XC™ Series GPFS™ Software Installation Guide (CLE 6.0 UP02) S-2569

Contents

1 About the XC™ Series GPFS™ Software Installation Guide

CLE 6.0 UP02

This publication describes procedures for installing IBM® General Parallel File System (GPFS™) software on Cray XC Series systems running CLE 6.0 UP02.

This publication describes an approach to using IBM Spectrum Scale™ products (hereinafter referred to as GPFS) within the CLE 6.X release of Cray's Configuration Management Framework (CMF). The CMF approach to system configuration is different from prior releases, which were based on configured file system images and a shared root file system.

This publication is to be used in conjunction with IBM GPFS documentation. The user will need IBM reference material on hand or can contact IBM Service for additional help.

Table 1. Record of Revision

Publication Title	Date	Notes
XC™ Series GPFS™ Software Installation Guide (CLE 6.0 UP02)	November 2016	Initial release.

Related Publications

Publication Title	Notes
XC™ Series System Administration Guide	XC series system administration procedures.
XC™ Series Configurator User Guide	XC series Cray Management Framework (CMF) configuration tool user guide.

Scope and Audience

This publication is written for system administrators who must install software on Cray XC series computers.

Feedback

Visit the Cray Publications Portal at *http://pubs.cray.com* and make comments online using the **Contact Us** button in the upper-right corner or email pubs@cray.com. Your comments are important to us and we will respond within 24 hours.

Typographic Conventions

`Monospace`	Indicates program code, reserved words, library functions, command-line prompts, screen output, file/path names, and other software constructs.
`Monospaced Bold`	Indicates commands that must be entered on a command line or in response to an interactive prompt.
`Oblique` or *`Italics`*	Indicates user-supplied values in commands or syntax definitions.
Proportional Bold	Indicates a **GUI Window**, **GUI element**, cascading menu (**Ctrl**→**Alt**→**Delete**), or key strokes (press **Enter**).
\ (backslash)	At the end of a command line, indicates the Linux® shell line continuation character (lines joined by a backslash are parsed as a single line).

Trademarks

2 GPFS Installation

Prerequisites

These procedures assume that GPFS is provided by a service node configured with an operational Ethernet or Infiniband connection to an existing Spectrum Scale cluster that provides access to GPFS file systems.

Table 2. Terminology

Terms	Description
GPFS Base RPMs	GPFS base RPMs used to install the base IBM Spectrum Scale software.
GPFS Upgrade RPMs	GPFS RPMs used to upgrade the installed base IBM Spectrum Scale software version.
GPFS portability layer RPM	A GPFS RPM that installs the GPFS kernel modules on the system image. The kernel modules versions must match the kernel in the system image.
GPFS Base Image	A system image with the GPFS base RPMs installed.
GPFS Production Image	A system image with the GPFS base software, GPFS software upgrades and the GPFS portability layer installed. This image is ready for a service node to boot up on and operate as a member of a GPFS cluster.
System Image	A system image used to boot nodes in the Cray system. Examples: compute, MOM, network, and service nodes.
Image Recipe	A named set of instructions in CLE 6.X used to build a system image. The recipe defines the software RPMs that will be installed, the repositories where the RPMs are held and the process by which the system image is built.
Repository (repo)	A software distribution container in CLE6.x. Software RPMs are placed in a repo and referenced by recipes to build system images. CLE 6.0 has repos for SLES and other applications.

Installation Overview

1. Create repositories (repo) for the GPFS Software.

2. Create a recipe to build the base GPFS image. Store the IBM RPMs in a repository separate from SuSE or CLE repositories to simplify tracking and management.
3. Create the base GPFS image.
4. Optional: Update GPFS version if needed.
5. Build the GPFS portability layer.
6. Add the portability layer RPM to the base GPFS recipe. The file system image is now ready for configuration as part of the GPFS cluster.
7. Build the production GPFS image.
8. Configure, using the Cray Configurator.
9. Follow the GPFS server steps to enable the system to control the GPFS.

Figure 1. Workflow to build a Base GPFS image.

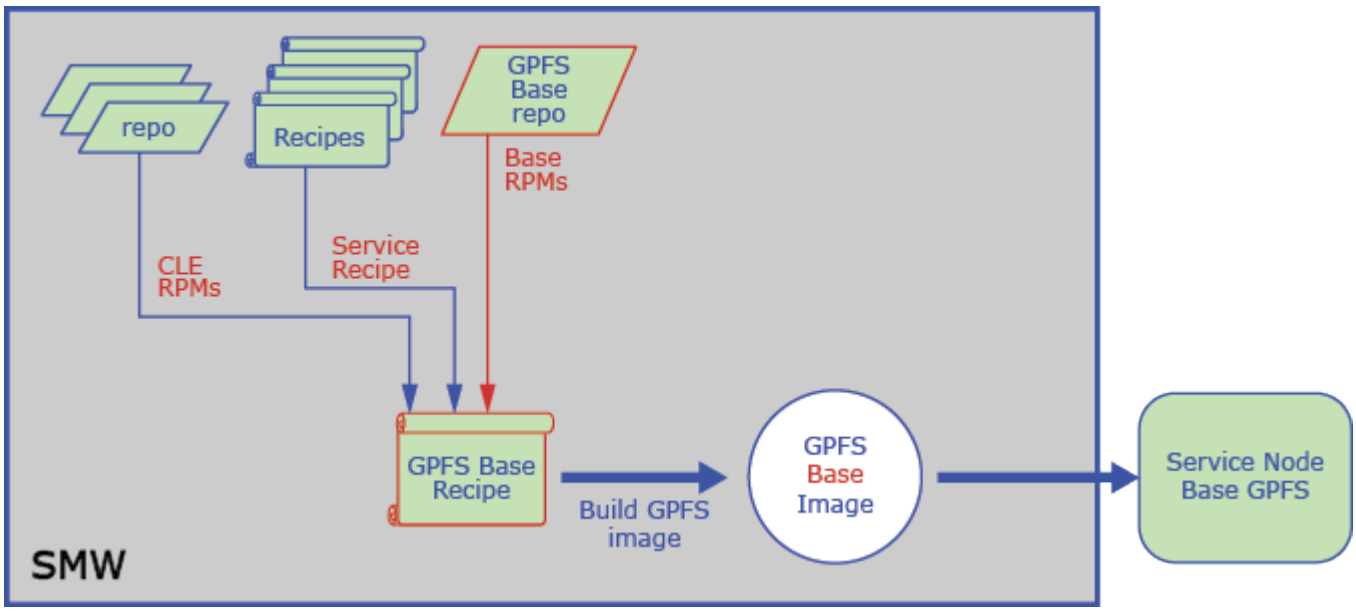
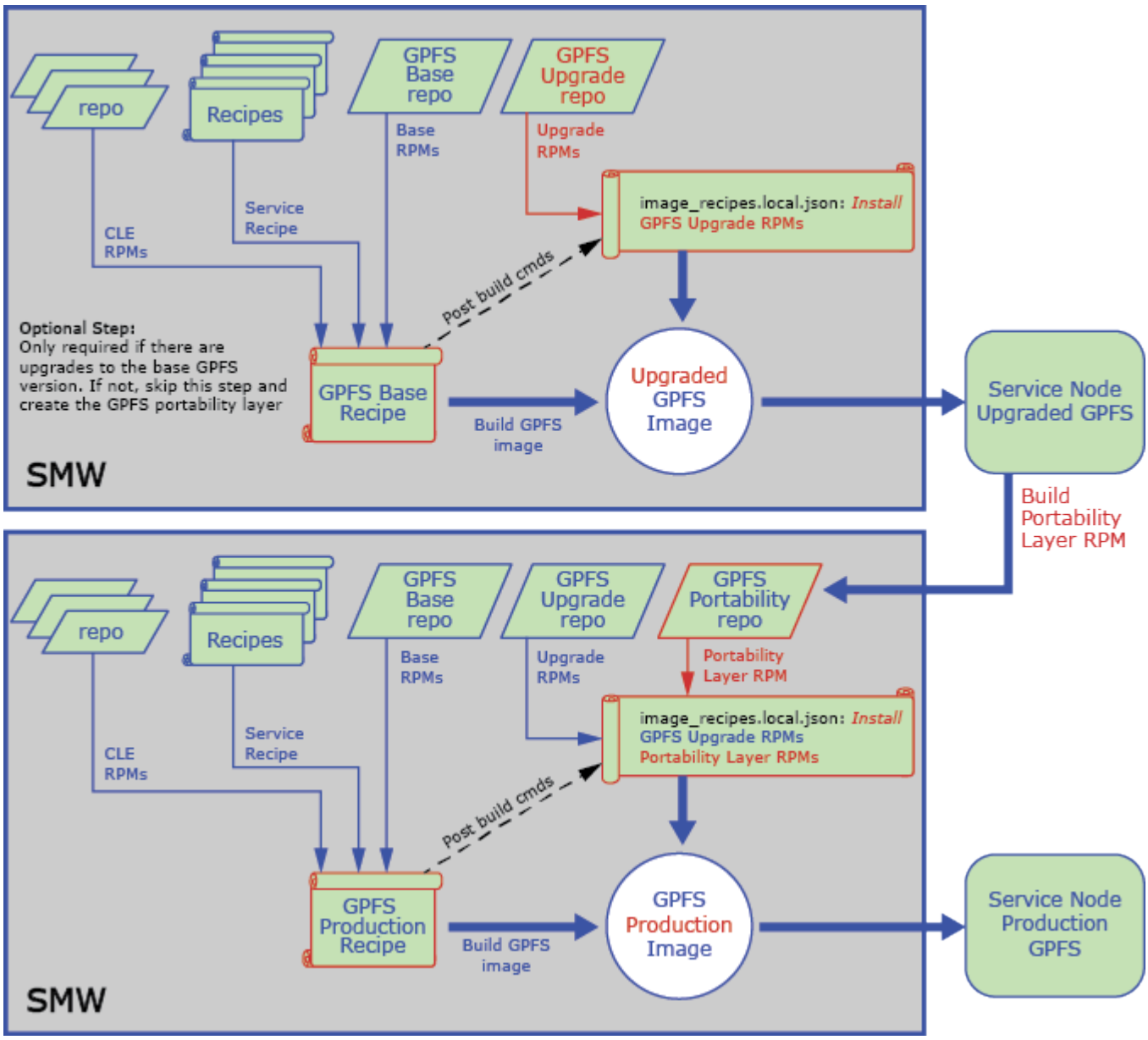


