XC™ Series DataWarp™ Installation and Administration Guide (CLE 6.0.UP01) S-2564 Rev C

Contents

1 About the DataWarp Installation and Administration Guide

Scope and Audience

XC™ Series DataWarp™ Installation and Administration Guide (S-2564) covers DataWarp installation, configuration and administrative concepts and tasks for Cray XC™ series systems installed with DataWarp SSD cards. It is intended for experienced system administrators.

> **IMPORTANT:** Due to the deprecation of Static DataWarp and the introduction of the CLE 6.0 configuration management system (CMS), the DataWarp installation procedure is greatly simplified. Therefore, this document supersedes the *DataWarp Installation Guide (S-2547)*, which is no longer published.

Release Information

XC™ Series DataWarp™ Installation and Administration Guide supports the 6.0.UP01 release of the Cray Linux Environment (CLE). Significant changes have occurred since the release of CLE 5.2.UP04, including:

- Completely new installation procedures
- introduction of transparent caching and compute node swap
- Additional new procedures to:
  - over-provision Intel P3608 SSDs
  - flash Fusion IO SSD firmware
  - modify advanced configuration settings
  - configure SSD protection settings
  - backup and restore DataWarp state data
  - remove a node from a storage pool
  - change a node's pool
  - replace a DataWarp SSD
  - recover from a backwards-incompatible upgrade
  - recover from database corruption
- removal of command reference pages (i.e., `dwcli(8)`) now available as online man pages
- removal of all references to Static DataWarp

Revision Information

- (June 2016) Initial release.

- (July 2016) Rev A: added a note to the installation procedure in the event that CLE patch set 6.0.UP01.PS28 is installed prior to the initial installation/configuration of DataWarp as it introduces a new configuraton setting. For further information, see Field Notice #6103a.
- (September 2016) Rev B: use systemd commands rather than deprecated sysvint commands to start/stop services.
- (October 2016) Rev C:
  - back out Rev B change; sysvint commands should still be used as systemd commands not fully supported by DataWarp
  - correct the over-provisioning value for Intel P3608 devices

Typographic Conventions

`Monospace`	Indicates program code, reserved words, library functions, command-line prompts, screen output, file/path names, key strokes (e.g., `Enter` and `Alt-Ctrl-F`), 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 graphical user interface window or element.
\ (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). Do not type anything after the backslash or the continuation feature will not work correctly.
`smaller font size`	Some screenshot and code examples require more characters than are able to fit on a line of a PDF file, resulting in the code wrapping to a new line. To prevent wrapping, some examples are displayed with a smaller font to preserve the file formatting.

Command Prompt Conventions

hostname in command prompts

Hostnames in command prompts indicate where the command must be run.

`hostname``#`	Run the command on this hostname.
`smw#`	Run the command on the SMW.
`cmc#`	Run the command on the CMC.
`boot#`	Run the command on the boot node.
`sdb#`	Run the command on the SDB node.
`login#`	Run the command on any login node.

`smw1#` `smw2#`	For a system configured with the SMW failover feature there are two SMWs—one in an active role and the other in a passive role. The SMW that is active at the start of a procedure is *smw1*. The SMW that is passive is *smw2*.
`smwactive#` `smwpassive#`	In some scenarios, the active SMW is *smw1* at the start of a procedure—then the procedure requires a failover to the other SMW. In this case, the documentation will continue to refer to the formerly active SMW as *smw1*, even though *smw2* is now the active SMW. If further clarification is needed in a procedure, the active SMW will be called *smwactive* and the passive SMW will be called *smwpassive*.

account name in command prompts

The account that must run the command is also indicated in the prompt.

`smw#`
`cmc#`
`boot#`
`sdb#`
`login#`
`hostname#`
	The `root` or super-user account always has the `#` character at the end of the prompt.

`crayadm@smw>` `crayadm@login>`	Any non-`root` account is indicated with *`account@hostname`*.
`user@hostname>`	A user account that is neither `root` nor `crayadm` is referred to as *user*.

command prompt inside chroot

If the `chroot` command is used, the prompt changes to indicate that it is inside a chroot'd environment on the hostname.

```
smw# chroot /path/to/chroot
chroot-smw#
```

directory path in command prompt

Sometimes the current path can be so long that including it in the prompt does not add clarity to the command example. Most of the time, the command can be executed from any directory. When it matters which directory the command is invoked within, the `cd` command is used to change into the directory, and the directory is referenced with a period (`.`) to indicate the current directory.

For example, here are actual prompts as they appear on the system:

```
smw:~ # cd /etc
smw:/etc# cd /var/tmp
smw:/var/tmp# ls ./file
smw:/var/tmp# su - crayadm
crayadm@smw:~> cd /usr/bin
crayadm@smw:/usr/bin> ./command
```

And here are the same prompts as they would appear in examples in this publication:

```
smw# cd /etc
smw# cd /var/tmp
smw# ls ./file
smw# su - crayadm
crayadm@smw> cd /usr/bin
crayadm@smw> ./command
```

Related Documents

Although this publication is all that is necessary for installing SMW and CLE software, the following publications contain additional information that may be helpful.

1. *XC™ Series Software Installation and Configuration Guide*
2. *XC™ Series Configurator User Guide*
3. *XC™ Series DataWarp™ User Guide*

Feedback

Visit the Cray Publications Portal at *http://pubs.cray.com* and provide comments online using the *Contact Us* button in the upper-right corner or Email *pubs@cray.com*.

2 Initial DataWarp Service (DWS) Installation

Prerequisites

- A Cray XC series system running CLE 6.0.UP01 with one or more nodes with SSD hardware
- Identification of the nodes (cname) with SSD hardware
- Identification of the DataWarp API gateway nodes (cname), i.e., the nodes on which `dwrest` will run (must be login and/or MOM nodes)

Additional Requirement

A parallel file system (PFS) must be mounted in the same location on all compute nodes as well as all service nodes identified as `managed_nodes` within this procedure. In other words, the mount points must look the same on compute and SSD-endowed service nodes. More than one PFS is allowed.

This requirement can be implemented before or after the installation of DataWarp.

About this task

This procedure invokes the system configurator to initially configure the DataWarp service (DWS) and other services required by DataWarp. Note that this procedure does not cover how to use `cfgset` or the configurator, which is invoked by `cfgset`. For details, see the `cfgset(8)` man page, the *XC™ Series System Administration Guide*, and the *XC™ Series Configurator User Guide*.

The configurator guides the user through the configuration process with explanations, options, and prompts. The majority of this dialog is not displayed in the steps below, only prompts and example responses are displayed.

> **TIP:** The configurator uses index numbering to identify configuration items. This numbering may vary; the value used in the examples may not be correct for all systems. The user must search the listing displayed on the screen to determine the correct index number for the service/setting being configured.

Procedure

1. Invoke the configurator to configure the **cray_dws** service.

```
smw# cfgset update -m interactive -s cray_dws p0_example
Service Configuration Menu (Config Set: p0_example, type: cle)

  cray_dws        [ status: disabled ]  [ validation: skipped ]

------------------------------------------------------------------------------------
  Selected     #      Settings                              Value/Status (level=basic)
------------------------------------------------------------------------------------
                      service
                1)      scheduler_node                     [ unconfigured, default=sdb ]
                2)      managed_nodes                      [ unconfigured, default=(none) ]
                3)      api_gateway_nodes                  [ unconfigured, default=(none) ]
                4)      external_api_gateway_hostnames     [ unconfigured, default=(none) ]
                5)      dwrest_cacheroot_whitelist         [ unconfigured, default=/lus/scratch ]
                6)      allow_dws_cli_from_computes        [ unconfigured, default=False ]
```

```
--------------------------------------------------------------------------------
...
Cray dws Menu [default: save & exit - Q] $
```

IMPORTANT: If the list of settings includes `dwrest_cachemount_whitelist`, this indicates that CLE patch set 6.0.UP01.PS28 has been installed. Refer to Cray Field Notice (FN) #6103a for details. Otherwise, refer to FN #6103a after completing this procedure and consider installing the patch.

a. Enable the service.

```
Cray dws Menu [default: save & exit - Q] $ E
```

b. Set **scheduler_node**.

`scheduler_node` is the cname or role name of a service node that acts as the scheduler - the node that runs `dwsd`. Unless otherwise instructed by Cray support personnel, this is the `sdb` node, the default setting. This is a required setting.

```
Cray dws Menu [default: save & exit - Q] $ 1
Cray dws Menu [default: configure - C] $ C
cray_dws.settings.service.data.scheduler_node
[<cr>=set 'sdb', <new value>, ?=help, @=less] $ <cr>
```

c. Set **managed_nodes**.

`managed_nodes` is a list of nodes that have SSD hardware to be managed by DataWarp. They are also known as DataWarp server nodes. Enter nodes one per line, by cname. `managed_nodes` cannot be set by accepting the default; a non-empty list is required. If no nodes are entered, DataWarp fails.

```
Cray dws Menu [default: save & exit - Q] $ 2
Cray dws Menu [default: configure - C] $ C
cray_dws.settings.service.data.managed_node
[<cr>=set 0 entries, +=add an entry, ?=help, @=less] $ +
Add managed_nodes (Ctrl-d to exit) $ c0-0c0s7n2
Add managed_nodes (Ctrl-d to exit) $ c0-0c0s0n2
Add managed_nodes (Ctrl-d to exit) $ <Ctrl-d>
cray_dws.settings.service.data.managed_nodes
[<cr>=set 2 entries, +=add an entry, ?=help, @=less] $ <cr>
```

d. Set **api_gateway_hostnames**.

`api_gateway_nodes` is a list of login and scheduler MOM nodes on which `dwrest`, `nginx`, and `gunicorn` run. These nodes must be able to resolve UID and GID. Enter nodes one per line, by cname. `api_gateway_nodes` cannot be set by accepting the default; a non-empty list is required. If no nodes are entered, DataWarp fails.

IMPORTANT: Although multiple gateway nodes are allowed, Cray recommends configuring only one API gateway node unless otherwise directed by Cray support. Improperly configured API gateway nodes can significantly degrade performance.

```
Cray dws Menu [default: save & exit - Q] $ 3
Cray dws Menu [default: configure - C] $ C
cray_dws.settings.service.data.api_gateway_nodes
[<cr>=set 0 entries, +=add an entry, ?=help, @=less] $ +
Add api_gateway_nodes (Ctrl-d to exit) $ c0-0c0s1n1
Add api_gateway_nodes (Ctrl-d to exit) $ <Ctrl-d>
cray_dws.settings.service.data.api_gateway_nodes
[<cr>=set 1 entries, +=add an entry, ?=help, @=less] $ <cr>
```

e. Set **external_api_gateway_hostnames** to the default value `none`, because this feature is not yet fully functional. This is a required setting.

```
Cray dws Menu [default: save & exit - Q] $ 4
Cray dws Menu [default: configure - C] $ C
cray_dws.settings.service.data.external_api_gateway_hostnames
[<cr>=set 0 entries, +=add an entry, ?=help, @=less] $ <cr>
```

f. Set **dwrest_cacheroot_whitelist**.

`dwrest_cacheroot_whitelist` is a list of parallel file system (PFS) paths on which users are allowed to mount cache file systems. For example, if `dwrest_cacheroot_whitelist` is set as `/lus/scratch`, a user can mount the cache file system `/lus/scratch/seymour`. The PFS paths must be set up on all DataWarp service nodes for cache configurations to function. Each file system path specified and any subdirectories are considered valid.

> **IMPORTANT:** This is a required setting and the default value (`/lus/scratch`) must be changed unless `/lus/scratch` actually exists on the PFS and is the desired location. A non-existent path causes failures for cache configurations.

```
Cray dws Menu [default: save & exit - Q] $ 5
Cray dws Menu [default: configure - C] $ C
...
       Default:                    Current:
            1) /lus/scratch             1) /lus/scratch
...
cray_dws.settings.service.data.dwrest_cacheroot_whitelist
[<cr>=set 1 entries, +=add an entry, ?=help, @=less] $ 1*
Modify dwrest_cacheroot_whitelist:/lus/scratch (Ctrl-d to exit) $ /pfs/path/scratch
...
|--- Information
|  *    Entry 1 modified successfully.
|---

cray_dws.settings.service.data.dwrest_cacheroot_whitelist
[<cr>=set 1 entries, +=add an entry, ?=help, @=less] $ +
Add dwrest_cacheroot_whitelist (Ctrl-d to exit) $ /pfs/path/scratch2
Add dwrest_cacheroot_whitelist (Ctrl-d to exit) $ Ctrl-d
cray_dws.settings.service.data.dwrest_cacheroot_whitelist
[<cr>=set 2 entries, +=add an entry, ?=help, @=less] $ <cr>
```

g. Set **allow_dws_cli_from_computes**.

`allow_dws_cli_from_computes` determines whether commands such as `dwstat` and `dwcli` are executable on compute nodes. This is a required setting and is `false` by default, because scaling problems can occur if large numbers of compute nodes access the `dwrest` gateway simultaneously.

```
Cray dws Menu [default: save & exit - Q] $ 6
Cray dws Menu [default: configure - C] $ C
cray_dws.settings.service.data.allow_dws_cli_from_computes
[<cr>=set 'false', <new value>, ?=help, @=less] $ <cr>
```

```
Service Configuration Menu (Config Set: p0_example, type: cle)

  cray_dws        [ status: enabled ]  [ validation: valid ]

-------------------------------------------------------------------------------
  Selected     #      Settings                          Value/Status (level=basic)
-------------------------------------------------------------------------------
                      service
                 1)     scheduler_node                  sdb
                 2)     managed_nodes                   c0-0c0s7n2, c0-0c0s0n2
                 3)     api_gateway_nodes               c0-0c0s1n1
```

```
               4)       external_api_gateway_hostnames     (none)
               5)       dwrest_cacheroot_whitelist         /lus/scratch,
                                                           /lus/peel
               6)       allow_dws_cli_from_computes        False

-------------------------------------------------------------------------------
...
```

h. Save the settings and exit the configurator session.

```
Cray dws Menu [default: save & exit - Q] $ Q
```

2. Invoke the configurator to configure the **cray_persistent_data** service for DataWarp.

```
smw# cfgset update -m interactive -s cray_persistent_data p0_example
Service Configuration Menu (Config Set: p0_example, type: cle)

  cray_persistent_data        [ status: enabled ]  [ validation: valid ]

-------------------------------------------------------------------------------
  Selected     #      Settings                              Value/Status (level=basic)
-------------------------------------------------------------------------------
               1)     mounts
                         mount_point: /var/opt/cray/alps         [ OK ]
                         mount_point: /var/opt/cray/aeld         [ OK ]
                         mount_point: /var/opt/cray/apptermd     [ OK ]
                         mount_point: /var/opt/cray/ncmd         [ OK ]
-------------------------------------------------------------------------------
...
Cray Persistent Data Menu [default: save & exit - Q] $
```

a. Enable the service **only if** `status: disabled` is displayed.

