gpfs 4.1.0 release
  - Gpfs-4.1.1 – gpfs updates 4.1.1.0, 4.1.1.4
- Create package collections with gpfs-base and gpfs-updates.
  - Cannot rely on dependencies since base requires an initial install
  - Pkgcoll syntax is quite picky
- Clone base service node recipe to modify for GPFS
  - recipe create –clone service_cle_6.0up00_sles_12_x86-64_ari nersc_gpfs_client
  - Local recipe is written in /etc/opt/cray/imps/image.recipes.d/ image_recipes.local.json
  - Add gpfs base pkgcoll and repo to recipe

Create new image

- Validate repos, recipe, pkgcoll
- Create image with local recipe
  - image create –r nersc_gpfs_client nersc_gpfs_client
  - image is written to /var/opt/cray/imps/image_roots
  - Image root is directory hierarchy
  - Can check install by chroot and rpm –qa
- At this point I had to cheat…
  - Copy gpfs update rpms to image_root/nersc_gpfs_client/tmp
  - chroot; rpm –Uvh /tmp/gpfs-4.1.1*rpm
- Now have an image root with GPFS installed
  - Tried a personality rpm test run
  - No kernel headers installed ☹

Try again…

- Find kernel-devel rpm and add to local recipe
- Create new image
  - Base first, then manually install GPFS update rpms
  - Try a test personality rpm run
  - Kernel header version.h not found ☹
  - No make installed ☹
  - No gcc installed ☹
- Find version.h and add a symlink
  - ln -s /usr/src/linux-3.12.48-52.27/include/uapi/linux/dvb/version.h /usr/src/linux-3.12.48-52.27/include/linux/version.h
- Find gcc rpms and add to local recipe

Try again…

- Create new image
  - You know the steps by now…
  - Success! At least so far
- Create a bootable image
  - image export nersc_gpfs_client
  - Bootable image is written to /var/opt/cray/imps/boot_images
- Assign image to a DVS node
  - cnode update –i /var/opt/cray/imps/boot_images/ nersc_gpfs_client.cpio –n c0-0cs3n2

Test Boot a DVS node

- Boot node
  - xtcli shutdown –n c0-0c0s3n2
  - xtbootsys –n c0-0c0s3n2
- Go to the dvs node
  - ssh dvs1
- Build the kernel modules for the personality rpm
  - /usr/lpp/mmfs/bin/mmbuildgpl
  - /usr/lpp/mmfs/bin/mmbuildgpl –build-package
  - Rpm is written to /root/rpmbuild
  - Copy the personality rpm back to the smw

How do we get the rest of the way?

- Rebuild the image with the personality RPM generated on the DVS server node.
  - This will have to be regenerated every time the kernel changes, even if GPFS doesn't change.
- Augment fstab with special mount options for NGF filesystems with simple_sync and ansible play
- Configure the DVS GPFS cluster
  - Standard operation – see the GPFS Administration docs
  - The GPFS cluster configuration is stored in /var/mmfs/gen/mmsdrfs. We need to disaster-proof the cluster configuration.
  - Non-volatile storage will do the job under normal circumstances, but need a method for disaster recovery.

New tools for GPFS config backup and restore

- New features in GPFS 4.1.1 (present) and 4.2 (documented)
  - mmsdrrestore – restores mmsdrfs from a file that you specify
  - mmsdrbackup callback – updates a backup copy of mmsdrfs every time the primary cluster manager sees a configuration change.
- If we can specify a writeable area on the exported bootraid, mmsdrbackup and mmsdrrestore will give us a path to transparent gpfs cluster recovery.
- Not implemented yet with CLE 6.0, but tested on NERSC's GPFS development cluster.

Summary

- GPFS installation under CLE 6.0 is still a work in progress.
- CLE 6.0/SMW 8.0 UP01 will have major improvements that will make GPFS installation somewhat easier.
- Not all issues have been resolved
  - Don't know all of what is changed for UP01
- Questions?

Acknowledgments

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. DEAC02-05CH11231.

References

- "Cray Management System for XC Systems with SMW 8.0/CLE 6.0 (Draft)", Harold Longley, May 2016.
- IBM Spectrum Scale Concepts, Planning, and Installation Information, IBM Corporation 2016