Building Non-standard Images for CSM Systems

Harold Longley
Hewlett Packard Enterprise
Minneapolis, Minnesota, United States
harold.longley@hpe.com

Isa Wazirzada
Hewlett Packard Enterprise
Comano, Switzerland
isa.wazirzada@hpe.com

Davide Tacchella
Hewlett Packard Enterprise
Basel, Switzerland
davide.tacchella@hpe.com

Dennis Walker
Hewlett Packard Enterprise
Las Vegas, Nevada, United States
dennis.walker@hpe.com

Andy Warner
Hewlett Packard Enterprise
Minneapolis, Minnesota, United States
andy.warner@hpe.com

ABSTRACT

HPC scientists increasingly must innovate using diverse toolchains crafted from various Linux distributions ensuring they meet individual and project-specific needs to tackle complex challenges. Advancements in User Services Software (USS) 1.3 and Cray System Management (CSM) 1.6 now enable booting non-standard images on managed nodes, moving beyond Cray Operating System (COS)/USS dependencies. This is facilitated by iSCSI-based OS image projection and the deprecation of DVS for this purpose.

This paper explores the creation of stock Linux images from RPMs--SUSE, OpenSUSE, RHEL, Rocky Linux--using CSM tools and recent CSM architectural enhancements. Whether developed in-house or externally sourced, images can be imported into rootfs artifact hosting. CSM tools further enable Ansible-driven image customization, booting the image, and applying post-boot Ansible configuration. Functionality is extended through enhancements with clients or agents for Slingshot Host Software, parallel file system, workload manager, node heartbeat, BOS, and CFS. Adhering to portable Ansible code best practices allows seamless integration for standard and non-standard OS images. SAT bootprep automates the creation of versioned boot and configuration artifacts.

CCS CONCEPTS

- Software Development Techniques
- Software Verification and Validation
- Configuration Management and Version Control Systems
- Development Frameworks and Environments
- Operating Systems
- Software Infrastructure
- Interoperability
- Software Performance
- Software Reliability

KEYWORDS

Cray System Management, CSM, image management, boot orchestration, iSCSI, Linux, distributions, OS, Cray EX, SuSE, RHEL, Ubuntu

1 Introduction

High-performance computing researchers are progressively driven to create novel solutions utilizing varied software stacks assembled from multiple Linux variants, adapting them to align with distinct user and project requirements for addressing intricate problems. Recent architectural enhancements to Cray System Management (CSM) have enabled booting artifacts on nodes other than the HPE-supported operating system which empowers the use of these other flavors of Linux to solve problems.

HPE Cray EX systems using CSM software can boot non-standard images for managed nodes instead of the Cray Operating System (COS)/User Services Software (USS) images based on SUSE Linux. CSM 1.6 and USS 1.3 have switched to iSCSI for the OS image projection to managed nodes which makes it easier to enable other operating systems that do not have Data Virtualization Service (DVS) clients built for them.

Further enabling this, the Boot Script Service (BSS) can deliver any set of boot artifacts (initrd, kernel, kernel parameters, and root squashfs) to a node when it is powered on and begins the PXEboot process. The initrd has Dracut scripts which allow the root filesystem delivery mechanism to be used to mount the root squashfs as a read-only network-based filesystem and then use an in-memory writable overlay.

This paper explores how to inject a new image into the artifact storage so that BSS can deliver it to a booting node and then progressively explores how to customize that image with agents to appear more like a COS/USS image so that the Boot Orchestration Service (BOS) and Configuration Framework Service (CFS), and

Hardware State Manager (HSM) services and various commands like cray CLI and System Admin Toolkit (SAT) tools are able to track the state of the node.

Creating an image on the CSM system that uses a stock Linux distribution based on rpms such as SUSE, openSUSE, RHEL, or Rocky Linux, can be done with IMS once the rpms have been added to their own set of versioned RPM repositories in Nexus. A kiwi-ng image description (IMS versioned recipe) can be created using the unique set of rpm repositories in Nexus, explicitly naming packages or package collections to be included in the image, and any special customization scripts or non-rpm content for the image.