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

b. Add a section to **mounts** for DataWarp.

```
Cray Persistent Data Menu [default: save & exit - Q] $ 1
Cray Persistent Data Menu [default: save & exit - Q] $ C
cray_persistent_data.settings.mounts
[<cr>=set 4 entries, +=add an entry, ?=help, @=less] $ +
cray_persistent_data.settings.mounts.data.mount_point
[<cr>=set '', <new value>, ?=help, @=less] $ /var/opt/cray/dws
cray_persistent_data.settings.mounts.data./var/opt/cray/dws.clients
[<cr>=set 0 entries, +=add an entry, ?=help, @=less] $ +
Add clients (Ctrl-d to exit) $ service
Add clients (Ctrl-d to exit) $ <Ctrl-d>
cray_persistent_data.settings.mounts.data./var/opt/cray/dws.clients
[<cr>=set 1 entries, +=add an entry, ?=help, @=less] $ <cr>
cray_persistent_data.settings.mounts
[<cr>=set 5 entries, +=add an entry, ?=help, @=less] $ <cr>
```

```
Service Configuration Menu (Config Set: p0_example, type: cle)

  cray_persistent_data        [ status: enabled ]  [ validation: valid ]

-------------------------------------------------------------------------------
  Selected     #      Settings                              Value/Status (level=basic)
-------------------------------------------------------------------------------
               1)     mounts
                         mount_point: /var/opt/cray/alps         [ OK ]
                         mount_point: /var/opt/cray/aeld         [ OK ]
                         mount_point: /var/opt/cray/apptermd     [ OK ]
                         mount_point: /var/opt/cray/ncmd         [ OK ]
                         mount_point: /var/opt/cray/dws          [ OK ]
```

```
--------------------------------------------------------------------------------
...
```

c. Save and exit the configurator.

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

The configurator takes the supplied values and ensures that they become part of the config set being created or updated.

3. Determine if the `cray_munge` service is enabled.

The `cray_munge` service defines munge attributes for creating and validating credentials. No DWS-specific settings are necessary; enabling `cray_munge` is sufficient.

```
smw# cfgset search --service-status p0_example | grep cray_munge
```

a. If `cray_munge.enabled: True` is displayed, proceed to step *4* on page 12.

b. If `cray_munge.enabled: False` is displayed, invoke the configurator and enable `cray_munge`.

```
smw# cfgset update -m interactive -s cray_munge p0_example
Service Configuration Menu (Config Set: p0_example, type: cle)

  cray_munge         [ status: disabled ]  [ validation: valid ]
...

Cray Munge Menu [default: save & exit - Q] $ E

Service Configuration Menu (Config Set: p0_example, type: cle)

  cray_munge         [ status: enabled ]  [ validation: valid ]
...
```

c. Save the settings and exit the configurator session.

```
Munge Menu [default: save & exit - Q] $ Q
```

4. Determine if the `cray_ipforward` service is enabled.

The `cray_ipforward` service enables IP forwarding between the service nodes and the SMW. It is enabled by default but may become disabled. No DWS-specific settings are necessary.

```
smw# cfgset search --service-status p0_example | grep cray_ipforward
```

a. If `cray_ipforward.enabled: True` is displayed, proceed to step *5* on page 13.

b. If `cray_ipforward.enabled: False` is displayed, determine if the `cray_ipforward` settings are inherited.

Because the `cray_ipforward` service has both a global and CLE template, it can be configured to inherit settings from the global config set. Therefore, it is first necessary to determine in which template it must be enabled.

```
smw# cfgset validate p0_example | grep cray_ipforward
```

c. If `'cray_ipforward': not enabled` is displayed, invoke the configurator for the `cray_ipforward` service within the CLE template, and enable the service.

```
smw# cfgset update -m interactive -s cray_ipforward p0_example
Service Configuration Menu (Config Set: p0_example, type: cle)

  cray_ipforward        [ status: disabled ]  [ validation: valid ]
...

Cray IP Forwarding Menu [default: save & exit - Q] $ E

Service Configuration Menu (Config Set: p0_example, type: cle)

  cray_ipforward        [ status: enabled ]  [ validation: valid ]
...
```

1. Save the settings and exit the configurator session.

   ```
   Cray IP Forwarding Menu [default: save & exit - Q] $ Q
   ```

2. Proceed to step 5 on page 13.

d. If `'cray_ipforward': inherits global config set values` is displayed, invoke the configurator for the `cray_ipforward` service within the `global` template, and enable the service.

```
smw# cfgset update -m interactive -s cray_ipforward global
Service Configuration Menu (Config Set: global, type: global)

  cray_ipforward        [ status: disabled ]  [ validation: valid ]
...

Cray IP Forwarding Menu [default: save & exit - Q] $ E

Service Configuration Menu (Config Set: global, type: global)

  cray_ipforward        [ status: enabled ]  [ validation: valid ]
...
```

e. Save the settings and exit the configurator session.

```
Cray IP Forwarding Menu [default: save & exit - Q] $ Q
```

5. Validate the global and CLE config sets. Correct any discrepancies before proceeding.

```
smw# cfgset validate global
...
INFO - ConfigSet 'global' is valid.
smw# cfgset validate p0_example
...
INFO - ConfigSet 'p0_example' is valid.
```

6. (Systems with Fusion IO SSD cards) Complete the *Create a New Service Node Image for Fusion IO SSDs* on page 13 procedure, if not already done, **before** rebooting the system in order to avoid a second reboot.

7. Reboot the system following the typical procedure in order to activate all DataWarp requirements.

   DWS is now enabled as part of CLE. Post boot configuration procedures are necessary to complete the DWS setup.

2.1 Create a New Service Node Image for Fusion IO SSDs

Prerequisites

- A Cray XC series system running CLE 6.0.UP01 with one or more Fusion IO (Sandisk) SSD cards installed
  - Fusion ioScale2 SSD cards are not supported
- Identification of the nodes (cname) with SSD hardware

About this task

Sites with Fusion IO (Sandisk) SSD cards must integrate the driver software into the service node image.

Procedure

1. Create a new image recipe for FIO service nodes and add a subrecipe of a service node image to it.

   Note that the font size is decreased in some examples below, because some line lengths are too wide for a PDF page. Unfortunately, some lines are so incredibly long that they still need to be continued on another line.

   TIP: Use `recipe list` to display current recipe names.

```
smw# recipe create fio-service_cle_6.0up01_sles_12_x86-64_ari
smw# recipe update --add-recipe service_cle_6.0up01_sles_12_x86-64_ari fio-service_cle_6.0up01_sles_12_x86-64_ari
smw# recipe update --add-coll datawarp-xtra_cle_6.0up01_sles_12 fio-service_cle_6.0up01_sles_12_x86-64_ari
smw# recipe update --add-repo \
passthrough-common_cle_6.0up01_sles_12_x86-64 fio-service_cle_6.0up01_sles_12_x86-64_ari
smw# recipe update --add-repo common_cle_6.0up01_sles_12_x86-64_ari fio-service_cle_6.0up01_sles_12_x86-64_ari
```

2. Edit the `cray_image_groups.yaml` file to add the image recipe and destination to the end of the `default` group.

```
smw# vi /var/opt/cray/imps/config/sets/global/config/cray_image_groups.yaml
```

   Add:

```
      - recipe: "fio-service_cle_6.0up01_sles_12_x86-64_ari"
        dest: "fio-service{note}_cle_{cle_release}-build{cle_build}{patch}_sles_12-created{date}.cpio"
        nims_group: "fio-service"
  fio-service:
      - recipe: "fio-service_cle_6.0up01_sles_12_x86-64_ari"
        dest: "fio-service{note}_cle_{cle_release}-build{cle_build}{patch}_sles_12-created{date}.cpio"
        nims_group: "fio-service"
```

   For example:

```
cray_image_groups:
  default:
      - recipe: "compute_cle_6.0up01_sles_12_x86-64_ari"
        dest: "{compute{note}_cle_{cle_release}-build{cle_build}{patch}_sles_12-created{date}.cpio"
        nims_group: "compute"
      - recipe: "login_cle_6.0up01_sles_12_x86-64_ari"
        dest: "login{note}_cle_{cle_release}-build{cle_build}{patch}_sles_12-created{date}.cpio"
        nims_group: "login"
      - recipe: "service_cle_6.0up01_sles_12_x86-64_ari"
        dest: "service{note}_cle_{cle_release}-build{cle_build}{patch}_sles_12-created{date}.cpio"
        nims_group: "service"
      - recipe: "fio-service_cle_6.0up01_sles_12_x86-64_ari"
        dest: "fio-service{note}_cle_{cle_release}-build{cle_build}{patch}_sles_12-created{date}.cpio"
        nims_group: "fio-service"
  fio-service:
      - recipe: "fio-service_cle_6.0up01_sles_12_x86-64_ari"
        dest: "fio-service{note}_cle_{cle_release}-build{cle_build}{patch}_sles_12-created{date}.cpio"
        nims_group: "fio-service"
```

3. Create the new group, and add the nodes to the group.

```
smw# cnode update -G service -g fio-service cname1,cname2,...
```

4. Build the updated image and update the node mappings.

```
smw# imgbuilder -g fio-service --map
```

5. Verify that the image is correctly assigned to the DataWarp nodes.

```
smw# cnode list cname1,cname2,...
```

   If the image is correctly assigned, proceed to the next step; otherwise, execute the following command to instruct NIMS to use the new image.

```
smw# cnode update -p p0_example --filter group=fio-service -i image_file
```

   Where *image_file* is the full path to the image, including the `cpio` file extension.

6. Complete the procedure: *Initial DataWarp Service (DWS) Installation* on page 8, if not already done, **before** rebooting the system in order to avoid a second reboot.

7. Reboot the system following the typical procedure in order to activate all DataWarp requirements.

   DWS is now enabled as part of CLE. Post boot configuration procedures are necessary to define a functional DWS state.

3 About DataWarp

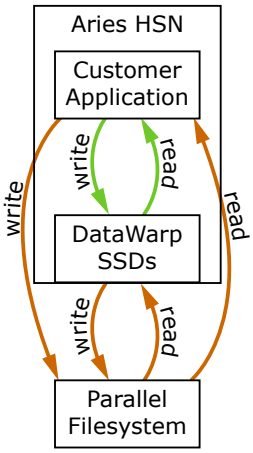
Cray DataWarp provides an intermediate layer of high bandwidth, file-based storage to applications running on compute nodes. It is comprised of commercial SSD hardware and software, Linux community software, and Cray system hardware and software. DataWarp storage is located on server nodes connected to the Cray system's high speed network (HSN). I/O operations to this storage completes faster than I/O to the attached parallel file system (PFS), allowing the application to resume computation more quickly and resulting in improved application performance. DataWarp storage is transparently available to applications via standard POSIX I/O operations and can be configured in multiple ways for different purposes. DataWarp capacity and bandwidth are dynamically allocated to jobs on request and can be scaled up by adding DataWarp server nodes to the system.

Each DataWarp server node can be configured either for use by the DataWarp infrastructure or for a site specific purpose such as a Hadoop Distributed File System (HDFS).

> **IMPORTANT:** Keep in mind that DataWarp is focused on performance and not long-term storage. SSDs can and do fail.

The following diagram is a high level view of how applications interact with DataWarp. SSDs on the Cray high-speed network enable compute node applications to quickly read and write data to the SSDs, and the DataWarp file system handles staging data to and from a parallel filesystem.

Figure 1. DataWarp Overview



DataWarp Use Cases

There are four basic use cases for DataWarp:

Parallel File System (PFS) cache	DataWarp can be used to cache data between an application and the PFS. This allows PFS I/O to be overlapped with an application's computation. In this release there are two ways to use DataWarp to influence data movement (staging) between DataWarp and the PFS. The first requires a job and/or application to explicitly make a request and have the DataWarp Service (DWS) carry out the operation. In the second way, data movement occurs implicitly (i.e., read-

ahead and write-behind) and no explicit requests are required. Examples of PFS cache use cases include:

- **Checkpoint/Restart**: Writing periodic checkpoint files is a common fault tolerance practice for long running applications. Checkpoint files written to DataWarp benefit from the high bandwidth. These checkpoints either reside in DataWarp for fast restart in the event of a compute node failure or are copied to the PFS to support restart in the event of a system failure.
- **Periodic output**: Output produced periodically by an application (e.g., time series data) is written to DataWarp faster than to the PFS. Then as the application resumes computation, the data is copied from DataWarp to the PFS asynchronously.
- **Application libraries**: Some applications reference a large number of libraries from every rank (e.g., Python applications). Those libraries are copied from the PFS to DataWarp once and then directly accessed by all ranks of the application.

Application scratch

DataWarp can provide storage that functions like a `/tmp` file system for each compute node in a job. This data typically does not touch the PFS, but it can also be configured as PFS cache. Applications that use out-of-core algorithms, such as geographic information systems, can use DataWarp scratch storage to improve performance.

Shared storage

DataWarp storage can be shared by multiple jobs over a configurable period of time. The jobs may or may not be related and may run concurrently or serially. The shared data may be available before a job begins, extend after a job completes, and encompass multiple jobs. Shared data use cases include:

- **Shared input**: A read-only file or database (e.g., a bioinformatics database) used as input by multiple analysis jobs is copied from PFS to DataWarp and shared.
- **Ensemble analysis**: This is often a special case of the above **shared input** for a set of similar runs with different parameters on the same inputs, but can also allow for some minor modification of the input data across the runs in a set. Many simulation stategies use ensembles.
- **In-transit analysis**: This is when the results of one job are passed as the input of a subsequent job (typically using job dependencies). The data can reside only on DataWarp storage and may never touch the PFS. This includes various types of workflows that go through a sequence of processing steps, transforming the input data along the way for each step. This can also be used for processing of intermediate results while an application is running; for example, visualization or analysis of partial results.