Figure 2. Workflow to build a Base GPFS image, with updates.



2.1 Create a GPFS Software Image

Prerequisites

The GPFS RPMs are loaded onto the SMW.

About this task

The Configuration Management Framework (CMF) provides a number of recipes to build standard images (for example: service, login, DAL or compute nodes). Use one of these image recipes as a base for the GPFS installation and change the boot parameters to get the appropriate node to use it. The first image compiles the portability layer and produces the version-dependent GPFS RPM. File system images are produced from IMPS recipes that specify which RPMs are required. A small set of RPMs leads to an extensive file system image that can be rolled up into a bootable archive and associated with a set of nodes. The service recipe is used as a base and is extended with new packages from a new repository holding the IBM-supplied base RPMs. Repositories are then searched to satisfy the requirements with the most up-to-date packages and their dependencies.

Procedure

1. Identify the base recipe on the service node to be modified to provide GPFS service. This example uses c0-0c0s0n2.

   ```
   smw# cnode list --fields image c0-0c0s0n2
   /var/opt/cray/imps/boot_images/service_cle_6.0.DV00-build201509140201_sles_12-
   created20150917.cpio
   ```

2. Search for the image root that produced the archive.

   NIMS provides the name of the `cpio` that booted the node:

   ```
   smw# grep Recipe service_cle_6.0.*/.imps_Image_metadata
   service_cle_6.0.DV00-build201508180201_sles_12-
   created20150820/.imps_Image_metadata:-
   '2015-08-20T07:19:57: Successful build of Recipe
   service_cle_rhine_sles_12_x86-64_ari
   service_cle_6.0.DV00-build201508250941_sles_12-
   created20150827/.imps_Image_metadata:-
   '2015-08-27T07:20:20: Successful build of Recipe
   service_cle_rhine_sles_12_x86-64_ari
   ```

   Although multiple hits appear, they are for the same recipe. This highlights the base recipe needed to modify the GPFS-bearing recipe.

3. Clone the recipe:

   ```
   smw# recipe create gpfs_service_cle_rhine_sles_12_x86-64_ari \
   --clone login_cle_rhine_sles_12_x86-64_ari
   ```

   This provides a place to make additions that allow control of software updates to either SuSE, Cray, or IBM components as they become available, without affecting other images.

4. Create the RPM repository.

   The IMPS image building tools gather content from numerous pools (or RPMs) into a single file system image root tree. Hold Spectrum Scale RPMs in a repository of their own, to manage them more easily:

   ```
   smw# repo create --arch x86-64 -t SLES12 repo-name
   ```

   The architecture for the image root is x86_64 and the O/S type is SLES 12. The repo name should allow easy human identification, but otherwise is not significant.

5. Add the required RPMs to the IBM-provided, base-licensed versions of GPFS RPMs in the repository:

It is assumed that these RPMS are unpacked in a directory in */path/to/media/gpfs_X.Y.Z*/Linux/x86_64.

```
smw# mkdir /tmp/repo
smw# cp /path/to/media/gpfs_X.Y.Z/Linux/x86_64/gpfs.{base,docs,gpl,msg}*rpm /tmp/repo
smw# repo update -a "/tmp/repo/*rpm" repo-name
```

6. Extend the recipe for the GPFS images:

   IBM provides four new packages that extend the recipe for the GPFS images: `gpfs.base`, `gpfs.docs`, `gpfs.gpl`, and `gpfs.msg.en_US`. The recipe clone above is updated with:

   ```
   smw# for p in gpfs.{base,docs,gpl,msg.en_US};\
   do recipe update -p $p gpfs_service_cle_rhine_sles_12_x86-64_ari;done
   ```

   Dependencies required by these packages are pulled in by the image creation process, without the need for specific nomination.

7. Build the GPFS software image for the first time:

   ```
   smw# image create -fr gpfs_service_cle_rhine_sles_12_x86-64_ari GPFS_image_name
   ```

   The last parameter, the name of the image, is conventionally related to the name of the recipe, and this document follows that convention.

8. Export the image root to create a bootable archive:

   ```
   smw# cd /var/opt/cray/imps/image_roots
   smw# image export gpfs_service_cle_rhine_sles_12_x86-64_ari
   smw# ls ../boot_images/gpfs*
   ../boot_images/gpfs_service_cle_rhine_sles_12_x86-64_ari.cpio
   ```

Everything is now in place for compiling the portability layer RPM on the booted image.

Continue on to boot the GPFS image.

2.2 Boot the GPFS Boot Image

Prerequisites

The GPFS base image has been created for CLE 6.

About this task

Use the base image to produce the version-specific portability layer RPM specified in the IBM documentation.

Procedure

1. Change the target node boot image to the newly extended image that includes the IBM RPMs:

   The example uses the target node `c0-0c0s0n2`.

   ```
   smw# cnode update --filter name=c0-0c0s0n2 -i \
   /var/opt/cray/imps/boot_images/gpfs_service_cle_rhine_sles_12_x86-64_ari.cpio
   ```

2. Reboot the node using standard `xtbootsys` invocation as `crayadm`:

```
crayadm@smw> xtbootsys --partition p0 -r 'Building RPM' --reboot c0-0c0s0n2
```

Continue on to create the recipe repository.

