eLogin at Scale

Georg Rath
NERSC
Berkeley, USA
gbrath@lbl.gov

Douglas Jacobsen
NERSC
Berkeley, USA
dmjacobsen@lbl.gov

Abstract— **NERSC is currently in the process of converting a Cray CS system to an 800 node eLogin cluster. We operate two high-throughput Cray CS systems (PDSF and Genepool), beside four Cray XC production supercomputers (Cori, Edison, Gerty and Alva). With the adoption of CLE6 and our SMWFlow system management framework, the configuration management of the supercomputers was unified, which lead to reduced cost of system management, consistency across machines and lower turnaround times for changes in system configuration. To apply these advantages to the high-throughput systems we replace the current way of managing those systems, using xCAT and CFEngine, to one based on the Cray eLogin management platform. We will describe the current state of the project and how it will lead to a more consistent user experience across our systems and a marked decrease in operational efforts. It will be the cornerstone of a common job submission system for all systems at NERSC.**

I. Introduction

Through the introduction of SMFlow[1], management of multiple Cray supercomputers using a common repository for the state of the system has become possible. This functionality currently does not extend to the Cray CS line, as it uses a management stack incompatible with the one of the XC systems. Changes affecting all systems need to be kept in sync manually, which is an error prone and laborious process and leads to subtle differences in configuration across the systems. By treating our CS clusters as external login nodes[2][3] we demonstrate how it is possible to reuse large parts of existing configuration, while providing a common management interface to operational staff and a consistent experience to our users.

II. Creating an eLogin Cluster

We set up Gregor as test and development system for our Mendel cluster. Mendel is hosting the PDSF and Genepool systems. Gregor is composed of 16 CS300 nodes, connected to an InfiniBand fabric and Gigabit Ethernet for management.

A. *Cray eLogin Infrastructure*

eLogin nodes contain two disks and are provisioned statefully, the first disk is used to boot host the image, while the second one persists state across upgrades of the image. Up until to CLE6.0UP05 the external login nodes on a Cray XC system are managed using OpenStack[3], hosted on the Cray Management Controller (CMC), which provides image storage, provisioning and orchestration capability. Images and ConfigSets are managed by the Cray System Management Workstation (SMW), both for the outside and the inside of the system and then pushed to the CMC, which hosts the images in the Glance[5] service and deploys them using Ironic[6]. On initial boot of the image, the ConfigSet is transferred to disk using Fuel[7]. Starting with CLE6.0UP06 the functionality of the CMC is moved into the SMW and the OpenStack services are replaced by custom ones.

B. *Preparing the ConfigSet*

We chose to organize the nodes in our eLogin cluster by role, currently into “login” and “compute” roles. The corresponding nodegroups were created in the nodegroups worksheet. In the relevant worksheets we defined the necessary management and InfiniBand networks and disabled the Cray Programming environment, as it would be of limited use on a non-XC cluster.

C. *Image and Infrastructure*

We created a new image recipe, by cloning an existing eLogin recipe and swapping out system specific packages where necessary. As the image runs a largely unmodified version of SUSE Linux Enterprise Server (SLES), it is fit to boot on generic hardware. The resulting image is exported in qcow2 format.

To be able to quickly validate changes to the ConfigSet and image, we used a virtual machine with the exported image as a primary disk and the KVM/qemu direct kernel boot functionality.

Before being able to boot this image on physical hardware with only one disk and without setting up the supporting OpenStack infrastructure, which would require significant effort, modifications to the image are necessary: provided that disk setup and provisioning are done through methods other than OpenStack, the only dependency on its services is a meta_data.json, file which is transferred by HTTP, from a link-local address on boot of the image. This file contains auxiliary data like the hostname and SSH keys. For testing, this file can be dropped into the image, which locks the image to a single host, or the link-local address can be replaced by an address providing a minimal web-service, serving out host-specific meta_data.json files, based on the address of the requester. In both cases this is done through modification of /etc/opt/cray/pre-pivot.d/10OpenStackMetaData.sh. To skip disk setup of the second hard drive, /etc/opt/cray/pre-pivot.d/05DiskSetup.sh needs to be removed or commented out and /etc/ansible/elogin_persistent.yaml modified accordingly.