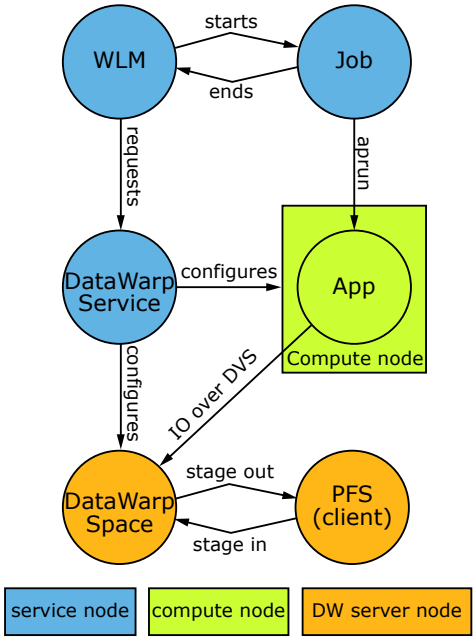
Compute node swap

When configured as swap space, DataWarp allows applications to over-commit compute node memory. This is often needed by pre- and post-processing jobs with large memory requirements that would otherwise be killed.

3.1 Overview of the DataWarp Process

The following figure provides visual representation of the DataWarp process.

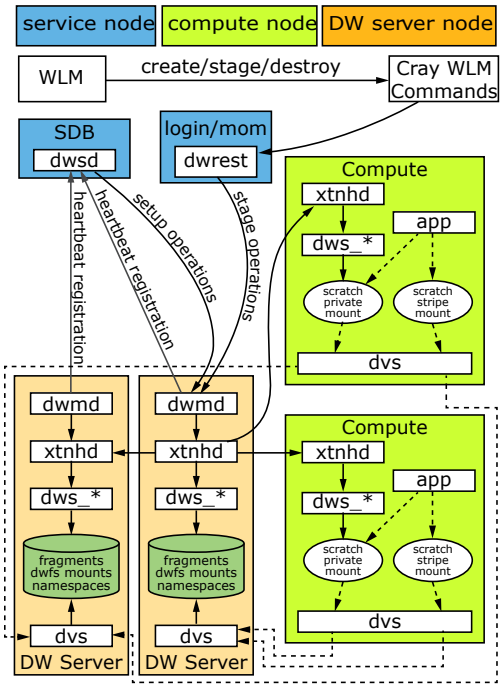
Figure 2. DataWarp Component Interaction - bird's eye view



1. A user submits a job to a workload manager. Within the job submission, the user must specify: the amount of DataWarp storage required, how the storage is to be configured, and whether files are to be staged from the parallel file system (PFS) to DataWarp or from DataWarp to the PFS.
2. The workload manager (WLM) provides queued access to DataWarp by first querying the DataWarp service for the total aggregate capacity. The requested capacity is used as a job scheduling constraint. When sufficient DataWarp capacity is available and other WLM requirements are satisfied, the workload manager requests the needed capacity and passes along other user-supplied configuration and staging requests.
3. The DataWarp service dynamically assigns the storage and initiates the stage in process.
4. After this completes, the workload manager acquires other resources needed for the batch job, such as compute nodes.
5. After the compute nodes are assigned, the workload manager and DataWarp service work together to make the configured DataWarp accessible to the job's compute nodes. This occurs prior to execution of the batch job script.
6. The batch job runs and any subsequent applications can interact with DataWarp as needed (e.g., stage additional files, read/write data).
7. When the batch job ends, the workload manager stages out files, if requested, and performs cleanup. First, the workload manager releases the compute resources and requests that the DataWarp service (DWS) make the previously accessible DataWarp configuration inaccessible to the compute nodes. Next, the workload manager requests that additional files, if any, are staged out. When this completes, the workload manager tells the DataWarp service that the DataWarp storage is no longer needed.

The following diagram includes extra details regarding the interaction between a WLM and the DWS as well as the location of the various DWS daemons.

Figure 3. DataWarp Component Interaction - detailed view



3.2 DataWarp Concepts

For basic definitions, refer to *Terminology* on page 69.

Instances

DataWarp storage is assigned dynamically when requested, and that storage is referred to as an *instance*. The space is allocated on one or more DataWarp server nodes and is dedicated to the instance for the lifetime of the instance. A DataWarp instance has a lifetime that is specified when the instance is created, either *job instance* or *persistent instance*. A job instance is relevant to all previously described use cases except the shared data use case.

- **Job instance**: The lifetime of a job instance, as it sounds, is the lifetime of the job that created it, and is accessible only by the job that created it.
- **Persistent instance**: The lifetime of a persistent instance is not tied to the lifetime of any single job and is terminated by command. Access can be requested by any job, but file access is authenticated and authorized based on the POSIX file permissions of the individual files. Jobs request access to an existing persistent instance using a persistent instance name. A persistent instance is relevant only to the shared data use case.

  IMPORTANT: New DataWarp software releases may require the re-creation of persistent instances.

When either type of instance is destroyed, DataWarp ensures that data needing to be written to the parallel file system (PFS) is written before releasing the space for reuse. In the case of a job instance, this can delay the completion of the job.

Application I/O

The DataWarp service (DWS) dynamically configures access to a DataWarp instance for all compute nodes assigned to a job using the instance. Application I/O is forwarded from compute nodes to the instance's DataWarp server nodes using the Cray Data Virtualization Service (DVS), which provides POSIX based file system access to the DataWarp storage.

A DataWarp instance is configured as scratch, cache, or swap. For scratch instances, all data staging between the instance and the PFS is explicitly requested using the DataWarp job script staging commands or the application C library API (`libdatawarp`). For cache instances, all data staging between the cache instance and the PFS occurs implicitly. For swap instances, each compute node has access to a unique swap instance that is distributed across all server nodes.

Scratch Configuration I/O

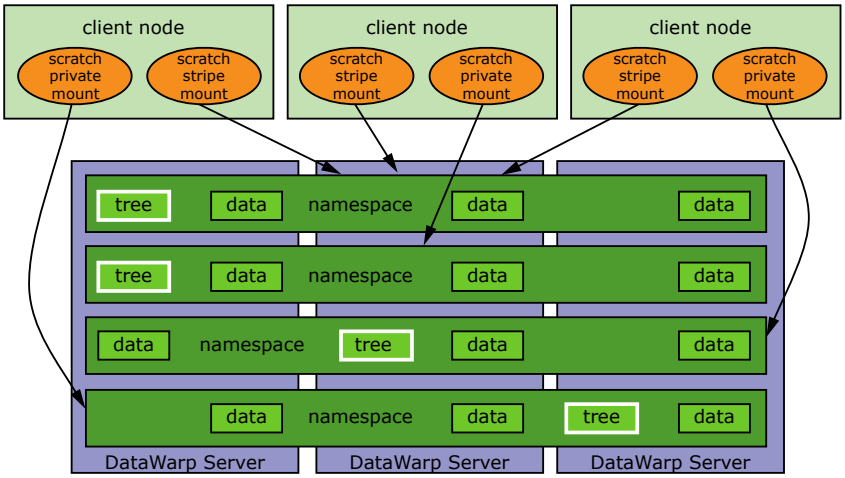
A scratch configuration is accessed in one or more of the following ways:

- **Striped**: In striped access mode individual files are striped across multiple DataWarp server nodes (aggregating both capacity and bandwidth *per file*) and are accessible by all compute nodes using the instance.
- **Private**: In private access mode individual files are also striped across multiple DataWarp server nodes (also aggregating both capacity and bandwidth *per file*), but the files are accessible only to the compute node that created them (e.g., /tmp). Private access is not supported for persistent instances, because a persistent instance is usable by multiple jobs with different numbers of compute nodes.
- **Load balanced**: (deferred implementation) In load balanced access mode individual files are replicated (read only) on multiple DataWarp server nodes (aggregating bandwidth but not capacity *per instance*) and compute nodes choose one of the replicas to use. Load balanced mode is useful when the files are not large enough to stripe across a sufficient number of nodes.

There is a separate file namespace for every scratch instance (job and persistent) and access mode (striped, private, loadbalanced) except persistent/private is not supported. The file path prefix for each is provided to the job via environment variables; see the *XC™ Series DataWarp™ User Guide*.

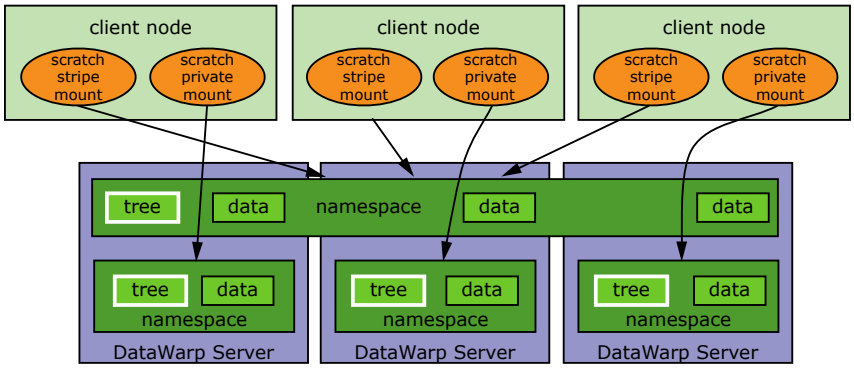
The following diagram shows a scratch private and scratch stripe mount point on each of three compute (client) nodes in a DataWarp installation configured with default settings for CLE 6.0.UP01; where `tree` represents which node manages metadata for the namespace, and `data` represents where file data may be stored. For scratch private, each compute node reads and writes to its own namespace that spans all allocated DataWarp server nodes, giving any one private namespace access to all space in an instance. For scratch stripe, each compute node reads and writes to a common namespace, and that namespace spans all three DataWarp nodes.

Figure 4. Scratch Configuration Access Modes (with Default Settings)



The following diagram shows a scratch private and scratch stripe mount point on each of three compute (client) nodes in a DataWarp installation where the scratch private access type is configured to not behave in a striped manner (`scratch_private_stripe=no` in the `dwsd.yaml` configuration file). That is, every client node that activates a scratch private configuration has its own unique namespace on only one server, which is restricted to one fragment's worth of space. This is the default for CLE 5.2.UP04 and CLE 6.0.UP00 DataWarp. For scratch stripe, each compute node reads and writes to a common namespace, and that namespace spans all three DataWarp nodes. As in the previous diagram, `tree` represents which node manages metadata for the namespace, and `data` represents where file data may be stored.

Figure 5. Scratch Configuration Access Modes (with `scratch_private_stripe=no`)



Cache Configuration I/O

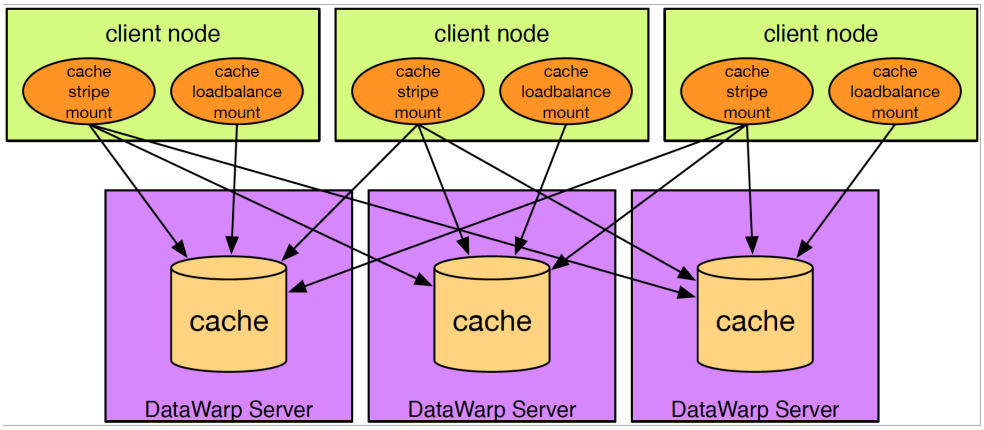
A cache configuration is accessed in one or more of the following ways:

- **Striped**: in striped access mode all read/write activity performed by all compute nodes is striped over all DataWarp server nodes.
- **Load balanced** (read only): in load balanced access mode, individual files are replicated on multiple DataWarp server nodes (aggregating bandwidth but not capacity *per instance*), and compute nodes choose one of the replicas to use. Load balanced mode is useful when the files are not large enough to stripe across a sufficient number of nodes or when data is only read, not written.

There is only one namespace within a cache configuration; that namespace is essentially the user-provided PFS path. Private access it is not supported for cached instances because all files are visible in the PFS.

The following diagram shows a cache stripe and cache loadbalance mount point on each of three compute (client) nodes.

Figure 6. Cache Configuration Access Modes

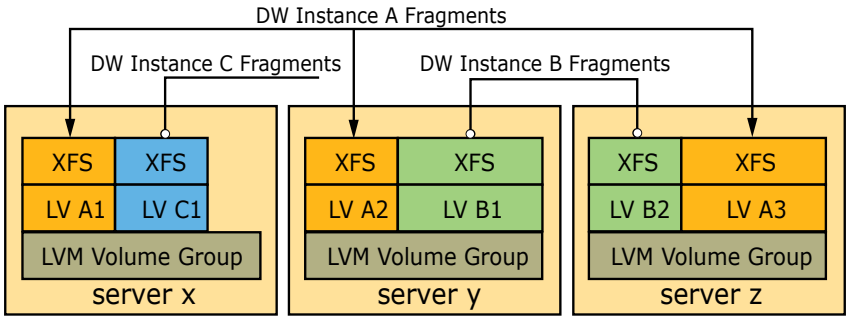


3.3 Instances and Fragments - a Detailed Look

The DataWarp Service (DWS) provides user access to subsets of storage space that exist between an arbitrary file system path (typically that of a parallel file system (PFS)) and a client (typically a compute node in a batch job). Storage space typically exists on multiple server nodes. On each server node, LVM combines block devices and presents them to the DWS as a Logical Volume Manager (LVM) volume group. All of the LVM volume groups on all of the server nodes compose the aggregate storage space. A specific subset of the storage space is called a DataWarp *instance*, and typically spans multiple server nodes. Each piece of a DataWarp instance (as it exists on each server node) is called a DataWarp instance fragment. A DataWarp instance fragment is implemented as an LVM logical volume.