2.3 Configure the Final GPFS Software Image

Prerequisites

The recipe repository has been created and an extended image is booted and is running on a node.

About this task

The base RPMs provided by IBM need one last package, which will be built *in situ* and then copied back into the repository that holds the other IBM RPMs, to create the final image. Build the image with the release, date, and cpio for gpfs_service.

Procedure

1. Install the compilation dependencies.

   a. Copy the RPMs from the kernel source to the boot node. This example uses node `gpfs1`.

   ```
   smw# cd /var/opt/cray/repos/common_cle_6.0up01_sles_12_x86-64_ari
   smw# scp -p kernel-source-*.x86_64.rpm root@boot:/tmp
   smw# ssh boot
   boot# scp -p /tmp/kernel-source-*.x86_64.rpm gpfs1:/tmp
   boot# ssh gpfs1
   ```

   b. Install the new package.

   ```
   gpfs1# rpm -ivh /tmp/kernel-source-*.x86_64.rpm -oldpackage
   ```

   For the rest of the current boot of this node, these tools remain available to the IBM documented process for building the final RPM. If the node is rebooted for any reason, these packages must be re-installed and the build process restarted in order to capture the RPM.

2. Build the portability layer.

   Per IBM's documentation, the portability layer must be rebuilt for every patch level installed.

   ```
   gpfs1# export PATH=$PATH:usr/lpp/mmfs:usr/lpp/mmfs/bin
   gpfs1# make Autoconfig
   gpfs1# make World
   gpfs1# make InstallImages
   gpfs1# make rpm
   ```

   When using GPFS version 4.1 or later, use the mmbuildgpl tool to build the portability layer..

   ```
   gpfs1# export PATH=$PATH:usr/lpp/mmfs:usr/lpp/mmfs/bin
   eLogin# mmbuildgpl
   ```

```
…
mmbuildgpl: Building GPL module completed successfully at <DATE>
```

3. Update the recipe to pull in this package:

   Copy the RPM back to the SMW so it can added to the recipe.

```
gpfs1# scp -p /usr/src/packages/RPMS/x86_64/gpfs.gplbin-3.*.rpm root@boot:/tmp
gpfs1# exit
boot# scp -p /tmp/gpfs.gplbin-3.*.rpm root@smw:/tmp/gpfs/
boot# exit
smw# recipe update gpfs_service -p\
 /tmp/gpfs/gpfs.gplbin-3.12.51-52.31.1_1.0600.9120-cray_ari_s-4.2.0-2.x86_64.rpm
```

   After this recipe is rebuilt and booted, it is ready to be a GPFS client but does not have extra, unused content, such as kernel source/compilers.

   The version-specific RPM will include a kernel version component in its name, for example:

```
gpfs.gplbin-3.12.44-52.10.1_1.0000.8886-cray_ari_s-4.1.1-0.x86_64.rpm
```

   The string `3.12.44-52.10.1_1.0000.8886-cray_ari_s` is a reference to the kernel version as produced for a CLE release for XC™ Series systems. In this example `4.1.1` is the IBM Spectrum Scale release.

4. Rebuild the GPFS software image (`gpfs_service_cle_rhine_sles_12_x86-64_ari`) with the new recipe:

```
smw# image create -fr recipe_name GPFS_image_name
```

5. Export the image root to create a new bootable `cpio` archive.

```
 smw# image export GPFS_image_name
```

After this image is exported to the `cpio` archive, it can be used to boot the intended GPFS client node using the same `xtbootsys` process as before.

3 Configure a Persistent Storage Area for GPFS on Cray Nodes

About this task

GPFS requires a persistent area for the files written on the GPFS client. The storage area must be mounted at `/var/mmfs`. GPFS populates `/var/mmfs` with subdirectories and files when the node is added to a GPFS cluster. Use the configurator to create the `/var/mmfs` persistent storage area for a Cray node. For brevity, the following steps show only prompts and example responses.

Procedure

1. Invoke the configurator to modify the CLE config set.

   The configurator index numbering may vary. Check the actual listing to determine the correct number for the service/setting being configured.

```
smw# cfgset update -m interactive p0
Service Configuration List Menu (Config Set: p0, type: cle)
------------------------------------------------------------------------------
  Selected      #      Service                 Status (level=basic, state=unset)
------------------------------------------------------------------------------
             1)     cray_alps                 [ OK ]
             2)     cray_auth                 [ OK ]
             3)     cray_batchlimit           valid, disabled
             4)     cray_boot                 [ OK ]
             5)     cray_ccm                  [ OK ]
             6)     cray_cnat                 valid, disabled
             7)     cray_drc                  [ OK ]
             8)     cray_dvs                  [ OK ]
             9)     cray_dws                  [ OK ]
             10)     cray_elogin_lnet          [ OK ]
             ...
             25)     cray_net                  [ OK ]
             26)     cray_netroot_preload      [ OK ]
             27)     cray_node_groups          [ OK ]
             28)     cray_node_health          [ OK ]
             29)     cray_persistent_data      [ OK ]
------------------------------------------------------------------------------
```

2. Create a GPFS node group on the configurator.

   a. Select and view the `cray_node_groups` service.

```
Service List Menu [default: save & exit - Q] $ 27
Service Configuration List Menu (Config Set: p0, type: cle)
------------------------------------------------------------------------------
  Selected      #      Service              Status (level=basic, state=unset)
------------------------------------------------------------------------------
 ...
  *            27)     cray_node_groups          [ OK ]
```

```
 ...
--------------------------------------------------------------------------------
Service List Menu [default: configure - C] $ v
Service Configuration Menu (Config Set: p0, type: cle)

  cray_node_groups        [ status: enabled ]  [ validation: valid ]

--------------------------------------------------------------------------------
  Selected     #      Settings                      Value/Status (level=basic)
--------------------------------------------------------------------------------
               1)     groups
...
                         group_name: tier2_nodes          [ OK ]
                         group_name: elogin               [ OK ]
                         group_name: datawarp_nodes       [ OK ]
                         group_name: dvs_nodes            [ OK ]
...
--------------------------------------------------------------------------------
```

b. Select the `groups` setting and configure Cray Node Groups.

```
Cray Node Groups Configuration Service Menu [default: save & exit - Q] $ 1
Cray Node Groups Configuration Service Menu [default: configure - C] $ C
```

c. Add a node group.

```
cray_node_groups.settings.groups
[<cr>=set 20 entries, +=add an entry, ?=help, @=less] $ +
cray_node_groups.settings.groups.data.group_name
[<cr>=set '', <new value>, ?=help, @=less] $ gpfs_nodes
 cray_node_groups.settings.groups.data.gpfs_nodes.description
[<cr>=set '', <new value>, ?=help, @=less] $ GPFS server nodes
```

d. Add and set node group members.

```
cray_node_groups.settings.groups.data.gpfs_nodes.members
[<cr>=set 0 entries, +=add an entry, ?=help, @=less] $+
Add members (Ctrl-d to exit) $ c2-0c1s1n2
Add members (Ctrl-d to exit) $ c2-0c1s1n1
Add members (Ctrl-d to exit) $ c2-0c0s1n2
Add members (Ctrl-d to exit) $ c2-0c0s1n1
Add members (Ctrl-d to exit) $ <Ctrl-d>
|--- Information
|  *    4 entries modified. Press <cr> to set.
|---
cray_node_groups.settings.groups.data.gpfs_nodes.members
[<cr>=set 4 entries, +=add an entry, ?=help, @=less] $ <cr>
```

e. Set group and return to service list.

```
cray_node_groups.settings.groups
[<cr>=set 21 entries, +=add an entry, ?=help, @=less] $<cr>
Cray Node Groups Configuration Service Menu [default: save & exit - Q] $ ^^
```