Despite liveupdates being disabled in the ConfigSet of the hosts, by modifiying /etc/opt/cray/liveupdates.conf to use a server hosting the necessary repositories, it will function anyway. In our case we set up an NGINX server proxying to the repositories hosted on the SMW.

To make the boot process more robust to repeated DHCP failures on heavily utilized management networks, the ifup script of the image initrd needed to modified to retry until success, as passing Dracut parameters via kernel command line did not prove to be effective.

D. *Preparing the Node*

The current setup of the test cluster requires the image to be present on the local disk, no automated mechanism for image rollout has been set up. In the current state of the project this is a minor issue, as nodes can be updated using a combination of the live update mechanism, for changes to the packages composing the image, and synchronization of ConfigSets from the SMW, for changes in configuration. We set up the partition structure as needed and chose the simplest way write the image to disk and used dd.

E. *Booting*

To boot the image we reuse the stateless boot model of the existing xCAT infrastructure. This enables us to debug issues with the image by providing an intermediary stage and acting as a de-facto boot loader. In the future we plan on using this stage to sync updated image versions and ConfigSets, using the InfiniBand network, on boot. After xCAT has booted into its stateless image, we use kexec to pivot onto the image stored on disk:

```
kexec --load /sda/vmlinuz
      --initrd=/sda/initrd
      --append="root=/dev/sda"
rmmod mlx5_ib mlx5_core
sync
umount -a
kexec -e
```

III. Discussion

A. *NERSC Experiences*

Setting up Gregor required a thorough and deep understanding of the Configuration Management Framework (CMF), the Image Management and Provisioning System (IMPS) and of the image boot process in general. Whereas high level management functions are documented, it was necessary to dive into low level portions of the stack. Implementation of the functionality in human-readable, interpreted code (as opposed to binary blobs), and building on open-source technologies like Ansible was very helpful, although a comparison of features and caveats between Ansible and Cray-Ansible would have been much appreciated, especially for people with prior experience with vanilla Ansible.

After the initial setup was complete and the configuration used on the other systems could be applied to Gregor, only minor changes in configuration parameters and Ansible plays were necessary to produce an exact copy of the environment we use on our XC machines. Changes and improvements made to our centralized Ansible plays now also apply our CS systems.

B. *Future Work*

1) Upgrade to UP06: With UP06 it will be possible to remove the custom xCAT intermediary layer and do disk setup and provisioning through the SMW.

2) Base image on OpenSUSE: For licensing reasons it could be beneficial to use OpenSUSE instead of SLES as a base operating system for the image. Preliminary tests have shown promising results, replacing SLES with OpenSUSE repositories provides a bootable image.

3) Federated Systems: Building on a unified compute environment, we plan to federate our Slurm clusters to a provide a seamless scheduling experience to our users.

IV. Conclusion

Through the adoption of a single management system across all the machines at NERSC, we reduce the effort necessary to manage those systems, while at the same time providing more consistent systems and ultimately enabling us to quickly provision and change systems to adapt to the ever changing requirements of the scientific community.

Acknowledgment

This manuscript has been authored by an author at Lawrence Berkeley National Laboratory under Contract No. DE-AC02-05CH11231 with the U.S. Department of Energy.

References

[1] Jacbsen, DM (2018): Managing the SMW as a git Branch, Available at https://github.com/NERSC/smwflow

[2] Cray Inc: XC™ Series SMW-managed eLogin Administration Guide (CLE 6.0.UP06) S-3021 Rev A

[3] Cray Inc: XC™ Series eLogin Administration Guide (CLE 6.0.UP05) S-2570

[4] OpenStack: https://www.openstack.org/

[5] OpenStack Glance: https://docs.openstack.org/glance/latest/

[6] OpenStack Ironic: https://wiki.openstack.org/wiki/Ironic

[7] OpenStack Fuel: https://wiki.openstack.org/wiki/Fuel