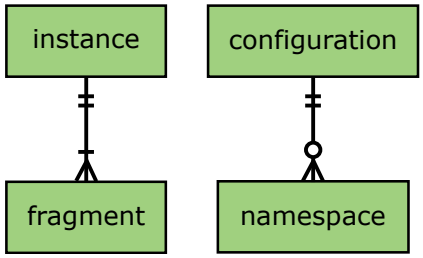
The following figure is an example of three DataWarp instances. DataWarp instance A consists of fragments that map to LVM logical volumes A1, A2, and A3 on servers x, y, z, respectively. DataWarp Instance B consists of fragments that map to LVM logical volumes y and z, respectively. DataWarp Instance C consists of a single fragment that maps to LVM logical volume C1 on server x.

Figure 7. Instances-Fragments LVM Mapping



The following diagram uses Crow's foot notation to illustrate the relationship between an instance-fragment and a configuration-namespace. One instance has one or more fragments; a fragment can belong to only one instance. A configuration has 0 or more namespaces; a namespace can belong to only one configuration.

Figure 8. Instance-Fragment and Configuration-Namespace Relationships



3.4 Storage Pools

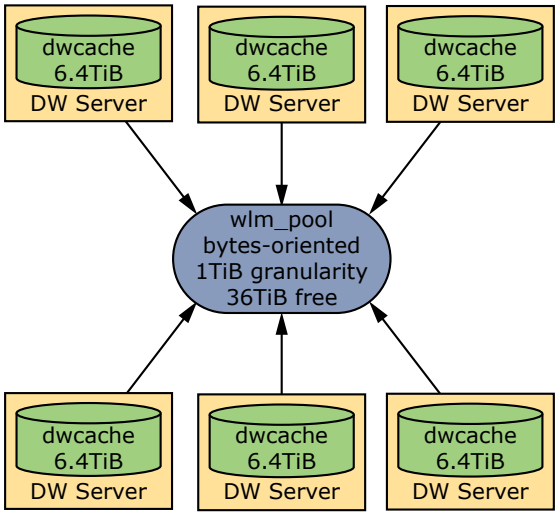
A storage pool groups nodes with storage together such that requests for space made against the pool are fulfilled from the nodes associated with the pool with a common allocation granularity. Pools have either byte or node allocation granularity (*pool_AG*). This release of DataWarp only supports byte allocation granularity. There are tradeoffs in picking allocation granularities too small or too large.

All pools must meet the following requirements:

- The byte-oriented allocation granularity for a pool must be at least 16MiB.
- Each node's volume group (`dwcache`, configured in *Initialize an SSD* on page 31) has a Physical Extent size (*PE_size*) and Physical Volume count (*PV_count*). The default *PE_size* is 4MiB, and *PV_count* is equal to the number of Physical Volumes specified during volume group creation. The DataWarp service (DWS) places the following restriction on nodes associated with a pool:
  - A node can only be associated with a storage pool if the node's granularity (*PE_size* * *PV_count*) is a factor of the pool's allocation granularity (*pool_AG*). The `dwstat nodes` command lists the node's granularity in the `gran` column.

The following diagram shows six different DataWarp nodes belonging to a storage pool `wlm_pool` with a 1TiB allocation granularity. Each DataWarp node has 6.4TiB of space, which means that 0.4TiB are wasted per node because only 6 allocation granularities fit on any one node.

Figure 9. Storage Pool Example



For an in depth look at pools, see *Create a Storage Pool* on page 35.

3.5 Registrations

A configuration represents a way to use the DataWarp space. Configurations are used in one of two ways:

- configurations are activated
- data is staged into or out of configurations

When either of these actions are performed, the action must supply a DataWarp session identifier in addition to other action-specific data such as a mount point. The session identifier is required because the DataWarp Service (DWS) keeps track of whether a configuration is used and which sessions used it. Then, when requested to remove either the configuration or session, the DWS cleans things up correctly.

The first time a configuration is used by a session, the DWS automatically creates a *registration* entry that binds together the session and configuration. The registration is automatically requested to be removed when either the linked configuration or session is requested to be removed. The actions performed at registration removal time depend on the type of configuration linked with the registration and the value of the registration's `wait` attribute. By default, `wait=true`, resulting in the execution of some configuration-specific actions prior to the complete removal of the registration.

DWS carries out the following operations for a registration based on the type of configuration with which it is linked:

- **scratch**
  1. Files marked for stage out by a user application (using `libdatawarp`) in a batch job with the `DW_STAGE_AT_JOB_END` stage type are transitioned to being immediately staged out.
  2. All existing stage out activity, including the stage out from the previous step, is allowed to fully complete.
- **cache**:
  1. All existing dirty data in the configuration is written out to the PFS.
- **swap**: no additional operations are carried out at registration removal time.

If the above processes are interrupted, e.g., a DataWarp server node crashes, the DWS attempts to restore everything associated with the node and restart the process after the node reboots. This includes restoring any logical volumes or mount points that are associated with the configuration.

There are times when the previous behavior is not desired. Consider either of the following:

- A DWS or underlying software bug exists that prevents the restoration of the DataWarp state on a crashed server node
- Hardware fails such that data on the SSD is permanently lost

In situations like this, set `wait=false` for the registration in order to tell the DWS to abort the normal cleanup process. For example, the following registration is in the process of being destroyed but cannot finish because a linked SSD has failed:

```
user@login> dwstat registrations
reg state sess conf wait
  2 D----    5   11 true
```

Instruct the DWS to abort the normal registration removal tasks by setting the `wait=false` with the following `dwcli` command:

```
user@login> dwcli update registration --id 2 --no-wait
```

WARNING: Use of `--no-wait` can lead to data loss because some data may not have been staged out to the PFS.

Workload Manager (WLM) Interaction with Registrations

Registration removal blocks batch job removal because the registration belongs to a session, which in turn belongs to a batch job. Each WLM provides its own way to force the removal of a batch job. Each of the DataWarp-integrated WLMs have been modified to automatically set the `wait` attribute of registrations to `false` when the WLM-specific job removal force option is used. It is only necessary to set `wait=false` using `dwcli` for registrations without a corresponding WLM batch job to force remove.

3.6 DataWarp Configuration Files and Advanced Settings

There are four DataWarp configuration files:

1. The scheduler daemon (`dwsd`) configuration file: `sdb:/etc/opt/cray/dws/dwsd.yaml`
2. The manager daemon (`dwmd`) configuration file: `sdb:/etc/opt/cray/dws/dwmd.yaml`
3. The DataWarp RESTful service (`dwrest`) configuration file: `api-gw:/etc/opt/cray/dws/dwrest.yaml`
4. The `dwrest` Gunicorn instance configuration file: `api-gw:/etc/opt/cray/dws/dwrestgun.conf`

Each file contains options that define limits, determine actions for different situations, and specify how to handle various requests. These options are considered *advanced* settings, and they are set with default values that are acceptable for most initial DataWarp configurations. Cray recommends not modifying the default values until there is a good understanding of the site's configuration and workload.

IMPORTANT: Do not directly modify any DataWarp configuration files (`dwsd.yaml`, `dwmd.yaml`, `dwrest.yaml`, `dwrestgun.conf`) as changes do not persist over a reboot. Modify the settings within

these files using the system configurator only; this ensures that the changes become part of the system config set.

At some point, an administrator very familiar with the site's configuration and usage history may want to modify one or more options to achieve a particular goal. For example, to allow the scheduler (`dwsd`) to grab more space than requested when processing instance create requests, the `equalize_fragments` option can be enabled using the system configurator. Before modifying any advanced setting, it is extremely important to understand the purpose of the setting, the format used to assign its value, and the impact of changing its value.

The configuration files contain descriptions for each setting. For example, the `equalize_fragments` setting is defined in `dwsd.yaml` as:

```
# Specifies whether the scheduler will attempt to create instances that are
# comprised of equal size fragments.  By default the scheduler will only pick
# as much space (roundup to pool granularity) as was requested at instance
# creation request time.  With this option, the scheduler will allot more space
# to instances in attempt to make all fragments within the instance be of equal
# size.  This can cause problems for workload managers but can provide for
# significantly improved performance to applications using DataWarp.
#
# equalize_fragments: no
```

In addition to the descriptions provided within the configuration files, certain topics within *XC™ Series DataWarp™ Installation and Administration Guide* contain more information about some advanced settings, and Cray support personnel are also an available resource.

Configuration File Formats

Within the YAML configuration files, options are set using the format: `option`: *`value`*. For example, the `expire_grace` option in `dwsd.yaml` is as follows:

```
# The length of time to allow an expired resource to linger before the scheduler
# automatically requests its destruction.
#
# expire_grace: 3600
```

Within the `dwrestgun.conf` file, options are set using the format: option=*`value`*. For example, `loglevel` accepts a quotation mark delimited text value, and `timeout` requires an integer value:

```
# the default logging level
loglevel = "info"
# Default timeout for a request, 10 minutes by default
timeout=600
```

CAUTION:

Advanced DataWarp settings must be modified with extreme care. The default values as released are acceptable for most installations. Sites that modify advanced settings are at risk of degrading DataWarp performance, decreasing SSD lifetime, and possibly other unknown outcomes. It is the administrator's responsibility to understand the purpose of any advanced settings changed, the formatting required, and the impact these changes may have.

Options incorrectly spelled or formatted are added but ignored, and the current value is not modified.

4 Post-boot Configuration

About this task

After the system boots, DataWarp requires further manual configuration. The steps required are:

Procedure

1. Over-provision all Intel P3608 cards.
2. Flash the firmware for all Fusion IO cards.
3. Initialize SSDs for use with the DataWarp service (DWS).
4. Create storage pools.
5. Assign nodes with space to a storage pool.

   IMPORTANT: Repeat these steps for each SSD-endowed node that DWS manages. i.e., those defined as `managed_nodes` during the DataWarp installation.
6. Verify the configuration.

4.1 Over-provision an Intel P3608 SSD

Prerequisites

- A Cray XC series system with one or more Intel P3608 SSD cards installed
- Ability to log in as `root`

About this task

This procedure is only valid for Intel P3608 SSDs.

WARNING: This procedure destroys any existing data on the SSDs.

Over-provisioning determines the size of the device available to the Logical Volume Manager (LVM) commands and needs to occur prior to executing any LVM commands. Typically, over-provisioning is done when the SSD cards are first installed.

TIP: Throughout these procedures, units of bytes are described using the binary prefixes defined by the International Electrotechnical Commission (IEC). For further information, see *Prefixes for Binary and Decimal Multiples* on page 71.

Procedure

1. Log in to an Intel P3608 SSD-endowed node as `root`.

   This example uses `nid00350`.

2. Remove any existing configuration.

   TIP: Numerous methods exist for creating configurations on an SSD; these instructions may not capture all possible cleanup techniques.

   a. Unmount file systems (if any).

   ```
   nid00350# df
   boot:/home                   20961280     11352064     9609216  55% /home
   tmp                       61504671488    624927640 57802802440   2% /scratch
   nid00350# umount -f /scratch
   ```

   b. Remove logical volumes (if any).

   ```
   nid00350# lvdisplay
     --- Logical volume ---
     LV Path                /dev/dwcache/s98i94f104o0
     LV Name                s98i94f104o0
     VG Name                dwcache
     LV UUID                910tio-RJXq-puYV-s3UL-yDM1-RoQl-HugeTM
     LV Write Access        read/write
     LV Creation host, time nid00350, 2016-02-22 13:29:11 -0500
     LV Status              available
     # open                 0
     LV Size                3.64 TiB
     Current LE             953864
     Segments               2
     Allocation             inherit
     Read ahead sectors     auto
     - currently set to     1024
     Block device           253:0

   nid00350# lvremove /dev/dwcache
   ```

   c. Remove volume groups (if any).

   ```
   nid00350# vgs
     VG       #PV #LV #SN Attr   VSize VFree
     dwcache    4   0   0 wz--n- 7.28t 7.28t
   nid00350# vgremove dwcache
     Volume group "dwcache" successfully removed
   ```

   d. Remove physical volumes (if any).

   ```
   nid00350# pvs
   PV           VG        Fmt Attr PSize PFree
   /dev/nvme0n1           lvm2 a--  1.82t 1.82t
   /dev/nvme1n1           lvm2 a--  1.82t 1.82t
   /dev/nvme2n1           lvm2 a--  1.82t 1.82t
   ```

```
/dev/nvme3n1          lvm2 a--  1.82t 1.82t

nid00350# pvremove /dev/nvme0n1 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1
  Labels on physical volume "/dev/nvme0n1" successfully wiped
  Labels on physical volume "/dev/nvme1n1" successfully wiped
  Labels on physical volume "/dev/nvme2n1" successfully wiped
  Labels on physical volume "/dev/nvme3n1" successfully wiped
```

e. Remove partitions for each device removed in the previous step (if any).

WARNING: This operation destroys any existing data on an SSD. Back up any existing data before proceeding.

```
nid00350# dd if=/dev/zero of=phys_vol bs=512 count=1
```

```
nid00350# dd if=/dev/zero of=/dev/nvme0n1 bs=512 count=1
nid00350# dd if=/dev/zero of=/dev/nvme1n1 bs=512 count=1
nid00350# dd if=/dev/zero of=/dev/nvme2n1 bs=512 count=1
nid00350# dd if=/dev/zero of=/dev/nvme3n1 bs=512 count=1
```

3. Reconfigure the device.

```
nid00350# module load linux-nvme-ctl
nid00350# nvme set-feature device -n 1 -f 0XC1 -v 3125623327
set-feature:193(Unknown), value:00000000
```

4. Confirm the change. Note that 0xba4d3a1f = 3125623327.

```
nid00350# nvme get-feature device -n 1 -f 0XC1 --sel=0
get-feature:193(Unknown), value:0xba4d3a1f
```

5. Repeat for all devices; for example: `/dev/nvme0` and `/dev/nvme1` on the first card and `/dev/nvme2` and `/dev/nvme3` on the second card (if installed).

6. Return to the SMW, and warm boot the node.

```
crayadm@smw> xtnmi cname
crayadm@smw> sleep 60
crayadm@smw> xtbootsys --reboot -r "warmboot for Intel SSD node" cname
```

7. Confirm that `SIZE` = `1600319143936` for all volumes.

```
nid00350# lsblk -b
NAME    MAJ:MIN RM          SIZE RO TYPE MOUNTPOINT
loop0     7:0    0        196608 0  loop /var/opt/cray/imps-distribution/squash/
loop1     7:1    0         65536 0  loop /var/opt/cray/imps-distribution/squash/
nvme0n1 259:0    0 1600319143936 0  disk
nvme1n1 259:1    0 1600319143936 0  disk
nvme2n1 259:2    0 1600319143936 0  disk
nvme3n1 259:3    0 1600319143936 0  disk
```

Contact Cray service personnel if `SIZE` is incorrect.