To become more useful as a compute node or an application node, the image may need to include the Slingshot Host Software, a filesystem client for Lustre or Spectrum Scale, and a workload manager client (Slurm or PBS Pro), as well as the agents for node heartbeat, BOS, and CFS.

A unique CFS configuration can be created for this image. The Ansible code should use Ansible modules rather than calling shell scripts to keep the Ansible code portable between the COS/USS image and this special image, especially if it is not based on SUSE Linux. However, Ansible code for this specific Linux image could be created if the configuration actions do not apply to a COS/USS image. IMS could apply this CFS configuration to customize the image after IMS has built the image.

The image maturity model starts with being able to boot a generic image from RHEL, SUSE, or the others, then customizing the image with the minimum packages to interact with the CSM orchestration services, and, finally, a full production image with comprehensive customizations. If the image includes the node heartbeat agent, then the Hardware State Manager (HSM) will be able to detect that the node has booted Linux. If the image includes the BOS agent, then BOS could be used to boot the node and track the entire booting process similar to COS/USS nodes. The BOS session template determines whether CFS will be used or skipped to configure the node after it boots. If the CFS agent is included in the image, then CFS will automatically configure the node post-boot.

SAT bootprep can automate the creation of a new image with the non-standard OS when any of the rpm content changes or there is a change in the configuration to be applied to it. SAT bootprep can create new CFS configuration (versioned), new IMS image (versioned), and new BOS session template (versioned).

Lastly, once an image has been created for a compute node, that image can be turned into an Open Container Initiative (OCI) image for use as a container managed by Podman, Singularity, or Kubernetes on compute nodes or as a User Access Instance (UAI) container managed by Kubernetes (K3s) on User Access Nodes (UANs).

1.1 Enabling Technology: iSCSI Image Projection

In versions of HPE software prior to CSM 1.6 and USS 1.3 in the 25.03 software stack release, the image projection from management nodes to the diskless managed nodes, such as the compute nodes and User Access Nodes (UANs), of the operating system rootfs and Cray Programming Environment (CPE) images was done using the Data Virtualization Service (DVS). DVS server and client code depend on tight integration between the kernel and the DVS and Lustre Network (LNET) kernel modules. Switching to iSCSI for image projection with the Scalable Boot Content Projection Service (SBPS) in CSM 1.6 removes the dependency on the modified kernel in the COS base software and the DVS kernel module.

Among the other benefits of the new open-source friendly solution are active/active I/O operation with iSCSI and device mapper multipath for seamless failover and failback for clients, content projection over either the Node Management Network (NMN) or the High Speed Network (HSN) based on Slingshot, and enablement of future improvements in the areas of image access control and multi-tenancy support.

The iSCSI server is implemented on management nodes which are Kubernetes worker nodes using the standard Linux kernel with user space iSCSI target services, see Figure 1. The SBPS Marshal Agent manages the lifecycle of projected IMS image content where the content is backed by the Ceph S3 storage nodes and targetcli creates the fileio backing store and iSCSI LUNs. DNS “SRV” and “A” records are created for HSN and NMN for iSCSI target discovery from the iSCSI initiators running on client nodes. The result is that IMS images with specified labels are mounted as iSCSI LUNs.

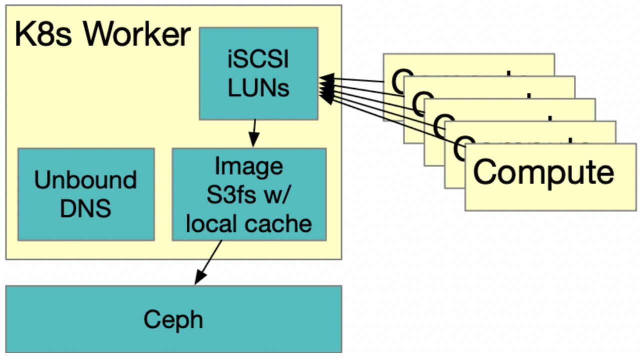

Figure 1. iSCSI image hosting with management worker nodes

The iSCSI client on a managed node uses the standard Linux kernel and user space iSCSI initiator services and multi-pathing software, see Figure 2. The iSCSI initiators are discovered using DNS “SRV” and “A” records provided to the node via kernel boot parameters. The Linux Device Mapper (DM) Multipath I/O is used for seamless failover and failback with active/active I/O load balancing. The kernel boot parameters and the rootfs image to be mounted on the node are declared in the BOS session template used to boot the nodes. There are Dracut scripts which must be in the initrd to interpret the kernel boot parameters related to how to mount the rootfs squashfs image over the HSN or NMN.