3. Configure a persistent storage area for GPFS files.

a. Select the `cray_persistent_data` service and view the settings

```
Service List Menu [default: save & exit - Q] $ 11
Service Configuration List Menu (Config Set: p0, type: cle)
```

```
--------------------------------------------------------------------------------
  Selected      #      Service                Status (level=basic, state=unset)
--------------------------------------------------------------------------------
...
         *     11)     cray_persistent_data       [ OK ]
...
--------------------------------------------------------------------------------
Service List Menu [default: save & exit - Q] $ v
Service Configuration List Menu (Config Set: p0, type: cle)
--------------------------------------------------------------------------------
  Selected     #      Settings                              Value/Status (level=basic)
--------------------------------------------------------------------------------
               1)     mounts
                         mount_point: /var/opt/cray/aeld                [ OK ]
                         mount_point: /var/opt/cray/ncmd                [ OK ]
                         mount_point: /var/spool/PBS                    [ OK ]
                         mount_point: /var/spool/pbs_srm                [ OK ]
                         mount_point: /var/opt/cray/apptermd            [ OK ]
                         mount_point: /var/opt/cray/alps                [ OK ]
                         mount_point: /var/opt/cray/dws                 [ OK ]
                         mount_point: /var/spool/moab                   [ OK ]
                         mount_point: /var/spool/torque                 [ OK ]
                         mount_point: /var/opt/cray/rdma-credentials    [ OK ]
                         mount_point: /var/lib/mongodb                  [ OK ]

--------------------------------------------------------------------------------
```

4. Add a persistent directory entry for GPFS to the `mounts` settings.

```
Cray Persistent Data Configuration Service Menu [default: save & exit - Q] $ 1
Cray Persistent Data Configuration Service Menu [default: configure - C] $ C
cray_persistent_data.settings.mounts
[<cr>=set 11 entries, +=add an entry, ?=help, @=less] $+
cray_persistent_data.settings.mounts.data.mount_point
[<cr>=set '', <new value>, ?=help, @=less] $ /var/mmfs
cray_persistent_data.settings.mounts.data./var/mmfs.client_groups
[<cr>=set 0 entries, +=add an entry, ?=help, @=less] $ +
Add client_groups (Ctrl-d to exit) $ gpfs_nodes
Add client_groups (Ctrl-d to exit) $ <Ctrl-d>
|--- Information
|  *    1 entry modified. Press <cr> to set.
|---
cray_persistent_data.settings.mounts.data./var/mmfs.client_groups
[<cr>=set 1 entries, +=add an entry, ?=help, @=less] $<cr>
cray_persistent_data.settings.mounts
[<cr>=set 12 entries, +=add an entry, ?=help, @=less] $<cr>
cray_persistent_data        [ status: enabled ]  [ validation: valid ]

--------------------------------------------------------------------------------
  Selected     #      Settings                              Value/Status (level=basic)
--------------------------------------------------------------------------------
               1)     mounts
                         mount_point: /var/opt/cray/aeld                [ OK ]
                         mount_point: /var/opt/cray/ncmd                [ OK ]
                         mount_point: /var/spool/PBS                    [ OK ]
                         mount_point: /var/spool/pbs_srm                [ OK ]
                         mount_point: /var/opt/cray/apptermd            [ OK ]
                         mount_point: /var/opt/cray/alps                [ OK ]
                         mount_point: /var/opt/cray/dws                 [ OK ]
                         mount_point: /var/spool/moab                   [ OK ]
                         mount_point: /var/spool/torque                 [ OK ]
                         mount_point: /var/opt/cray/rdma-credentials    [ OK ]
                         mount_point: /var/lib/mongodb                  [ OK ]
                         mount_point: /var/mmfs                         [ OK ]

--------------------------------------------------------------------------------
```

5. Save and exit the configurator session.

```
Cray Persistent Data Configuration Service Menu [default: save & exit - Q] $ Q
```

6. Shut down and reboot the system.

There is now an area on the DVS server/ GPFS client nodes for the GPFS required files to be updated by the GPFS servers, including file system status.

4 GPFS Install with Updates

Prerequisites

1. One service node and one compute node are available for s/w builds and testing.
2. The IBM documentation for the GPFS version is available for reference.
3. The GPFS base software and any required updates are available on the SMW in the following locations:
   - Base RPMs: `/home/crayadm/gpfs/gpfs_source/lpp/gpfs_rpms`
   - Upgraded RPMs: `/home/crayadm/gpfs/gpfs_source/lpp-update/gpfs_rpms`

See IBM documentation for setup of the file system and definition of the clients.

About this task

Update the image root and boot archive for the nodes using Spectrum Scale whenever IBM releases updates from Fix Central. The supported process involves upgrading existing packages, not installing new ones.

Updated releases (patches) are applied by the `rpm` command `-U` option.

A mechanism within IMPS augments the installation process to accommodate non-RPM installation.

GPFS repositories are created to maintain the GPFS software on the SMW using the same method as the Cray system's software. This standardizes the installation, maintenance, and support procedures.

Procedure

1. Login to the SMW as `root` and start a script session to capture the command and output generated during the work session.

```
smw# ssh root@smw1
smw# script -af /home/crayadm/typescripts/gpfs-updates/gpfs-upgrade
```

2. Copy the GPFS source on the SMW.

```
smw# cd /home/crayadm/gpfs/gpfs_source/lpp
smw# scp -r user@source_system:/source_dir/lpp .
smw# cd /home/crayadm/gpfs/gpfs_source/lpp-update
smw# scp -r user@source_system:/source_dir/lpp-update .
```

3. Create the GPFS repositories.
   a. Create the repositories.

      Base RPMs:

```
smw# repo create gpfs-4.1.1 -t SLES12 --arch x86-64
```

Updated RPMs:

```
smw# repo create gpfs-4.1.1_updates -t SLES12 --arch x86-64
```

GPFS portability layer for 4.1.1-4:

```
smw# repo create gpfs-4.1.1-4_portability_layer -t SLES12 --arch x86-64
smw# repo create elogin gpfs-4.1.1-4_portability_layer -t SLES12 --arch x86-64
```

b. Verify the repositories were created.

```
smw# repo list | grep gpfs
smw# repo show gpfs-4.1.1
smw# repo show gpfs-4.1.1_updates
```

c. Add the GPFS RPMs to the new repositories.

```
smw# cd /home/crayadm/gpfs/gpfs_source/lpp/
smw# repo update -a ./4.1.1/gpfs_rpms/gpfs.base-4.1.1-0.x86_64.rpm gpfs-4.1.1
smw# repo update -a ./4.1.1/gpfs_rpms/gpfs.docs-4.1.1-0.noarch.rpm gpfs-4.1.1
smw# repo update -a ./4.1.1/gpfs_rpms/gpfs.gpl-4.1.1-0.noarch.rpm gpfs-4.1.1
smw# repo update -a ./4.1.1/gpfs_rpms/gpfs.msg.en_US-4.1.1-0.noarch.rpm \
gpfs-4.1.1
```

d. Verify the RPMs were added.

```
smw# repo show gpfs-4.1.1
smw# ls -l/var/opt/cray/repos/gpfs-4.1.1
smw# cd /home/crayadm/gpfs/gpfs_source/lpp-update
smw# repo update -a ./4.1.1/gpfs_rpms/gpfs.base-4.1.1-0.x86_64.rpm \
gpfs-4.1.1_updates
smw# repo update -a ./4.1.1/gpfs_rpms/gpfs.docs-4.1.1-0.noarch.rpm \
gpfs-4.1.1_updates
smw# repo update -a ./4.1.1/gpfs_rpms/gpfs.gpl-4.1.1-0.noarch.rpm \
gpfs-4.1.1_updates
smw# repo update -a ./4.1.1/gpfs_rpms/gpfs.msg.en_US-4.1.1-0.noarch.rpm \
gpfs-4.1.1_updates
```

e. Verify the updated RPMs were added.

```
smw# repo show gpfs-4.1.1_updates
smw# ls –l /var/opt/cray/repos/gpfs-4.1.1_updates
```

4. Create a GPFS recipe to build the image that will be used to build the GPFS portability RPMs.

a. Create a new empty GPFS recipe.