4.2 Update Fusion ioMemory Firmware

Prerequisites

- A Cray XC series system with one or more nodes with Fusion IO SSD hardware
- Initial installation of DataWarp is complete
- Identification of the nodes with Fusion IO SSD hardware

About this task

After the Fusion ioMemory VSL software is integrated in the FIO service node image (during the initial DataWarp installation), it is necessary to ensure that the Fusion ioMemory device firmware is up-to-date.

WARNING:

- It is extremely important that the power not be turned off during a firmware upgrade, as this could cause device failure
- Do not use this utility to downgrade the Fusion ioMemory device to an earlier version of the firmware. Doing so may result in data loss and void your warranty.

For further details, see *Fusion ioMemory™ VSL® 4.2.1 User Guide for Linux*.

Procedure

1. Log in to a node with Fusion IO SSD hardware installed and determine the firmware status.

   If the firmware needs to be updated, running the "fio-status -a" command displays the following message, “The firmware on this device is not compatible with the currently installed version of the driver,” and the device will not attach.

   ```
   nid00078# fio-status -a
   Found 1 VSL driver package:
      4.2.1 build 1137 Driver: loaded

   Found 1 ioMemory device in this system
   ...

   The firmware on this device is not compatible with the currently installed
   version of the driver
   ...

   There are active errors or warnings on this device!  Read below for details.
   ...

   This means the firmware needs to be updated.
   ```

2. Skip the remainder of this procedure if the firmware is compatible with the driver.

3. Determine the location of the `fio-firmware-fusion` file.

   ```
   nid00078# rpm -q --list fio-firmware-fusion
   /usr/share/doc/fio-firmware/copyright
   /usr/share/fio/firmware/fusion_4.2.1-20150611.fff
   ```

4. Update the firmware.

```
nid00078# fio-update-iodrive firmware-path
```

For example:

```
nid00078# fio-update-iodrive /usr/share/fio/firmware/fusion_4.2.1-20150611.fff
WARNING: DO NOT TURN OFF POWER OR RUN ANY IODRIVE UTILITIES WHILE THE FIRMWARE
UPDATE IS IN PROGRESS
Please wait...this could take a while

Updating: [====================] (100%)
fct0 - successfully updated the following:
Updated the firmware from 8.7.6 rev 20140819 to 8.9.1 rev 20150611
  Updated CONTROLLER from 8.7.6.117378 to 8.9.1.118126
  Updated SMPCTRL from 0.0.39 to 0.0.44
  Updated NCE from 0.1.4.672 to 1.0.8.100749

Reboot this machine to activate new firmware.
```

5. Repeat the previous steps for all nodes with Fusion IO SSD hardware.

6. Warm boot the node(s).

```
crayadm@smw> xtnmi cname
crayadm@smw> sleep 60
crayadm@smw> xtbootsys --reboot -r "warmboot for Fusion IO SSD node" cname
```

7. Log in to the node(s) and verify that the firmware update is recognized.

```
nid00078# fio-status -a
```

If no error messages are displayed, the node is ready for initialization. Otherwise, contact Cray service personnel.

4.3 Initialize an SSD

Prerequisites

- `root` privileges
- **Intel P3608 SSDs** must be over-provisioned
- Up-to-date firmware on **Fusion IO SSDs**

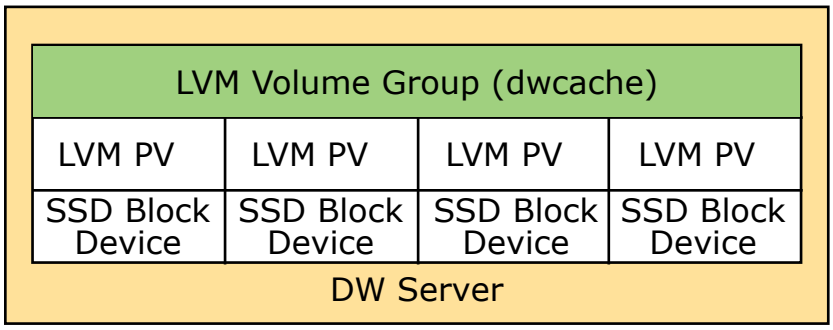
About this task

During the DataWarp installation process, the system administrator defines SSD-endowed nodes whose space the DataWarp service (DWS) will manage. This step ensures that the DataWarp manager daemon, `dwmd`, is started at boot time on these nodes. It **does not** prepare the SSDs for use with the DWS; this is performed manually using the following instructions.

After CLE boots, the following one-time manual device configuration must be performed **for each node** specified in the `managed_nodes` setting for the `cray_dws` service.

The diagram below shows how the Logical Volume Manager (LVM) volume group `dwcache` is constructed on each DW node. In this diagram, four SSD block devices have been converted to LVM physical devices with the `pvcreate` command. These four LVM physical volumes were combined into the LVM volume group `dwcache` with the `vgcreate` command.

Figure 10. LVM Volume Group



TIP: Throughout these procedures, units of bytes are described using the binary prefixes defined by the International Electrotechnical Commission (IEC). For further information, see *Prefixes for Binary and Decimal Multiples* on page 71.

Procedure

1. Log in to an SSD-endowed node as `root`.

   This example uses `nid00350`.

2. Identify the SSD block devices.

   NVMe SSDs:

   ```
   nid00350# lsblk
   NAME    MAJ:MIN RM   SIZE RO MOUNTPOINT
   loop0     7:0    0        196608 0  loop /var/opt/cray/imps-distribution/squash/
   loop1     7:1    0         65536 0  loop /var/opt/cray/imps-distribution/squash/
   nvme0n1 259:0    0 1600319143936 0  disk
   nvme1n1 259:1    0 1600319143936 0  disk
   nvme2n1 259:2    0 1600319143936 0  disk
   nvme3n1 259:3    0 1600319143936 0  disk
   ```

   Fusion IO SSDs:

   ```
   nid00350# lsblk
   NAME    MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
   fioa    254:0    0   2.9T  0 disk
   fiob    254:1    0   2.9T  0 disk
   loop0     7:0    0 196608  0 /var/opt/cray/impsdistribution/squash/
   loop1     7:1    0  65536  0 /var/opt/cray/impsdistribution/squash/
   ```

3. (Intel P3608 SSDs only) Proceed to step *5* on page 34 as the following step was completed during the over-provisioning procedure.

4. (All non-Intel P3608 SSDs) Remove any existing configuration.

TIP: Numerous methods exist for creating configurations on an SSD; these instructions may not capture all possible cleanup techniques.

a. Unmount file systems (if any).

```
nid00350# df
boot:/home                    20961280     11352064     9609216  55% /home
tmp                        61504671488    624927640 57802802440   2% /scratch
nid00350# umount -f /scratch
```

b. Remove logical volumes (if any).

```
nid00350# lvdisplay
  --- Logical volume ---
  LV Path                /dev/dwcache/s98i94f104o0
  LV Name                s98i94f104o0
  VG Name                dwcache
  LV UUID                910tio-RJXq-puYV-s3UL-yDM1-RoQl-HugeTM
  LV Write Access        read/write
  LV Creation host, time nid00350, 2016-02-22 13:29:11 -0500
  LV Status              available
  # open                 0
  LV Size                2.92 TiB
  Current LE             953864
  Segments               2
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     1024
  Block device           253:0

nid00350# lvremove /dev/dwcache
```

c. Remove volume groups (if any).

```
nid00350# vgs
  VG       #PV #LV #SN Attr   VSize VFree
  dwcache    2   0   0 wz--n- 2.92t 2.92t
nid00350# vgremove dwcache
  Volume group "dwcache" successfully removed
```

d. Remove physical volumes (if any).

```
nid00350# pvs
PV           VG        Fmt Attr PSize PFree
/dev/fioa              lvm2 a--  1.46t 1.46t
/dev/fiob              lvm2 a--  1.46t 1.46t

nid00350# pvremove /dev/fioa /dev/fiob
  Labels on physical volume "/dev/fioa" successfully wiped
  Labels on physical volume "/dev/fiob" successfully wiped
```

e. Remove partitions for each device removed in the previous step (if any).

WARNING: This operation destroys any existing data on an SSD. Back up any existing data before proceeding.

```
nid00350# dd if=/dev/zero of=phys_vol bs=512 count=1

nid00350# dd if=/dev/zero of=/dev/fioa bs=512 count=1
nid00350# dd if=/dev/zero of=/dev/fiob bs=512 count=1
```

5. Initialize each physical device for later use by LVM. Note that Cray currently sells systems with 1, 2, or 4 physical devices on a node.

   WARNING: This operation destroys any existing data on an SSD. Back up any existing data before proceeding.

   ```
   nid00350# pvcreate phys_vol [phys_vol...]
   ```

   NVMe SSDs:

   ```
   nid00350# pvcreate /dev/nvme0n1 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1
   Physical volume "/dev/nvme0n1" successfully created
   Physical volume "/dev/nvme1n1" successfully created
   Physical volume "/dev/nvme2n1" successfully created
   Physical volume "/dev/nvme3n1" successfully created
   ```

   FIO SSDs:

   ```
   nid00350# pvcreate /dev/fioa /dev/fiob
   Physical volume "/dev/fioa" successfully created
   Physical volume "/dev/fiob" successfully created
   ```

6. Create an LVM volume group called `dwcache` that uses these physical devices.

   Requirements for the LVM physical volumes specified are:

   - Any number of physical devices may be specified.
   - Each physical volume specified **must be** the exact same size.
     - To verify physical volume size, execute the command: `pvs --units b` and examine the `PSize` column of the output.

       TIP: If sizes differ between physical volumes, it is likely that either an over-provisioning step was forgotten or there is mixed hardware on the node. Address this issue before proceeding.

   ```
   nid00350# vgcreate dwcache phys_vol [phys_vol...]
   ```

   NVMe SSDs:

   ```
   nid00350# vgcreate dwcache /dev/nvme0n1 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1
   Volume group "dwcache" successfully created
   ```

   FIO SSDs:

   ```
   nid00350# vgcreate dwcache /dev/fioa /dev/fiob
   Volume group "dwcache" successfully created
   ```

7. Restart the `dwmd` service.

   For example:

   ```
   nid00350# service dwmd stop
   nid00350# service dwmd restart
   ```

8. Verify that DWS recognizes the node with storage.

   NVMe SSDs:

```
nid00350# module load dws
nid00350# dwstat nodes
    node  pool online drain  gran capacity insts activs
nid00350    -    true false  8MiB  3.64TiB     0      0
```

FIO SSDs:

```
nid00350# module load dws
nid00350# dwstat nodes
    node  pool online drain  gran capacity insts activs
nid00350    -    true false  4MiB  2.92TiB     0      0
```

4.4 Create a Storage Pool

A storage pool groups nodes with storage together such that requests for space made against the pool are fulfilled from the nodes associated with the pool with a common allocation granularity. Pools have either byte or node allocation granularity (*pool_AG*). This release of DataWarp only supports byte allocation granularity. There are tradeoffs in picking allocation granularities too small or too large.

Choosing a pool allocation granularity equal to the capacity of each node prevents sharing of nodes across usages of DataWarp. However, application performance is more deterministic as bandwidth is dedicated and unrelated usages of DataWarp are less likely to perturb each other. Further, requests for a small amount of capacity will be placed on only one server node.

Picking a small pool allocation granularity allows for higher utilization of DataWarp resources and potentially greatly increased performance, but at the cost of less deterministic and potentially worse performance. With a small pool allocation granularity, each usage of DataWarp capacity is more likely to be spread out over more or all DataWarp servers. If only one application is actively performing I/O to DataWarp, then it gets all of the bandwidth provided by all of the servers. Multiple applications performing I/O concurrently to the same DataWarp servers will share the available bandwidth. Finally, even requests for a small amount of capacity are more likely to be placed on multiple server nodes.

Considerations When Determining Pool Allocation Granularity

Determining an optimal pool allocation granularity for a system is a function of several factors, including the number of SSD nodes, the number of SSD cards per node, the size of the SSDs, as well as software requirements, limitations, and bugs. Therefore, the best value is site specific and likely to change over time.

All pools must meet the following requirements:

- The byte-oriented allocation granularity for a pool must be at least 16MiB.
- Each node's volume group (`dwcache`, configured in *Initialize an SSD* on page 31) has a Physical Extent size (*PE_size*) and Physical Volume count (*PV_count*). The default *PE_size* is 4MiB, and *PV_count* is equal to the number of Physical Volumes specified during volume group creation. The DataWarp service (DWS) places the following restriction on nodes associated with a pool:
    - A node can only be associated with a storage pool if the node's granularity (*PE_size* * *PV_count*) is a factor of the pool's allocation granularity (*pool_AG*). The `dwstat nodes` command lists the node's granularity in the `gran` column.

Ideally, a pool's allocation granularity is defined as a factor of the aggregate space of each node within the pool; otherwise, some space is not usable and, therefore, is wasted. For example, if a node contributes 6.4TiB of capacity, but the pool allocation granularity is 1TiB, then 0.4TiB of capacity is wasted per node.

With this release of DataWarp, valid pool granularities are those that meet the above requirements. For most installations, this means that the pool allocation granularity must be a multiple of 16MiB. Choosing such a pool allocation granularity will result in a functioning, but perhaps non-optimal, DataWarp environment.

How DataWarp Interacts with Pool Allocation Granularity

The default behavior of DataWarp leads to space allocation imbalances that can lead to poor performance. A request for space through DataWarp is called an instance. When DataWarp processes an instance create request, the request is fulfilled by carving out some capacity from one or more nodes. Each piece of the instance is called a fragment. If even one fragment of an instance is sized differently from other fragments in the instance, the result can be greatly reduced performance. By default, DataWarp creates fragments such that only as much space as was requested, rounded up to the nearest multiple of the pool allocation granularity, is taken to satisfy the request. This frequently leads to at least one instance fragment being sized differently from all other fragments. The DataWarp scheduler daemon option `equalize_fragments` in `dwsd.yaml` adjusts scheduler behavior to attempt to avoid the situation. With `equalize_fragments`, the scheduler is given the freedom to grab more space than requested when processing instance create requests. The option is off by default, because not every Workload Manager functions correctly with it on. See *Modify DWS Advanced Settings* on page 41 for details on enabling `equalize_fragments`.