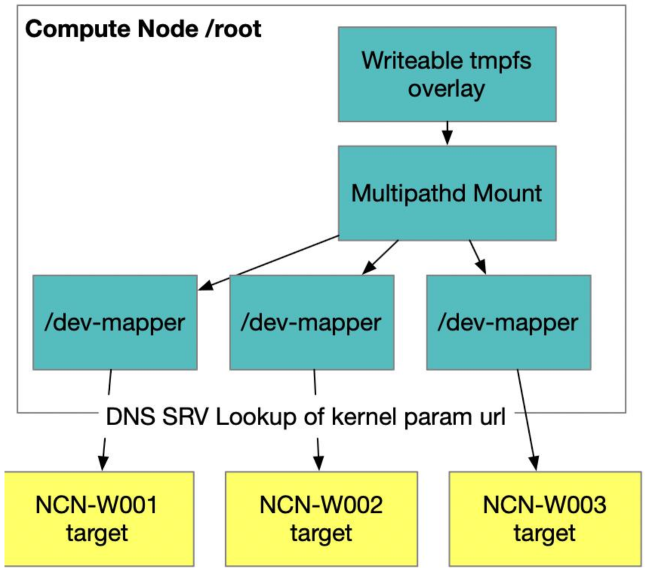

Figure 2. Compute node filesystem mount mechanisms

2 Image Maturity Model

The image maturity model is built in layers with the foundation of a minimal image from one of the Linux distributions, such as SUSE, OpenSUSE, RHEL, or Rocky Linux. The next set of content is added for the system management tooling needed for configuration of the image rootfs, modifying the initrd with Dracut script content in the initrd for interpretation of kernel boot parameters, node attestation, configuration of the High Speed Network (HSN) interfaces, preparation of iSCSI and device multipath targets from the iSCSI servers, node identity, node heartbeat, and credentials to permit post-boot configuration of the node. And finally, content and configuration are added to make the image ready for production use with site configuration such as the mounting of parallel file systems from Lustre, Spectrum Scale (GPFS), or external DVS-projected file systems, selecting versions of the programming environment, GPU tools, monitoring tools, and workload manager agents and prolog/epilog scripts.

There may be some iterative work to find the right set of software packages to be included in the image from the three sections. A minimal or a comprehensive set of packages from the Linux distribution can be chosen based on the intended function of the node which will receive the image. Commonly, a compute node may have fewer base Linux packages installed than a login node which needs to provide a richer environment with shells and programming and visualization utilities.

The core set of CSM packages presented in the next section enables the standard management tooling as if the image was built with the full COS and USS content. However, since DVS clients are not required for the image projection to the node, the DVS kernel modules with tight integration to the kernel are not needed in the initrd of the image. If DVS clients are needed for any externally projected file systems, that configuration can be done after the node has been booted.

CSM expects node attestation tooling in the image to verify the node's identity as it boots and ensure it is properly requesting access to appropriate services. Scripts are included so the network interfaces on either the Node Management Network (NMN) or the HSN are be discovered, have udev rules applied with possible renaming of the devices, and can be configured with appropriate network settings. The Linux iSCSI and device mapper multipath files are configured to be able to mount the iSCSI targets which have the rootfs over either the NMN or the HSN, and then pivot root from the Dracut environment into the read-write overlayfs on top of the read-only iSCSI-mounted rootfs. The image should also have packages and configuration applied to it so that log data is aimed to syslog aggregators and that the telemetry data from the node is aimed at monitoring collectors.