Service node:

```
smw# recipe create gpfs-service-lustre-2.7_cle_6.0up02_sles_12_x86-64
```

eLogin server:

```
smw# recipe create \
gpfs-custom-elogin-large-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari
```

b. Extend the new recipe with the latest service or eLogin recipe that includes Lustre.

Service node:

```
smw# recipe update \
--add-recipe gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari \
gpfs-service-lustre-2.7_cle_6.0.UP01_sles_12_x86-64_ari
```

eLogin server:

```
smw# recipe update --add-recipe \
custom-elogin-large-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari \
gpfs-custom-elogin-large-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari
```

c. Add the GPFS base repository to the recipes.

Service node:

```
smw# recipe update --add-repo gpfs-4.1.1 \
gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari
```

eLogin server:

```
smw# recipe update --add-repo gpfs-4.1.1 \
gpfs-custom-elogin-large-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari
```

d. Add the GPFS RPMs from the base GPFS repository to the recipes.

Service node:

```
smw# for p in gpfs.{base,docs,gpl,msg.en_US}; do recipe update –p \
$p  gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari  ; done
```

eLogin server:

```
smw# for p in gpfs.{base,docs,gpl,msg.en_US}; do recipe update –p \
$p gpfs-custom-elogin-large-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari ; done
```

e. Add any additional GPFS RPMs required depending on the cluster configuration.

Service node:

```
smw# recipe update -p \
/var/opt/cray/repos/gpfs-4.1.1/gpfs.gskit-8.0.50-40.x86_64.rpm \
gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari
smw# recipe update -p \
/var/opt/cray/repos/gpfs-4.1.1/gpfs.ext-4.1.1-0.x86_64.rpm \
gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari
```

eLogin server:

```
smw# recipe update -p \
/var/opt/cray/repos/gpfs-4.1.1/gpfs.gskit-8.0.50-40.x86_64.rpm \
gpfs-custom-elogin-large-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari
smw# recipe update -p \
/var/opt/cray/repos/gpfs-4.1.1/gpfs.ext-4.1.1-0.x86_64.rpm \
gpfs-custom-elogin-large-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari
```

f. Print the contents of the recipe to verify the RPMs were added.

Service node:

```
smw# recipe show gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari
```

eLogin server:

```
smw# recipe show gpfs-custom-elogin-large-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari
```

5. Add the GPFS update RPMs to the recipe (if needed).

   Service node:

   ```
   #smw recipe update --add-repo \
   gpfs-4.1.1_updates gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari
   ```

   eLogin server:

   ```
   #smw recipe update --add-repo \
   gpfs-4.1.1_updates \
   gpfs-custom-elogin-large-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari
   ```

6. Add post image build commands to the appropriate recipe's subsection of the `/etc/opt/cray/imps/image_recipes.d/image_recipes.local.json` file.

   a. Backup the `/etc/opt/cray/imps/image_recipes.d/image_recipes.local.json` file.

   b. Edit the `/etc/opt/cray/imps/image_recipes.d/image_recipes.local.json` file.

   c. Find the subsection for the new GPFS recipe.

      Service node: Search for `gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari` in the file.

      eLogin server: Search for: `gpfs-custom-elogin-large-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari` in the file.

   d. Add commands to upgrade the GPFS version after the image has been created. The post image build commands are inserted immediately preceding the “recipes” stanza in the new GPFS recipe. The post image build commands are inserted immediately preceding the “recipes” subsection in the new GPFS recipe.

      Recipe subsection:

      ```
      "recipes": [
                  "gpfs-service-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari"
              ],
      ```

      Add the post build chroot and copy sections.

      ```
       "postbuild_chroot": [
                  "rpm -Uvh $IMPS_POSTBUILD_FILES/*rpm"
              ],
              "postbuild_copy": [
                  "/var/opt/cray/repos/gpfs-4.1.1-4_updates/
      gpfs.base-4.1.1-4.x86_64.update.rpm",
                  "/var/opt/cray/repos/gpfs-4.1.1-4_updates/
      gpfs.gpl-4.1.1-4.noarch.rpm",
                  "/var/opt/cray/repos/gpfs-4.1.1-4_updates/
      gpfs.docs-4.1.1-4.noarch.rpm",
                  "/var/opt/cray/repos/gpfs-4.1.1-4_updates/
      gpfs.msg.en_US-4.1.1-4.noarch.rpm",
                  "/var/opt/cray/repos/gpfs-4.1.1-4_updates/
      gpfs.gskit-8.0.50-47.x86_64.rpm",
                  "/var/opt/cray/repos/gpfs-4.1.1-4_updates/
      ```

```
gpfs.ext-4.1.1-4.x86_64.update.rpm",
        ],
```

e. Save and exit the file.

f. Verify that the file syntax is correct.

Service node:

```
smw# recipe show gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari
```

eLogin server:

```
smw# recipe show gpfs-custom-elogin-large-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari
```

- If the recipe prints correctly, the syntax is correct.
- If the recipe is partially printed and exits with an error, the file edits have violated the file syntax. Review and edits until the syntax is correct.

7. Build a GPFS image using the new GPFS recipe.

Service node:

```
smw# Image create -vfr gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari \
gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari-Created$(date +%Y%m%d)
```

eLogin server:

```
smw# Image create -vfr  gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari \
gpfs-custom-elogin-large-lustre-2.7_cle_6.0up01_sles_12_x86-64_ari-Created$(date+%Y%m%d)
```

8. Verify that the new GPFS image has been built correctly.

a. Look for the following lines printed to the screen after the build:

Service node:

```
INFO - Image gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari-
Created20160720' RPM database successfully rebuilt.
```

eLogin server:

```
INFO - Image gpfs-custom-elogin-large-
lustre-2.7_cle_6.0up02_sles_12_x86-64_ari-Created20160720' RPM database
successfully rebuilt.
```

b. Verify that the post image build commands have completed successfully. Post image build messages and errors will be printed after the following debug output line:

```
DEBUG - * Executing post-build chroot script: 'rpm -Uvh
$IMPS_POSTBUILD_FILES/*rpm'
```

c. Verify that the new image contains the correct GPFS RPMs.

Service node:

```
smw# rpm -r /var/opt/cray/imps/image_roots \
/gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari-Created20160720 \
 -qa | grep gpfs
gpfs.msg.en_US-4.1.1-4.noarch
gpfs.gskit-8.0.50-47.x86_64
```

```
gpfs.docs-4.1.1-4.noarch
gpfs.gpl-4.1.1-4.noarch
gpfs.base-4.1.1-4.x86_64
```

eLogin server:

```
smw# rpm -r /var/opt/cray/imps/image_roots \
/gpfs-custom-elogin-large-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari-Created20160720 \
 -qa | grep gpfs
gpfs.msg.en_US-4.1.1-4.noarch
gpfs.gskit-8.0.50-47.x86_64
gpfs.docs-4.1.1-4.noarch
gpfs.gpl-4.1.1-4.noarch
gpfs.base-4.1.1-4.x86_64
```

9. Boot a service node on the new GPFS image. For eLogin, skip to step 15.

a. Export the image.

b. Assign the image to a service node.

c. Boot the service node using the GPFS image.