Certain usages of DataWarp have limitations where having too small of a pool allocation granularity can lead to situations where not all capacity requested is accessible by that usage. For scratch usages of DataWarp, any instance consisting of more than 4096 allocation granularities is not guaranteed to have all of its space usable by the scratch usage. The scratch usage is still functional, but not as much data may be able to be written to it as expected. Requesting more space than is strictly necessary helps to alleviate the problem. If having the guarantee is important, the `dwpoolhelp` command can suggest pool allocation granularity values for a particular system that provide the guarantee.

The `dwpoolhelp` command

The `dwpoolhelp` command is only needed for DataWarp configurations of type scratch when the administrator wants to guarantee that all of the requested space is usable for scratch usages. `dwpoolhelp` calculates and displays pool allocation granularity values for a range of node granularity units along with waste per node and waste per pool values in bytes.

```
== Optimal pool granularities per granules per node ==
Gran / node     Pool granularity  Waste per node       Waste per pool
          1        1599992758272         4194304              8388608
          2         799987990528        20971520             41943040
          3         533330919424         4194304              8388608
          4         399985606656        54525952            109051904
          5         319991840768        37748736             75497472
        ...
```

As mentioned earlier, the best pool allocation granularity value is site specific; therefore, when using `dwpoolhelp`, there are no set guidelines for choosing the best value. In general, the goal is to pick a pool allocation granularity value that minimizes the `Waste per pool` value, although this might not be the case if smaller pool granularity is important. *Example 2: Create a storage pool using dwpoolhelp to first determine allocation granularity* on page 37 shows the `dwpoolhelp` output sorted by amount of waste per node. See the dwpoolhelp(8) man page for further details.

Recommendations

Taken together, Cray recommends the following:

- For performance reasons, create pools that only contain nodes with homogeneous SSD hardware.
- If possible, turn on the `equalize_fragments` `dwsd.yaml` option. See *Modify DWS Advanced Settings* on page 41.
- Use the smallest possible pool allocation granularity, which is typically 16MiB.
- If having the guarantee of being able to access all requested space for scratch usages of DataWarp is important, use the `dwpoolhelp` command to pick an alternative pool allocation granularity. Note that `dwpoolhelp` assumes a homogeneous system and is not likely to calculate pool allocation granularity values that will provide optimal performance for systems with more than one type of SSD installed.
    - Do not use `dwpoolhelp` for pools with only one node. Single node pools should always use a pool granularity value of 16MiB or the node granularity (`gran` column in `dwstat nodes -b`), whichever is greater.

Example 1: Create a storage pool with allocation granularity = 16MiB.

As a DataWarp administrator logged on to a CLE service node:

```
crayadm@login> module load dws
crayadm@login> dwcli create pool --name wlm_pool --granularity 16777216
created pool id wlm_pool

# verify the pool was created
crayadm@login> dwstat pools
    pool  unit quantity free   gran
wlm_pool bytes        0    0  16MiB
```

Example 2: Create a storage pool using `dwpoolhelp` to first determine allocation granularity

IMPORTANT: Note that this is for example purposes only; optimal pool allocation granularity is site specific.

As a DataWarp administrator logged on to a CLE service node:

```
# Determine node capacity and allocation granularity
crayadm@login> dwstat -b nodes
    node     pool online drain    gran      capacity insts activs
nid00028        -   true false 8388608 4000795590656     0      0
nid00029        -   true false 8388608 4000795590656     0      0
nid00089        -   true false 8388608 4000795590656     0      0
nid00090        -   true false 8388608 4000795590656     0      0

# Use the above information with the dwpoolhelp command
crayadm@login> dwpoolhelp -n 4 -g 8388608 -c 4000795590656
== Starting Values ==
Number of nodes: 4
Node capacity: 4000795590656
Allocation granularity on nodes: 8388608
Using 16777216 bytes for actual allocation granularity on nodes to satisfy XFS requirements

== Calculating maximum granules per node ==
Max number of granules in an instance while still being able to access all capacity is 4096
floor(max_stripes / nodes) -> floor(4096 / 4) = 1024Maximum granules per node: 409
```

```
== Optimal pool granularities per granules per node ==
Gran / node     Pool granularity  Waste per node       Waste per pool
          1        4000795590656               0                    0
          2        2000397795328               0                    0
          3        1333587345408        33554432            134217728
          4        1000190509056        33554432            134217728
          5         800155762688        16777216             67108864
        ...

# Too much output to weed through, sort on 'Waste per pool'
crayadm@login> dwpoolhelp -n 4 -g 8388608 -c 4000795590656 | egrep '^ |Gran' | sort -bg --
key=3
Gran / node     Pool granularity  Waste per node       Waste per pool
          1        4000795590656               0                    0
          2        2000397795328               0                    0
        185          21625831424        16777216             67108864
         37         108129157120        16777216             67108864
          5         800155762688        16777216             67108864
        108          37044092928        33554432            134217728
        ...

# Create a pool with 185 granularities per node
crayadm@login> dwcli create pool --name wlm_pool2 --granularity 21625831424
created pool id wlm_pool2

# verify the pool was created
crayadm@login> dwstat pools
     pool  unit quantity free     gran
 wlm_pool bytes        0    0    16MiB
wlm_pool2 bytes        0    0  20.1GiB
```

4.5 Assign a Node to a Storage Pool

Prerequisites

- At least one storage pool exists
- At least one SSD is initialized for use with the DataWarp service (DWS)
- DataWarp administrator privileges

About this task

Follow this procedure to associate an SSD-endowed node with an existing storage pool.

Procedure

1. Log in to a booted CLE service node as a DWS administrator.
2. Load the DataWarp Service module.

   ```
   crayadm@login> module load dws
   ```

3. Associate an SSD-endowed node with a storage pool.

   ```
   crayadm@login> dwcli update node --name hostname --pool pool_name
   ```

For example, to associate a node (hostname `nid00349`) with a storage pool called `wlm_pool`:

```
crayadm@login> dwcli update node --name nid00349 --pool wlm_pool
```

The association may fail. If it does, ensure that the pool exists (`dwstat pools`) and that the node's granularity (`dwstat nodes -b`) is a factor of the pool's granularity (`dwstat pools -b`).

4. Verify the node is associated with the pool.

```
crayadm@login> dwstat pools nodes
    pool units quantity    free    gran
wlm_pool bytes  3.64TiB 3.64TiB  910GiB

    node     pool online  drain  gran   capacity insts activs
nid00349 wlm_pool   true  false  8MiB    3.64TiB     0      0
```

4.6 Verify the DataWarp Configuration

Prerequisites

- At least one SSD node is assigned to a storage pool
- DataWarp administrator privileges

About this task

There are a few ways to verify that the DataWarp configuration is as desired.

Procedure

1. Log in to a booted service node and load the DataWarp Service (DWS) module.

```
crayadm@login> module load dws
```

2. Request status information about DataWarp resources.

```
crayadm@login> dwstat pools nodes
    pool units quantity    free    gran
wlm_pool bytes   4.0TiB 3.38TiB  128GiB

    node     pool online drain  gran   capacity  insts  activs
nid00065 wlm_pool   true false 16MiB       1TiB      1       0
nid00066 wlm_pool   true false 16MiB       1TiB      0       0
nid00069 wlm_pool   true false 16MiB       1TiB      0       0
nid00070 wlm_pool   true false 16MiB       1TiB      0       0
nid00004        -   true false 16MiB    3.64TiB      0       0
nid00005        -   true false 16MiB    3.64TiB      0       0
```

3. Check the following combinations for each row.
   - If **pool** is – and **capacity** ≠ `0`: This is a server node that has not yet been associated with a storage pool. See *Assign a Node to a Storage Pool* on page 38.

- If **pool** is `-` and **capacity** is `0`: This is a non-server node (e.g., client/compute) and does not need to be associated with a storage pool.
- If **pool** is *`something`* and **capacity** ≠ `0`: This is a server node that belongs to the pool called <something>.
- If **pool** is *`something`* and **capacity** is `0`: This is a non-server node that belongs to a pool. Since the non-server node contributes no space to the pool, this association is not necessary but harmless.

This completes the process to configure DataWarp with DWS. Refer to the site-specific workload manager (WLM) documentation for further configuration steps to integrate the WLM with Cray DataWarp.

5 DataWarp Administrator Tasks

5.1 Modify DWS Advanced Settings

Prerequisites

- Ability to log in to the SMW as `root`

About this task

This procedure describes how to modify DataWarp advanced settings using the system configurator. Cray recommends that DataWarp administrators have a thorough understanding of these settings before modifying them. *DataWarp Configuration Files and Advanced Settings* on page 25 provides an overview of the advanced settings and the configuration files in which they are defined. The configuration files include a description of each setting, and various topics within the *XC™ Series DataWarp™ Installation and Administration Guide* provide additional information for certain advanced settings.

CAUTION:

Advanced DataWarp settings must be modified with extreme care. The default values as released are acceptable for most installations. Sites that modify advanced settings are at risk of degrading DataWarp performance, decreasing SSD lifetime, and possibly other unknown outcomes. It is the administrator's responsibility to understand the purpose of any advanced settings changed, the formatting required, and the impact these changes may have.

Options incorrectly spelled or formatted are added but ignored, and the current value is not modified.

Procedure

1. Invoke the configurator to access the advanced DataWarp settings.

   The configurator displays the basic settings, as defined during the initial installation, as well as **some** of the advanced DataWarp settings. All advanced DataWarp settings are modifiable whether or not they are displayed by the configurator. Any advanced settings that have been modified through the configurator are displayed, and some settings that are currently set to their default values are also displayed.

```
smw# cfgset update -m interactive -l advanced -s cray_dws p0_example
Service Configuration Menu (Config Set: p0_example, type: cle)

  cray_dws        [ status: enabled ]  [ validation: valid ]

------------------------------------------------------------------------------------------------------------
  Selected      #      Settings                            Value/Status (level=advanced)
------------------------------------------------------------------------------------------------------------
                       service
                1)       scheduler_node                    sdb
                2)       managed_nodes                     c0-0c0s7n2, c0-0c0s0n2
```

```
                3)       api_gateway_nodes                  c0-0c0s1n1
                4)       external_api_gateway_hostnames     (none)
                5)       dwrest_cacheroot_whitelist         /lus/scratch
                6)       allow_dws_cli_from_computes        False
                7)       lvm_issue_discards                 0

                       dwmd
                8)       dwmd_conf                          iscsi_initiator_cred_path:
                                                            /etc/opt/cray/dws/iscsi_target_secret,
                                                            iscsi_target_cred_path:
                                                            /etc/opt/cray/dws/iscsi_initiator_secret,
                                                            capmc_os_cacert:
                                                            /etc/pki/trust/anchors/certificate_authority.pem

                       dwsd
                9)       dwsd_conf                          log_mask: 0x7, instance_optimization_default:
                                                            bandwidth, scratch_limit_action: 0x3

                       dwrest
               10)       dwrest_conf                        port: 2015

                       dwrestgun
               11)       dwrestgun_conf                     max_requests=1024

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

2. Proceed based on the setting to be modified:

- To modify a displayed setting, proceed to step *3* on page 42.
- To modify a non-displayed setting, proceed to step *4* on page 43.
- To reset a displayed setting to its default value, proceed to step *5* on page 43.

3. Modify a displayed setting:

This example changes the value of `instance_optimization_default` within the `dwsd` settings.

a. Select `dwsd`.

> **TIP:** The configurator uses index numbering to identify configuration items. This numbering may vary; the value used in the examples may not be correct for all systems. The user must search the listing displayed on the screen to determine the correct index number for the service/setting being configured.

```
Cray dws Menu [default: save & exit - Q] $ 9
Cray dws Menu [default: configure - C] $ C

************************ cray_dws.settings.dwsd.data.dwsd_conf **************************

      dwsd_conf -- dwsd Config
        Internal dwsd settings.  Change with extreme caution. See
        /etc/opt/cray/dws/dwsd.yaml for variables and syntax.

        Default:                          Current:
            1) log_mask: 0x7                                  1) log_mask: 0x7
            2) instance_optimization_default: bandwidth       2) instance_optimization_default: bandwidth
            3) scratch_limit_action: 0x3                      3) scratch_limit_action: 0x3
...
```

b. Choose to change the entry for `instance_optimization_default`, enter the new value, and then set the entries.

Use the format `#*` to indicate which entry to change, where `#` is the index number for `instance_optimization_default`.

```
cray_dws.settings.dwsd.data.dwsd_conf
[<cr>=set 3 entries, +=add an entry, ?=help, @=less] $ 2*
Modify dwsd_conf:instance_optimization_default: bandwidth (Ctrl-d to exit) $ wear
...
        Default:                          Current:
            1) log_mask: 0x7                                  1) log_mask: 0x7
            2) instance_optimization_default: bandwidth       2) instance_optimization_default: wear
            3) scratch_limit_action: 0x3                      3) scratch_limit_action: 0x3
...
cray_dws.settings.dwsd.data.dwsd_conf
[<cr>=set 3 entries, +=add an entry, ?=help, @=less] $ <cr>
```

The configurator displays all `cray_dws` basic and visible advanced settings (as in step 1), including the new value for `instance_optimization_default`.

c. Continue to modify advanced settings if desired; otherwise, proceed to step *6* on page 44.

4. Modify a non-displayed setting:

This example enables the `equalize_fragments` option that is defined in the `dwsd` settings.

a. Select `dwsd_conf`.

```
Cray dws Menu [default: save & exit - Q] $ 9
Cray dws Menu [default: configure - C] $ C

************************ cray_dws.settings.dwsd.data.dwsd_conf *************************

      dwsd_conf -- dwsd Config
        Internal dwsd settings.  Change with extreme caution. See
        /etc/opt/cray/dws/dwsd.yaml for variables and syntax.

        Default:                          Current:
            1) log_mask: 0x7                                  1) log_mask: 0x7
            2) instance_optimization_default: bandwidth       2) instance_optimization_default: bandwidth
            3) scratch_limit_action: 0x3                      3) scratch_limit_action: 0x3
...
```

b. Choose to add entries.

```
cray_dws.settings.dwsd.data.dwsd_conf
[<cr>=set 3 entries, +=add an entry, ?=help, @=less] $ +
```

c. Enter the setting(s).

> **TIP:** Use the format: *option*: *value* to set a value for `dwsd_conf`, `dwmd_conf`, and `dwsrest_conf` options. Use the format: *option*=*value* for `dwrestgun_conf` options, with text values delimited by quotation marks.