Once there is a functional system-manager-aware image that boots, the focus would move towards adding content to the image for the real function of the nodes which will use it. This functionality is needed by real user applications utilizing the shared file systems and may require a few iterations of image building, booting, and testing to ensure that appropriate performance tuning has been optimized for the workload. The collection of performance information from the node may be needed with the standard set of collectors for Lightweight Distributed Metrics Service (LDMS) or may be augmented with specific collectors for network performance counters, power utilization counters, GPU data, or site-specified metric data.

3 Image Curation, Build, and Boot Toolchain

On CSM systems images can be created from kiwi-ng recipes using the Image Management Service (IMS). These images can be customized via the Configuration Framework Service (CFS), where image level customizations are enumerated declaratively in Ansible. For the work we carried out, we created a custom kiwi-ng recipe and uploaded it to IMS. The unique recipe identifier of the recipe we uploaded to IMS can be seen in Figure 1, specifically in the `image` stanza at the `id` key. CFS can also be used for Ansible based post-boot node personalization. Last, the Boot Orchestration Service (BOS) can be utilized to build a session template which enumerates critical boot related metadata such as:

- An image identifier, this must match what's in the S3 object store
- Kernel parameters
- Optionally, a CFS configuration name can also be provided

With the advent of the System Admin Toolkit (SAT), one no longer needs to interact with the services manually and one can leverage the `bootprep` module to bring together the CFS configuration, image recipe, new image name, and BOS session template information. A specific example of a SAT bootprep file can be found in Figure 1. where one can see how one can declare specific operational parameters for each service. In Figure 1, there is a stanza for each service starting with CFS, then proceeding with IMS, and finally there is a stanza for BOS. The bootprep.yaml file is consumed by the `sat bootprep` command which orchestrates the curation and creation of boot artifacts. More information about CSM boot related services can be found at the link in reference 1.

```
---
schema_version: 1.0.2
configurations:
- name: "{{default.note}}compute-{{recipe.version}}{{default.suffix}}"
  layers:
  - name: csm-packages-{{csm.version}}
    playbook: csm_packages.yml
    product:
      name: csm
      version: "{{csm.version}}"
  - name: shs-{{default.network_type}}_install-{{slingshot_host_software.working_branch}}
    playbook: shs_{{default.network_type}}_install.yml
    product:
      name: slingshot-host-software
      version: "{{slingshot_host_software.version}}"
      branch: "{{slingshot_host_software.working_branch}}"
    special_parameters:
      ims_require_dkms: true

images:
- name: "{{default.note}}{{base.name}}{{default.suffix}}"
  ref_name: base_uss_image.x86_64
  base:
    ims:
      type: recipe
      id: e6669c1d-8d34-41bf-8e81-69c45939c2cc

- name: "compute-{{base.name}}"
  ref_name: compute_image.x86_64
  base:
    image_ref: base_uss_image.x86_64
  configuration: "{{default.note}}compute-{{recipe.version}}{{default.suffix}}"
  configuration_group_names:
  - Compute

session_templates:
- name: "{{default.note}}compute-{{recipe.version}}.x86_64{{default.suffix}}"
  image:
    image_ref: compute_image.x86_64
 configuration:"{{default.note}}compute-{{recipe.version}}{{default.suffix}}"
  bos_parameters:
    boot_sets:
      compute:
       arch: X86
      kernel_parameters: console=ttyS0,115200 crashkernel=512M@4G ip=dhcp quiet
spire_join_token=${SPIRE_JOIN_TOKEN}
        node_roles_groups:
        - Compute
        rootfs_provider: "sbps"
        rootfs_provider_passthrough:"sbps:v1:iqn.2023-06.csm.iscsi:_sbps-
hsn._tcp.{{default.system_name}}.{{default.site_domain}}:300"
```

Figure 3. SAT bootprep.yaml

4 CSM Required Packages

See Table 1 for the packages are required in the image to provide early boot, CSM, and iSCSI image projection content.