10. Build the GPFS portability layer RPM for service nodes and place it in the GPFS portability RPM repository. In this example, the cnode argument `c0-0c0s3n1` is used.

a. Copy the kernel source code to the service node running GPFS.

```
smw# cd /var/opt/cray/repos
smw# scp –r –o ProxyCommand=”ssh root@boot nc c0-0c0s3n1 22” \
./common_cle_6.0up02_sles_12_x86-64_ari/kernel-source-*.x86_64.rpm \
c0-0c0s3n1:/tmp
```

b. Login to the service node running the new GPFS image and install the kernel source on the service node.

```
smw# ssh root@boot
boot# ssh c0-0c0s3n1
```

c. Build the portability layer RPM.

```
c0-0c0s3n1# cd /usr/lpp/mmfs/src
c0-0c0s3n1# export  PATH=$PATH:usr/lpp/mmfs:usr/lpp/mmfs/bin
c0-0c0s3n1# make Autoconfig;make World; make InstallImages
c0-0c0s3n1# make rpms
```

When using GPFS version 4.1 or later, use the mmbuildgpl tool to build the portability layer.

```
gpfs1# export PATH=$PATH:usr/lpp/mmfs:usr/lpp/mmfs/bin
eLogin# mmbuildgpl
…
mmbuildgpl: Building GPL module completed successfully at <DATE>
```

d. Capture the following line from the `make rpms` command.

```
/usr/src/packages/RPMS/x86_64/gpfs.gplbin-3.12.51-52.31.1_1.0600.9146-
cray_ari_s-4.1.1-4.x86_64.rpm
```

e. Copy the new portability layer RPM to the SMW and import it into the portability layer repository.

```
c0-0c0s3n1# scp /usr/src/packages/RPMS/x86_64/
gpfs.gplbin-3.12.51-52.31.1_1.0600.9146-cray_ari_s-4.1.1-4x86_64.rpm \
```

```
boot:/var/tmp
c0-0c0s3n1# exit
boot# exit
```

On SMW: (Copy from boot node and into repo.)

```
smw# scp boot:/var/tmp/gpfs.gplbin-3.12.51-52.31.1_1.0600.9146-
cray_ari_s-4.1.1-4.x86_64.rpm \
/var/tmp/gpfs-4.1.1-4_portability_layer -t
smw# repo update -a gpfs-4.1.1-4_portability_layer \
/var/tmp/gpfs.gplbin-3.12.51-52.31.1_1.0600.9146-cray_ari_s-4.1.1-4.x86_64.rpm
```

11. Add the portability layer post image build command to the service node's recipe subsection in `/etc/opt/cray/imps/image_recipes.d/image_recipes.local.json`.

a. Back up `/etc/opt/cray/imps/image_recipes.d/image_recipes.local.json`.

b. Edit `/etc/opt/cray/imps/image_recipes.d/image_recipes.local.json`.

c. Find the subsection for the new GPFS recipe. Search for `gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari` in the file.

d. Add commands to upgrade the GPFS version after the image has been created.

The post image build commands are inserted immediately preceding the “recipes” subsection in the new GPFS recipe.

```
"recipes": [
"gpfs-service-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari"
],
```

The post-image build commands to insert for a service node or an eLogin server are as follows. Do not remove any of the existing post build commands.

```
"postbuild_chroot": [
            "rpm -Uvh $IMPS_POSTBUILD_FILES/*rpm"
        ],
        "postbuild_copy": [
          ...
            "/var/opt/cray/repos/gpfs-4.1.1-4_updates/
gpfs.ext-4.1.1-4.x86_64.update.rpm",
            "/var/opt/cray/repos/ gpfs-4.1.1-4_portability_layer/
gpfs.gplbin-3.12.51-52.31.1_1.0600.9146-cray_ari_s-4.1.1-4.x86_64.rpm",
        ],
```

e. Save and exit the file.

f. Verify that the file syntax is still correct.

```
smw# recipe show gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari
```

- If the recipe prints correctly the syntax is correct.
- If the recipe is partially printed and exits with an error. The file edits have violated the file syntax. Review and correct the edits until the syntax is correct.

12. Build the production GPFS image. Rebuild a GPFS image using the new GPFS recipe that includes the portability layer.

```
smw# Image create -vfr gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari \
gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari-Created$(date +%Y%m%d)
```

13. Verify that the new GPFS image has been built correctly, and includes the portability layer.

a. Look for the following lines printed to the screen after the build:

```
INFO - Image gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari-
Created20160720' RPM database successfully rebuilt.
```

b. Verify that the post image build commands have completed successfully. Post image build messages and errors will be printed after the following debug output line:

```
DEBUG - * Executing post-build chroot script: 'rpm -Uvh
$IMPS_POSTBUILD_FILES/*rpm'
```

c. Verify that the new image contains the correct GPFS RPMs.

```
smw# rpm -r /var/opt/cray/imps/image_roots \
/gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari-Created20160720 \
 -qa | grep gpfs
gpfs.msg.en_US-4.1.1-4.noarch
gpfs.gskit-8.0.50-47.x86_64
gpfs.docs-4.1.1-4.noarch
gpfs.gpl-4.1.1-4.noarch
gpfs.base-4.1.1-4.x86_64
gpfs-4.1.1-4_portability_layer \
/gpfs.gplbin-3.12.51-52.31.1_1.0600.9146-cray_ari_s-4.1.1-4.x86_64.rpm
```

14. Boot a service node on the new GPFS image.

a. Export the image.

b. Assign the image to a service node.

c. Boot the service node using the GPFS image.

The service node has now been booted with a Production GPFS image. Exit the procedure.

15. Boot an eLogin server on the new GPFS image.

a. Export the image.

```
smw# image export --format qcow2 -d \
glance:cmc:gpfs-custom-elogin-large-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari-
Created20160720 \
gpfs-custom-elogin-large-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari-Created20160720
```

b. Verify that the image was registered in Glance by looking for the following message in the output:

```
INFO - Image
          'gpfs-custom-elogin-large-
lustre-2.7_cle_6.0up01_sles_12_x86-64_ari-Created20160720'
          registered to Glance Service
```

16. Move the eLogin image to the CMC.

```
smw# cfgset push -d cmc1 global
smw# cfgset push -d cmc1 p0
```

17. Copy the kernel source to the CMC.

```
smw# cd /var/opt/cray/repos/common_cle_6.0up01_sles_12_x86-64_ari
smw# scp -p kernel-source-*.x86_64.rpm root@cmc1:/var/tmp
```

18. Import the config set changes into the CMC.

```
smw# ssh cmc1
cmc1# source ~/admin.openrc
cmc# add_configset -c global -e \
/etc/opt/cray/elogin/exclude_lists/elogin_cfgset_excludelist
cmc# add_configset -c p0 -e \
/etc/opt/cray/elogin/exclude_lists/elogin_cfgset_excludelist
```

19. Boot eLogin with the GPFS image, as per normal procedure. In this example, the server *eLogin1* is used. For further information, refer to *XC Series eLogin Installation Guide S-2566* and *XC Series eLogin Administration Guide S-2570*.