```
Add dwsd_conf (Ctrl-d to exit) $ equalize_fragments: yes
Add dwsd_conf (Ctrl-d to exit) $ <Ctrl-d>

****************************** cray_dws.settings.dwsd.data.dwsd_conf *******************************

      dwsd_conf -- dwsd Config
        Internal dwsd settings.  Change with extreme caution. See
        /etc/opt/cray/dws/dwsd.yaml for variables and syntax.

        Default:                                                  Current:
            1) log_mask: 0x7                                          1) log_mask: 0x7
            2) instance_optimization_default: bandwidth               2) instance_optimization_default: bandwidth
            3) scratch_limit_action: 0x3                              3) scratch_limit_action: 0x3
                                                                      4) equalize_fragments: yes
...
cray_dws.settings.dwsd.data.dwsd_conf
[<cr>=set 4 entries, +=add an entry, ?=help, @=less] $ <cr>
```

The configurator displays all `cray_dws` basic and visible advanced settings (as in step 1), including the new value for `equalize_fragments`.

d. Continue to modify advanced settings if desired; otherwise, proceed to step *6* on page 44.

5. Reset a displayed setting to its default value:

This example resets `equalize_fragments` to its default value.

a. Select `dwsd_conf`.

```
Cray dws Menu [default: save & exit - Q] $ 9
Cray dws Menu [default: configure - C] $ C

****************************** cray_dws.settings.dwsd.data.dwsd_conf *******************************
...
        Default:                                                  Current:
            1) log_mask: 0x7                                          1) log_mask: 0x7
```

```
           2) instance_optimization_default: bandwidth            2) instance_optimization_default: bandwidth
           3) scratch_limit_action: 0x3                           3) scratch_limit_action: 0x3
                                                                  4) equalize_fragments: yes
...
```

b. Remove the `equalize_fragments` setting.

Use the format `#-` to remove a setting, thereby resetting it to its default value.

```
cray_dws.settings.dwsd.data.dwsd_conf
[<cr>=set 4 entries, +=add an entry, ?=help, @=less] $ 4-

|--- Information
|  *    Entry 'equalize_fragments: yes' removed successfully. Press <cr> to set.
|---

cray_dws.settings.dwsd.data.dwsd_conf
[<cr>=set 3 entries, +=add an entry, ?=help, @=less] $ <cr>


Service Configuration Menu (Config Set: p0_example, type: cle)

  cray_dws        [ status: enabled ]  [ validation: valid ]

-------------------------------------------------------------------------------------------------------------
  Selected      #      Settings                              Value/Status (level=advanced)
-------------------------------------------------------------------------------------------------------------
                       service
                 1)      scheduler_node                      sdb
                 2)      managed_nodes                       c0-0c0s7n2, c0-0c0s0n2
                 3)      api_gateway_nodes                   c0-0c0s1n1
                 4)      external_api_gateway_hostnames      (none)
                 5)      dwrest_cacheroot_whitelist          /lus/scratch
                 6)      allow_dws_cli_from_computes         False
                 7)      lvm_issue_discards                  0

                       dwmd
                 8)      dwmd_conf                           iscsi_initiator_cred_path:
                                                             /etc/opt/cray/dws/iscsi_target_secret,
                                                             iscsi_target_cred_path:
                                                             /etc/opt/cray/dws/iscsi_initiator_secret,
                                                             capmc_os_cacert:
                                                             /etc/pki/trust/anchors/certificate_authority.pem

                       dwsd
                 9)      dwsd_conf                           log_mask: 0x7, instance_optimization_default:
                                                             bandwidth, scratch_limit_action: 0x3

                       dwrest
                10)      dwrest_conf                         port: 2015

                       dwrestgun
                11)      dwrestgun_conf                      max_requests=2048

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

c. Continue to modify advanced settings if desired; otherwise, proceed to the next step.

6. Save and exit the configurator.

```
Cray dws Menu [default: save & exit - Q] $ Q
```

7. Activate the changes on all appropriate nodes. Proceed based on the configuration files modified.

- `dwsd_conf`: execute the following commands on the scheduler node.

```
sdb# /etc/init.d/cray-ansible start
sdb# systemctl reload dwsd
```

- `dwmd_conf`: execute the following commands on all DataWarp managed nodes.

```
nid# /etc/init.d/cray-ansible start
nid# systemctl reload dwmd
```

- `dwrest_conf` or `dwrestgun_conf`: execute the following commands on the API gateway node.

```
api-gw# /etc/init.d/cray-ansible start
api-gw# systemctl reload dwrest
api-gw# systemctl reload nginx
```

5.2 Configure SSD Protection Settings

Prerequisites

- Ability to log in as `root`
- Read *Modify DWS Advanced Settings* on page 41

About this task

The possibility exists for a user program to unintentionally cause excessive activity to SSDs, and thereby diminish the lifetime of the devices. To mitigate this issue, DataWarp includes both administrator-defined configuration options and user-specified job script command options that help the DataWarp service (DWS) detect when a program's behavior is anomalous and then react based on configuration settings.

This procedure describes how to modify the administrator-defined SSD protection settings using the system configurator. For user-defined settings, see the *XC™ Series DataWarp™ User Guide*.

> **IMPORTANT:** Do not directly modify any DataWarp configuration files (`dwsd.yaml`, `dwmd.yaml`, `dwrest.yaml`, `dwrestgun.conf`) as changes do not persist over a reboot. Modify the settings within these files using the system configurator only; this ensures that the changes become part of the system config set.

Protection Settings within the `dwsd` Configuration File

The DataWarp scheduler daemon (`dwsd`) configuration file (`sdb:/etc/opt/cray/dws/dwsd.yaml`) contains options for the following DataWarp SSD protection features:

- Action upon error
- Write tracking
- File creation limits
- File size limits

The configurable SSD protection options are:

`cache_limit_action`

The action to take when one of the cache limits is exceeded; this action applies to all limits. Default: 0x3

0x1: log only
0x2: error on file system operations
0x3: log and error

`cache_max_file_size_default`

The maximum size (bytes) of any one file that may exist in a cache configuration. In other words, the maximum byte offset for a file that may be read from or written to. When the threshold is exceeded, a message displays on the system console and an error is reported

back to the file system operation that triggered the limit. A value of 0 means no limit. User requests may override this value. Default: 0

cache_max_file_size_max

The maximum value a user may request when overriding `cache_max_file_size_default`. The value of 0 means there is no max. Default: 0

instance_write_window_length_default

The default number of seconds used when calculating the simple moving average of writes to an instance. Note that the configurations using the instance must provide support for this (e.g., does not apply to swap). A value of 0 means the write window limit is not used. Default: 86400

instance_write_window_length_max

The maximum value a user may request when overriding `instance_write_window_length_default`. A value of 0 means there is no maximum. Default: 0

instance_write_window_length_min

The minimum value a user may request when overriding `instance_write_window_length_default`. A value of 0 means there is no minimum. Default: 0

instance_write_window_multiplier_default

The default multiplier to use against an instance size to determine the maximum number of bytes written portion of the moving average calculation for purposes of detecting anomalous write behavior. The multiplier must be an integer of 0 or more. A value of 0 means the write window limit is not used. For example, if the multiplier is 10, the instance size is 2 TiB, and the write window is 86400, then 20 TiB may be written to the instance in any 24 hour sliding window. Default: 10

instance_write_window_multiplier_max

The maximum value a user may request when overriding `instance_write_window_multiplier_default`. A value of 0 means there is no maximum. Default: 0

scratch_limit_action

The action to take when one of the scratch limits is exceeded; this action applies to all limits. Default: 0x3

0x1: log only
0x2: error on file system operations
0x3: log and error

scratch_namespace_max_files_default

The maximum number of files that may be created in a scratch configuration namespace. When the threshold is exceeded, a message displays on the system console and no new files can be created in the namespace. A value of 0 means no limit. User requests may override this value. Default: 0

scratch_namespace_max_files_max

The maximum value a user may request when overriding `scratch_namespace_max_files_default`. A value of 0 means no limit. Default: 0

`scratch_namespace_max_file_size_default`

The maximum size (bytes) of any one file that may exist in a scratch configuration namespace. When the threshold is exceeded, a message displays on the system console and an error is reported back to the file system operation that triggered the limit. A value of 0 means no limit. User requests may override this value. Default: 0

`scratch_namespace_max_file_size_max`

The maximum value a user may request when overriding `scratch_namespace_max_file_size_default`. The value of 0 means there is no maximum. Default: 0

The administrator selects default values, min/max user limits, and the action taken when a limit is exceeded. The options within `/etc/opt/cray/dws/dwsd.yaml` are considered *advanced* options and must be modified with extreme caution. This procedure describes how to modify these advanced settings using the system configurator.

CAUTION:

Advanced DataWarp settings must be modified with extreme care. The default values as released are acceptable for most installations. Sites that modify advanced settings are at risk of degrading DataWarp performance, decreasing SSD lifetime, and possibly other unknown outcomes. It is the administrator's responsibility to understand the purpose of any advanced settings changed, the formatting required, and the impact these changes may have.

Options incorrectly spelled or formatted are added but ignored, and the current value is not modified.

For further details, see *Modify DWS Advanced Settings* on page 41.

Procedure

1. Invoke a configurator session to modify `cray_dws` advanced settings.

2. Select `dwsd_conf`.

   TIP: The configurator uses index numbering to identify configuration items. This numbering may vary; the value used in the examples may not be correct for all systems. The user must search the listing displayed on the screen to determine the correct index number for the service/setting being configured.

```
Cray dws Menu [default: save & exit - Q] $ 9
Cray dws Menu [default: configure - C] $ C
**************************** cray_dws.settings.dwsd.data.dwsd_conf *******************************

      dwsd_conf -- dwsd Config
        Internal dwsd settings.  Change with extreme caution. See
        /etc/opt/cray/dws/dwsd.yaml for variables and syntax.

        Default:                                                Current:
            1) log_mask: 0x7                                        1) log_mask: 0x7
            2) instance_optimization_default: bandwidth             2) instance_optimization_default: bandwidth
            3) scratch_limit_action: 0x3                            3) scratch_limit_action: 0x3

...
```

   Only a sampling of the `dwsd_conf` settings are displayed, although all settings are modifiable.

3. Modify one or more settings.

This example demonstrates how to modify the `scratch_namespace_max_files_default` and `scratch_namespace_max_files_max` settings.

```
cray_dws.settings.dwsd.data.dwsd_conf
[<cr>=set 3 entries, +=add an entry, ?=help, @=less] $ +
Add dwsd_conf (Ctrl-d to exit) $ scratch_namespace_max_files_default: 10000
Add dwsd_conf (Ctrl-d to exit) $ scratch_namespace_max_files_max: 30000
Add dwsd_conf (Ctrl-d to exit) $<Ctrl-d>

cray_dws.settings.dwsd.data.dwsd_conf
[<cr>=set 3 entries, +=add an entry, ?=help, @=less] $ +
Add dwsd_conf (Ctrl-d to exit) $ scratch_namespace_max_files_default: 10000
Add dwsd_conf (Ctrl-d to exit) $ scratch_namespace_max_files_max: 30000
Add dwsd_conf (Ctrl-d to exit) $
****************************** cray_dws.settings.dwsd.data.dwsd_conf ******************************

      dwsd_conf -- dwsd Config
        Internal dwsd settings.  Change with extreme caution. See
        /etc/opt/cray/dws/dwsd.yaml for variables and syntax.

        Default:                                                Current:
            1) log_mask: 0x7                                        1) log_mask: 0x7
            2) instance_optimization_default: bandwidth             2) instance_optimization_default: bandwidth
            3) scratch_limit_action: 0x3                            3) scratch_limit_action: 0x3
                                                                    4) scratch_namespace_max_files_default: 10000
                                                                    5) scratch_namespace_max_files_max: 30000
...
cray_dws.settings.dwsd.data.dwsd_conf
[<cr>=set 5 entries, +=add an entry, ?=help, @=less] $ <cr>

# Complete cray_dws displayed here with new settings included
...
Cray dws Menu [default: save & exit - Q] $ Q
INFO -
...

INFO - ConfigSet 'p0' has been updated.
INFO - Run 'cfgset search -s cray_dws --level advanced p0' to review the current settings.
```

4. Validate the config set.

```
smw# cfgset validate p0
...
INFO - ConfigSet 'p0' is valid.
```

Correct any discrepancies before proceeding.

5. Log in to the `sdb` node and activate the config set changes.

```
sdb# /etc/init.d/cray-ansible start
sdb# systemctl reload dwsd
```

5.3 Back Up and Restore DataWarp State Data

The DataWarp scheduler daemon, `dwsd`, relies on specific node and pool information for correct operation. This information is stored in a state database and should be backed up periodically to minimize any potential impact of events that may cause loss of this information (e.g., drive failure or backwards incompatible DWS upgrade). Cray recommends creating a DataWarp backup `cron` job or including it as part of a periodic maintenance checklist. Additionally, it is important to create a backup of the DataWarp configuration data at the following times:

- after initial installation and configuration of DataWarp
- after configuration changes
- prior to system upgrades

Note that the `dws` module must be loaded to use the backup and restore commands.

```
sdb# module load dws
```

Back Up

Two commands are available to back up the configuration data from the `dwsd` database, `dwcli config backup` and `dwbackup`. Any DataWarp administrator (e.g., `root`, `crayadm`) can run `dwcli config backup` when the `dwrest` service is running. This is slightly less restrictive and may be preferable to `dwbackup`, which must be run by `root` from the node on which `dwsd` runs (typically `sdb`).

The `dwbackup` command reads data from `sdb:/var/opt/cray/dws/dwsd.db`, whereas the `dwcli config backup` command reads data through the RESTful API (i.e., through `dwrest`). Both commands output the node and pool properties to `stdout`, and neither command backs up any user data.

The `dwbackup` command is normally used after an upgrade if the state database configuration information was not backed up using either command prior to the upgrade. This is because the upgrade may include a backwards incompatible change to the database. In this case, DataWarp fails to come up properly and `dwrest` fails to run, which prevents the use of `dwcli config backup`.