Package Name	**Source**	**Purpose**
iscsid	Linux	iSCSI
multipathd	Linux	Device mapper multipath
bos-reporter	CSM	Reports to state to BOS
cfs-debugger	CSM	CFS debugging agent
cfs-state-reporter	CSM	Reports state to CFS
cfs-trust	CSM	CFS authentication
cray-spire-dracut	CSM	Node attestation
craycli	CSM	cray command
csm-auth-utils	CSM	Node attestation
csm-node-heartbeat	CSM	Node heartbeat
hpe-yq	CSM	yq-utility
spire-agent	CSM	Node attestation
tpm-provisioner-client	CSM	Node attestation
cray-cps-dracut	USS	Content Projection
cray-cps-utils	USS	Content Projection
cray-netif-dracut	USS	Network Interface Config
cray-scripts-dracut	USS	Early boot configuration and rootfs pivot
cray-system-setup-scripts	USS	Node level tunings

Table 1. Required packages

See Table 2 for some packages which are optional or related to the workload execution.

Package Name
Workload Manager (SLURM or PBS)
LNet
bpcmdmod – Power Utilization Counters
Network Drivers and Kernel Modules
Lustre Client
xpmem
cray-crash-utility
msr-tools
cray-atom-energy-plugin
cray-freemem
cray-rasdaemon
cray-pals
msr-safe
cray-hugepage-setup
cray-libhugetlbfs

Table 2. Workload-related packages

5 Create a Stock SUSE Image

A stock SLE 15SP6 image can be created with the base Linux image as the foundation. The Linux distribution repositories need to be loaded into Nexus. The package managers will point to the URL for Nexus for rpm repositories. A kiwi-ng recipe with a customized list of packages and the repositories where they can be found should be used to create the base image. Otherwise, a base image can be downloaded from most Linux distro providers and imported into CSM.

For SuSE, Sample kiwi-ng recipes have been provided by CSM and USS in IMS for SUSE Linux versions. SUSE documentation describes many features in kiwi-ng enabling customization of the image built from this package list or inclusion of non-package content into the image.

Once the base image is available, a CFS configuration must be created to define the layers of configuration from CSM, SHS, and site-specific configuration Ansible code, although other layers from SMA, USS, and CPE may be needed for the production image. The base image will be modified using this new CFS configuration to create the customized image. The customized image and the CFS configuration will be placed into a BOS session template. The BOS session template can then be used to boot nodes.

6 Create a Non-SUSE Image

The previous section has described how an image can be built from a kiwi-ng recipe, but the CSM tools also enable importing an image built somewhere else into IMS and then customizing it.

This section describes how to create a base OS image outside of IMS, import the external image into IMS, create a new CFS configuration with at least CSM, SHS, and site-specific layers to add more packages and their configuration, use the new CFS configuration to customize the imported IMS image into a new, customized image, assign that customized image to nodes via a BOS session template, and then successfully boot the nodes.

7 Summary

In this paper, we have shown how to leverage the standard CSM 1.6 tools (CFS, IMS, BOS, and SAT bootprep) to curate a kiwi-ng recipe, build an image, customize an image, and boot an image with the stock SUSE SLE 15SP6 kernel instead of the COS base kernel or to import an image built externally with the same tools to customize it and boot it using a different kernel and Linux distribution.

8 Future Work

Due to time constraints before paper submission, some experiments were not completed. This future work will further validate the image model. Extensive testing of GPU tools and the Cray Programming Environment (CPE) software was not completed on stock SLE 15 SP6, but the content was confirmed to be present in the images built. Building images using kiwi-ng recipes for OpenSUSE, RHEL, and Rocky Linux was not attempted. Importing and then customizing images built elsewhere did work but was not extensively tested. Conversion of an image into an OCI container for use as either a UAI container on a UAN or on a compute node via podman, Singularity, or Kubernetes was not explored, although there is standard documentation in the USS Administration Guide [2] to describe this type of manipulation of compute node images for use as containers.

REFERENCES

[1] HPE Cray System Management 1.6 documentation. https://cray-hpe.github.io/docs-csm/en-16/

[2] HPE Cray Supercomputing User Services Software Administration Guide: CSM on HPE Cray Supercomputing EX Systems (1.3.0) (S-8063)