20. Install the kernel source on the eLogin server.

```
cmc1# scp -p /var/tmp/kernel-source-*.x86_64.rpm \
eLogin1# /var/tmp
eLogin1# rpm -ivh /tmp/kernel-source-*.x86_64.rpm
```

21. Build the portability layer on the eLogin server.

```
eLogin1# cd /usr/lpp/mmfs/src
eLogin1# export  PATH=$PATH:usr/lpp/mmfs:usr/lpp/mmfs/bin
eLogin1# make Autoconfig;make World;make InstallImages
eLogin1# make rpms
```

When using GPFS version 4.1 or later, use the mmbuildgpl tool to build the portability layer.

```
eLogin# export PATH=$PATH:usr/lpp/mmfs:usr/lpp/mmfs/bin
eLogin# mmbuildgpl
…
mmbuildgpl: Building GPL module completed successfully at <DATE>
```

22. Capture the following line from the `make rpms` command:

```
Wrote: /usr/src/packages/RPMS/x86_64/ gpfs.gplbin-3.12.60-52.57-
default-4.1.1-4.x86_64
```

23. Copy the new portability layer RPM to the SMW and import into the portability layer repository.

a. On the CMC:

```
cmc1# scp eLogin1:/usr/src/packages/RPMS/x86_64 \
/gpfs.gplbin-3.12.60-52.57-default-4.1.1-4.x86_64  /var/tmp
```

b. On the SMW:

```
smw# scp cmc1:/var/tmp/gpfs.gplbin-3.12.60-52.57-default-4.1.1-4.x86_64 \
/var/tmp/gpfs.gplbin-3.12.60-52.57-default-4.1.1-4.x86_64
smw# repo update -a elogin_gpfs-4.1.1-4_portability_layer \
/var/tmp/gpfs.gplbin-3.12.60-52.57-default-4.1.1-4.x86_64
```

24. Build the eLogin production image on the SMW. Add the portability layer to the eLogin post image build command list in `/etc/opt/cray/imps/image_recipes.d/image_recipes.local.json`.

a. Back up `/etc/opt/cray/imps/image_recipes.d/image_recipes.local.json`.

b. Edit `/etc/opt/cray/imps/image_recipes.d/image_recipes.local.json`.

c. Find the subsection for the new GPFS recipe. Search for `gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari` in the file.

d. Add commands to upgrade the GPFS version after the image has been created.

The post image build commands are inserted immediately preceding the “recipes” subsection in the new GPFS recipe.

```
"recipes": [
"gpfs-service-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari"
],
```

The post-image build commands to insert for a service node or an eLogin server are as follows. Do not remove any of the existing post build commands.

```
"postbuild_chroot": [
            "rpm -Uvh $IMPS_POSTBUILD_FILES/*rpm"
        ],
        "postbuild_copy": [
          ...
            "/var/opt/cray/repos/gpfs-4.1.1-4_updates/
gpfs.ext-4.1.1-4.x86_64.update.rpm",
            "/var/opt/cray/repos/ gpfs-4.1.1-4_portability_layer/
gpfs.gplbin-3.12.51-52.31.1_1.0600.9146-cray_ari_s-4.1.1-4.x86_64.rpm",
        ],
```

e. Save and exit the file.

f. Verify that the file syntax is still correct.

```
smw# recipe show gpfs-service-lustre-2.7_cle_6.0.UP02_sles_12_x86-64_ari
```

- If the recipe prints correctly the syntax is correct.
- If the recipe is partially printed and exits with an error. The file edits have violated the file syntax. Review and correct the edits until the syntax is correct.

25. Boot the eLogin server on the production image.

a. Export the image.

```
smw# image export --format qcow2 -d \
glance:cmc:gpfs-custom-elogin-large-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari-
Created20160720 \
gpfs-custom-elogin-large-lustre-2.7_cle_6.0up02_sles_12_x86-64_ari-Created20160720
```

b. Verify that the image was registered in Glance by looking for the following message in the output:

```
INFO - Image
          'gpfs-custom-elogin-large-
lustre-2.7_cle_6.0up01_sles_12_x86-64_ari-Created20160720'
          registered to Glance Service
```

26. Move the eLogin image to the CMC.

```
smw# cfgset push -d cmc1 global
smw# cfgset push -d cmc1 p0
```

27. Copy the kernel source to the CMC.

```
smw# cd /var/opt/cray/repos/common_cle_6.0up01_sles_12_x86-64_ari
smw# scp -p kernel-source-*.x86_64.rpm root@cmc1:/var/tmp
```

28. Import the config set changes into the CMC.

```
smw# ssh cmc1
cmc1# source ~/admin.openrc
cmc# add_configset -c global -e \
/etc/opt/cray/elogin/exclude_lists/elogin_cfgset_excludelist
cmc# add_configset -c p0 -e \
/etc/opt/cray/elogin/exclude_lists/elogin_cfgset_excludelist
```

29. Boot eLogin with the GPFS image, as per normal procedure.

The procedure is now complete.

- The GPFS Production image for Cray service nodes and/or eLogin nodes are created.
- The GPFS update and the portability layer are installed in the image.
- The GPFS base, update, and portability layer RPMs are stored in repositories.
- A build image used for upgrading the portability layer and testing after the system kernel updates are installed.

5 Configure a Persistent Storage Area for GPFS on eLogin

About this task

GPFS requires a persistent area for the files written on the GPFS client. The storage area must be mounted at `/var/mmfs`. GPFS populates the `/var/mmfs` directory with subdirectories and files when the node is added to a GPFS cluster. Different procedures are used to create the persistent storage area for a Cray and an eLogin server. Use bind mounts to mount a directory on the local disk on an eLogin server.

Procedure

1. Create the persistent storage directory on the eLogin local disk.

```
elogin1# mkdir /var/opt/cray/persistent/mmfs
elogin1# chown root:root/var/opt/cray/persistent/mmfs
elogin1# chmod 755/var/opt/cray/persistent/mmfs
```

2. Bind mount the persistent storage area at boot time before GPFS is started.

```
elogin1# mount -o bind /var/opt/cray/persistent/mmfs/ /var/mmfs
```

3. Verify the bind mount.

```
elogin1# ls /var/opt/cray/persistent/mmfs
ces  etc  gen data  mmbackup  mmpmon
elogin1# df /var/opt/cray/persistent/mmfs
Filesystem     1K-blocks      Used   Available Use%   Mounted on
/dev/sdb1      961334660 120785704 840548956  13% /var/opt/cray/persistent
elogin1# mount | grep mmfs
/dev/sdb1 on /var/mmfs type xfs (rw,relatime,attr2,inode64,noquota)
```

There is now an area on the DVS server/ GPFS client nodes for the GPFS required files to be updated by the GPFS servers, including file system status.

6 GPFS Cluster Configuration

Prerequisites

- The config set has been extended.
- Spectrum Scale requires that cluster members be able to `ssh` between nodes without the need of password or other interactive prompts.
- The configuration data used to provide access to the required GPFS file systems has been captured. Configuration data must be captured and replaced into the nodes' file systems as they boot on each machine restart.
- The boot time action to start GPFS services has been executed.

There are two configurations that can be used to connect Cray nodes and eLogin servers to existing GPFS clusters and file systems.

Additional information, requirements, and implementation details can be found in the IBM Spectrum Series documentation.

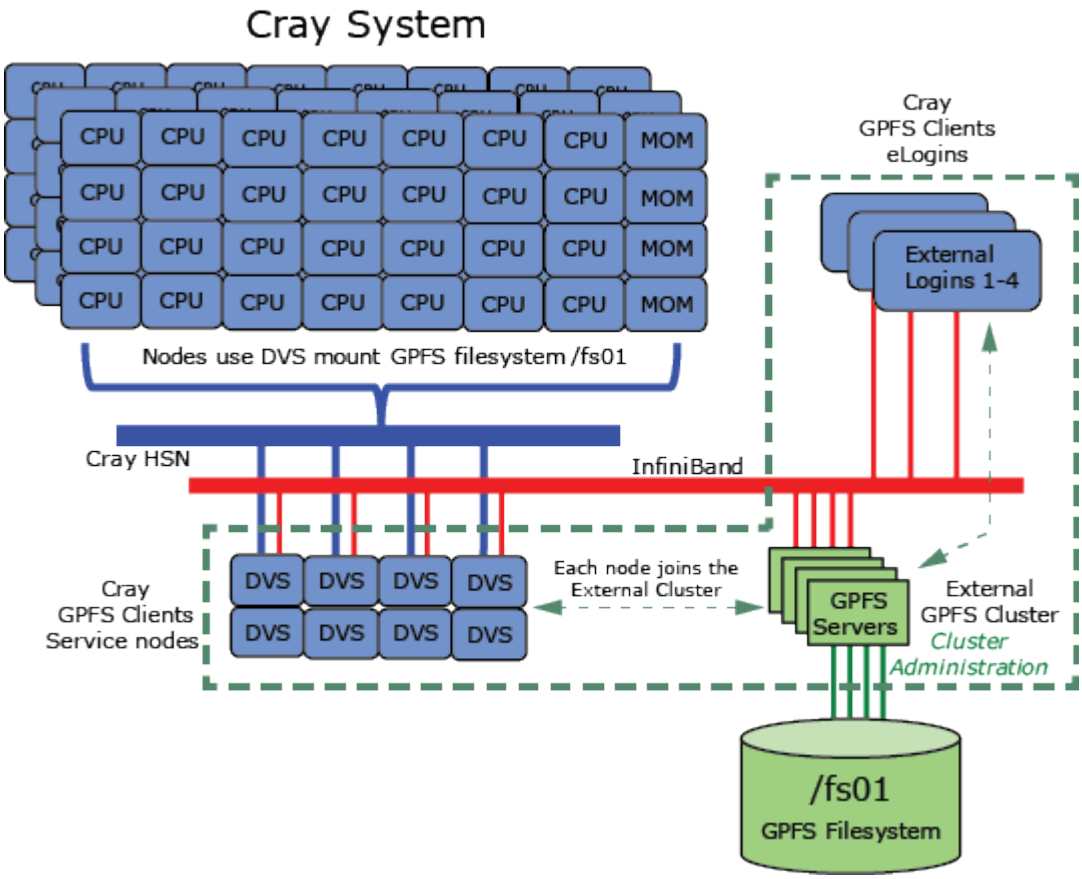
Table 3. GPFS Cluster Configuration Types

Configuration	Requirements
GPFS Client	GPFS client license is required for each node or server.
Remote Cluster Mount	GPFS server license is required for each node or server.

GPFS Client Configuration

Each node and/or server joins an existing cluster as a new GPFS client. The new Cray GPFS clients are managed by the existing GPFS administration environment along with any other nodes in the cluster.

Figure 1. Client configuration diagram displaying connections between nodes and clusters.

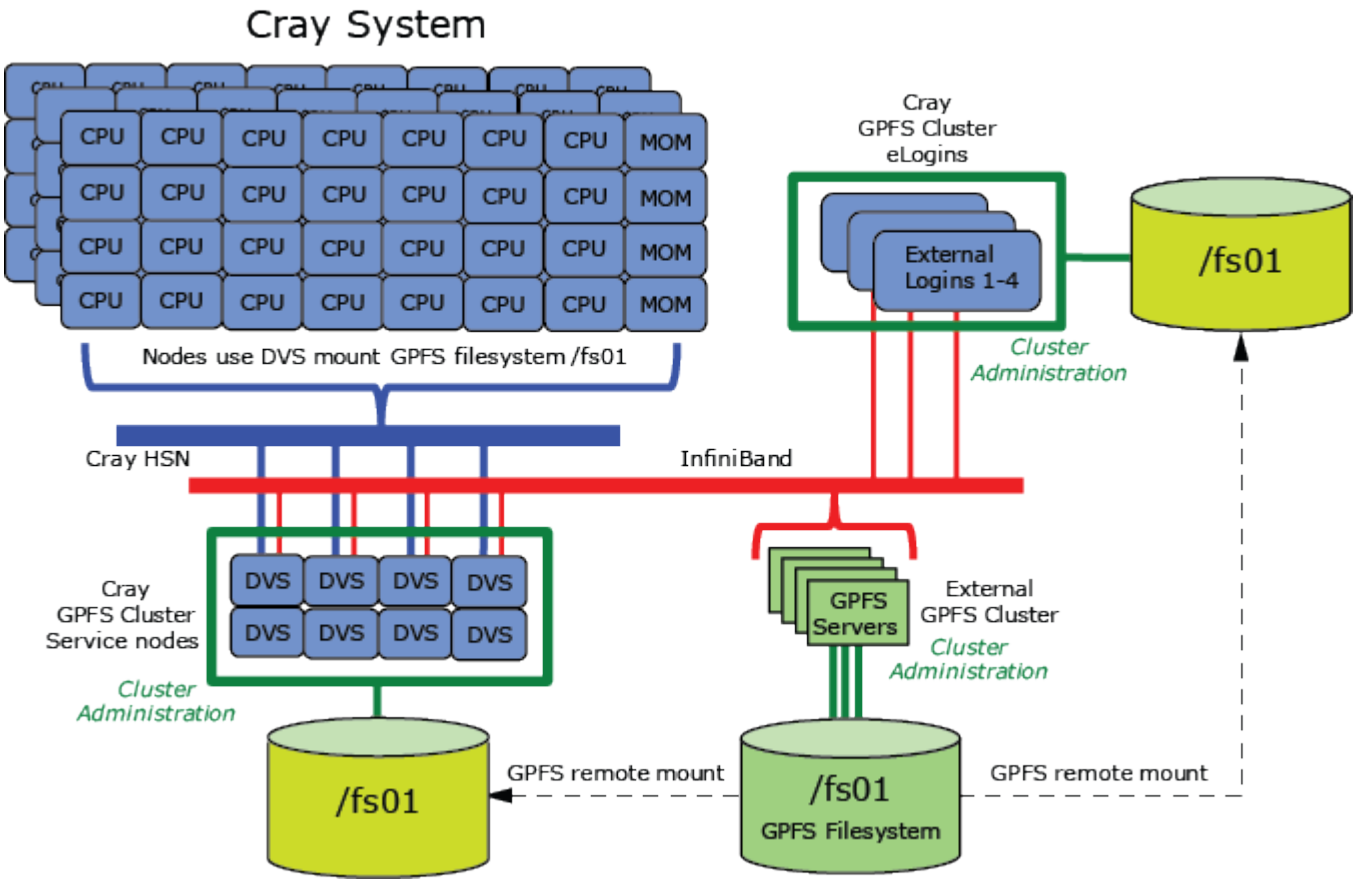


GPFS Remote Cluster Mount Configuration

The Cray DVS nodes and the eLogin nodes are configured as two or more standalone GPFS clusters that do not have Network Share Disks (NSDs). Each of the new Cray standalone GPFS clusters remotely mount GPFS file system(s) from an existing GPFS cluster.

An internal GPFS administration environment manages the standalone GPFS clusters. All administrative changes and faults are managed within the standalone clusters. GPFS propagates any change and/or fault notifications to the external cluster as needed.

Figure 2. Cluster mount configuration diagram displaying connections between nodes and clusters.



7 GPFS Client Recovery

About this task

These procedures recover nodes (`/var/mfs/` or `/var/mfs/gen/`*`system_files`*) if missing or corrupted.

If the persistent storage area `/var/mmfs/` is not available or the data has been deleted, the node will lose its GPFS configuration. However, the node will still be present in the GPFS cluster configuration on the GPFS management nodes.

Procedure

1. Determine if the cluster is in CCR mode.

   ```
   gpfsnode1# run /usr/lpp/mmfs/bin/mmlsconfig
   ```

   Look for `ccrEnabled yes`.

2. Take the next step depending on the cluster mode.

   a. If the cluster is not running CCR mode:

      ```
      gpfsnode1# mmsdrrestore -p primaryServer -R /usr/bin/scp -N recoveredNode
      ```

   b. If the cluster is running in CCR mode:

      ```
      gpfsnode1# mmsdrrestore -p primaryServer
      ```

The node has been recovered.