Example 1: Run `dwbackup` with default option settings

As `root` on to the `sdb` node:

```
sdb# dwbackup
{
  "nodes": [
    {
      "links": {
        "pool": "example"
      },
      "drain": false,
      "id": "example-node"
    }
  ],
  "pools": [
    {
      "id": "example",
      "units": "bytes",
      "granularity": 16777216
    }
  ]
}
```

Example 2: Use `dwbackup` to create a backup file

As `root` on the `sdb` node:

```
sdb# dwbackup > /persistent/storage/my_dws_backup.json
```

Example 3: Use `dwcli config backup` to create a backup file

As a DataWarp administrator on a login node:

```
crayadm@login> dwcli config backup > /persistent/storage/my_dws_backup.json
```

Restore

Any DataWarp administrator can run `dwcli config restore` to restore a DataWarp configuration as captured in a backup created by either backup command.

Example 4: Restore a saved configuration

```
crayadm@sdb> dwcli config restore < /persistent/storage/my_dws_backup.json
```

For further information, see the `dwcli(8)` and `dwbackup(8)` man pages.

5.4 Drain a Storage Node

About this task

After an administrator assigns a node to a pool, any capacity on the node may be used when satisfying instance requests. There are times when a site does not want to allow new instances to be placed on a node and also does not want to disassociate the node from a pool. The `drain` attribute on a node controls this behavior. If a node is in a drain state, the DataWarp service (DWS) will not place new instances on the node and will also remove that node's free capacity contribution to a pool. The `dwstat nodes pools` command displays this information.

Procedure

1. Check the node and pool information.

```
crayadm@sdb> module load dws
crayadm@sdb> dwstat nodes pools
     node    pool online drain  gran   capacity insts activs
nid00022 wlm_pool   true false  8MiB    3.64TiB     0      0
nid00023 wlm_pool   true false  8MiB    3.64TiB     0      0

    pool units quantity    free   gran
wlm_pool bytes  7.28TiB 7.28TiB 128GiB
```

2. Drain the storage node.

```
crayadm@sdb> dwcli update node --name hostname --drain
```

where *hostname* is the hostname of the node to be drained

For example:

```
crayadm@sdb> dwcli update node --name nid00022 --drain
crayadm@sdb> dwstat nodes pools
     node    pool online drain  gran   capacity insts activs
nid00022 wlm_pool   true  true  8MiB    3.64TiB     0      0
nid00023 wlm_pool   true false  8MiB    3.64TiB     0      0

    pool units quantity    free   gran
wlm_pool bytes  7.28TiB 3.64TiB 128GiB
```

3. (Optional) If shutting down a node after draining it, wait for existing instances to be removed from the node. The `dwstat nodes` command displays the number of instances present in the `inst` column; `0` indicates no instances are present. In a healthy system, instances are removed over time as batch jobs complete. If it takes longer than expected, or to clean up the node more quickly, identify the fragments (pieces of instances) on the node by consulting the `node` column output of the `dwstat fragments` command and then finding the corresponding instance by looking at the `inst` column output:

```
crayadm@sdb> dwstat fragments
frag state inst  capacity     node
 102  CA--   47 745.19GiB nid00022
```

4. (Optional) Remove that instance.

```
crayadm@sdb> dwcli rm instance --id 47
```

   Persistent DataWarp instances, which have a lifetime that may span multiple batch jobs, must also be removed, either through a WLM-specific command or with `dwcli`.

5. When the node is fit for being used by the DWS again, unset the drain, thereby allowing the DWS to place new instances on the node.

```
crayadm@sdb> dwstat nodes pools
     node    pool online drain  gran   capacity insts activs
nid00022 wlm_pool   true false  8MiB    3.64TiB     0      0
nid00023 wlm_pool   true false  8MiB    3.64TiB     0      0

    pool units quantity    free   gran
wlm_pool bytes  7.28TiB 7.28TiB 128GiB
```

5.5 Remove a Node From a Storage Pool

Prerequisites

- DataWarp administrator privileges

About this task

If a node no longer exists or is no longer a DataWarp server node, it should be removed from the pool to which it is assigned.

Procedure

1. Log in to a CLE service node as a DataWarp administrator and load the `dws` module.

```
crayadm@login> module load dws
```

2. Verify node names.

```
crayadm@login> dwstat pools nodes
    pool units quantity    free    gran
wlm_pool bytes   4.0TiB  4.0TiB  128GiB
```

```
    node     pool online drain  gran   capacity  insts  activs
nid00065 wlm_pool   true false 16MiB       1TiB      0       0
nid00066 wlm_pool   true false 16MiB       1TiB      0       0
nid00069 wlm_pool   true false 16MiB       1TiB      0       0
nid00070 wlm_pool   true false 16MiB       1TiB      0       0
nid00004        -   true false 16MiB    3.64TiB      0       0
nid00005        -   true false 16MiB    3.64TiB      0       0
```

3. Remove the desired node from its pool.

```
crayadm@login> dwcli update node --name hostname --rm-pool
```

For example:

```
crayadm@login> dwcli update node --name nid00066 --rm-pool
```

The node is no longer assigned to the pool: `wlm_pool`, decreasing the pool's storage capacity.

```
crayadm@login> dwstat pools nodes
    pool units quantity    free    gran
wlm_pool bytes   3.0TiB  3.0TiB  128GiB

    node     pool online drain  gran   capacity  insts  activs
nid00065 wlm_pool   true false 16MiB       1TiB      0       0
nid00069 wlm_pool   true false 16MiB       1TiB      0       0
nid00070 wlm_pool   true false 16MiB       1TiB      0       0
nid00004        -   true false 16MiB    3.64TiB      0       0
nid00005        -   true false 16MiB    3.64TiB      0       0
nid00066        -   true false 16MiB       1TiB      0       0
```

5.6 Change a Node's Pool

Prerequisites

- DataWarp administrator privileges

About this task

Changing a node's pool involves reassigning it to a different pool; there is no need to remove it from its original pool.

Procedure

1. Log in to a CLE service node as a DataWarp administrator and load the `dws` module.

```
crayadm@login> module load dws
```

2. Verify pool and node names.

```
crayadm@login> dwstat pools nodes
    pool units quantity    free    gran
pvt_pool bytes  3.64TiB 3.64TiB   16GiB
wlm_pool bytes   4.0TiB  4.0TiB  128GiB
```

```
    node     pool online drain  gran   capacity  insts  activs
nid00004 pvt_pool   true false 16MiB    3.64TiB      0       0
nid00065 wlm_pool   true false 16MiB       1TiB      0       0
nid00066 wlm_pool   true false 16MiB       1TiB      0       0
nid00069 wlm_pool   true false 16MiB       1TiB      0       0
nid00070 wlm_pool   true false 16MiB       1TiB      0       0
nid00005         -   true false 16MiB    3.64TiB      0       0
```

3. Reassign the desired node to the desired pool.

```
crayadm@login> dwcli update node --name hostname --pool pool_name
```

To reassign node `nid00066` to pool `pvt_pool`:

```
crayadm@login> dwcli update node --name nid00066 --pool pvt_pool
```

Node `nid00066` is assigned to the pool: `pvt_pool`; resulting in increased storage capacity for `pvt_pool` and decreased capacity for `wlm_pool`.

```
crayadm@login> dwstat pools nodes
    pool units quantity    free    gran
pvt_pool bytes  4.64TiB 4.64TiB   16GiB
wlm_pool bytes   3.0TiB  3.0TiB  128GiB

    node     pool online drain  gran   capacity  insts  activs
nid00004 pvt_pool   true false 16MiB    3.64TiB      0       0
nid00066 pvt_pool   true false 16MiB       1TiB      0       0
nid00065 wlm_pool   true false 16MiB       1TiB      0       0
nid00069 wlm_pool   true false 16MiB       1TiB      0       0
nid00070 wlm_pool   true false 16MiB       1TiB      0       0
nid00005         -   true false 16MiB    3.64TiB      0       0
```

5.7 Replace a Blown Fuse

After a workload manager sends DataWarp requests to the DataWarp service (DWS), the DWS begins preparing the SSDs and compute nodes for the corresponding batch job. When things are working well, this process is quick and does not require admin intervention. The `dwstat` command reports `CA---` or `CA--` in the `state` column for all objects associated with the batch job.

If the DWS encounters difficulty creating or destroying an object, it retries a configurable number of times but eventually stops trying. To convey that the retry threshold has been exceeded, the DWS borrows terminology from another domain and reports that the object's *fuse* is blown. The `dwstat` command reports this as an `F` in the 3rd hyphen position of the `state` column. For example, `C-F--` as in the following `dwstat activations` output:

```
crayadm@sdb> module load dws
crayadm@sdb> dwstat activations
activ state sess conf nodes
    2 C-F--    5   11     1
```

When `dwstat` reports that an object's fuse is blown, it likely indicates a serious error that needs investigating. Clues as to what broke and why may be found in the Lightweight Log Manager (LLM) log file for `dwmd`, found at `smw:/var/opt/cray/log/p#-current/dws/dwmd-`*`date`*. In this log file, each session/instance/configuration/registration/activation is abbreviated as `sid`/`iid`/`cid`/`rid`/`aid`; therefore, information for the

resource with the blown fuse is searchable by ID. For example, if a fuse is blown on configuration 16, `grep` the log for `cid:16`:

```
crayadm@smw> cd /var/opt/cray/log/p3-current/dws
crayadm@smw> grep -A 4 cid:16 dwmd-20160520
2016-05-20 12:39:41 (11964): <76> [os-172-30-196-191]: (cid:16,sid:32,stoken:987333) dws_realm_member
ERROR:Invalid host found nid12345 in [u'nid12345']
2016-05-20 12:39:41 (11964): <76> [os-172-30-196-191]: (cid:16,sid:32,stoken:987333) dws_realm_member
ERROR:realm_member_create2 2 failed: gen_host_list_file failed for dwfs2_id=2 realm_id=2
2016-05-20 12:39:41 (11964): <76> [os-172-30-196-191]: Traceback (most recent call last):
2016-05-20 12:39:41 (11964): <76> [os-172-30-196-191]:   File "/opt/cray/dws/1.3-1.0000.67025.34.35.tcp/
lib/dws_realm_member.py", line 223, in realm_member_create2
2016-05-20 12:39:41 (11964): <76> [os-172-30-196-191]:     % (dwfs_id, realm_id))
2016-05-20 12:39:41 (11964): <76> [os-172-30-196-191]: RuntimeError: gen_host_list_file failed for
dwfs2_id=2 realm_id=2
```

Alternatively, if the batch job experiencing the failure is known, `grep` the same `dwmd` log file for the batch job ID. For example:

```
crayadm@smw> cd /var/opt/cray/log/p3-current/dws
crayadm@smw> grep -A 4 stoken:987333 dwmd-20160520
2016-05-20 12:39:41 (11964): <76> [os-172-30-196-191]: (cid:16,sid:32,stoken:987333) dws_realm_member
ERROR:Invalid host found nid12345 in [u'nid12345']
2016-05-20 12:39:41 (11964): <76> [os-172-30-196-191]: (cid:16,sid:32,stoken:987333) dws_realm_member
ERROR:realm_member_create2 2 failed: gen_host_list_file failed for dwfs2_id=2 realm_id=2
2016-05-20 12:39:41 (11964): <76> [os-172-30-196-191]: Traceback (most recent call last):
2016-05-20 12:39:41 (11964): <76> [os-172-30-196-191]:   File "/opt/cray/dws/1.3-1.0000.67025.34.35.tcp/
lib/dws_realm_member.py", line 223, in realm_member_create2
2016-05-20 12:39:41 (11964): <76> [os-172-30-196-191]:     % (dwfs_id, realm_id))
2016-05-20 12:39:41 (11964): <76> [os-172-30-196-191]: RuntimeError: gen_host_list_file failed for
dwfs2_id=2 realm_id=2
...
```

When the issue is understood and resolved, use the `dwcli` command to replace the blown fuse associated with the object, and thereby inform DWS to retry the operations associated with the failed object. For example, continuing with the above failed activation:

```
crayadm@sdb> dwcli update activation --id 2 --replace-fuse
```

Use `dwstat` to find the status of the object again. Fuses are replaceable as many times as necessary.

For further information, see the `dwstat(1)` and `dwcli(8)` man pages.

5.8 Enable the Node Health Checker DataWarp Plugin (if Necessary)

Prerequisites

- Ability to log in as `root`

About this task

The Node Health Checker (NHC) DataWarp plugin is enabled by default at system installation but may become disabled. This procedure describes how to verify that the DataWarp plugin is enabled and, if not, walks through the steps to enable it.

When enabled, NHC is automatically invoked upon the termination of every application. It performs specified tests to determine if compute nodes allocated to the application are healthy enough to support running subsequent applications. The `DataWarp` plugin is a script to check that any reservation-affiliated DataWarp mount points have

been removed; it can only detect a problem once the last reservation on a node completes. The behavior of NHC after a job has terminated is determined through settings in the configurator.

The configurator guides the user through the configuration process with explanations, options, and prompts. The majority of this dialog is not displayed in the steps below, only prompts and example responses are displayed.

> **TIP:** The configurator uses index numbering to identify configuration items. This numbering may vary; the value used in the examples may not be correct for all systems. The user must search the listing displayed on the screen to determine the correct index number for the service/setting being configured.

For further information about NHC, see the `intro_NHC(8)` man page and XC™ Series System Administration Guide.

Procedure

1. Determine if `Plugin DataWarp` is enabled in the `cray_node_health` service of the CLE config set.

```
smw# cfgset search -s cray_node_health -l advanced -t DataWarp p0_example
```

   a. Exit this procedure if the DataWarp plugin is enabled.

```
# 11 matches for 'DataWarp' from cray_node_health_config.yaml
#-------------------------------------------------------------------------------
cray_node_health.settings.plugins.data.Plugin DataWarp.name: Plugin
cray_node_health.settings.plugins.data.Plugin DataWarp.enabled: True
cray_node_health.settings.plugins.data.Plugin DataWarp.command: datawarp.sh -v
cray_node_health.settings.plugins.data.Plugin DataWarp.action: Admindown
cray_node_health.settings.plugins.data.Plugin DataWarp.warntime: 30
cray_node_health.settings.plugins.data.Plugin DataWarp.timeout: 360
cray_node_health.settings.plugins.data.Plugin DataWarp.restartdelay: 65
cray_node_health.settings.plugins.data.Plugin DataWarp.uid: 0
cray_node_health.settings.plugins.data.Plugin DataWarp.gid: 0
cray_node_health.settings.plugins.data.Plugin DataWarp.sets: Reservation
cray_node_health.settings.plugins.data.Plugin DataWarp.command: datawarp.sh -v
```

   b. Continue with this procedure if the DataWarp plugin is not enabled.

```
# No matches found in the configuration data for the given search terms.

INFO - Matches may be hidden by level/state/service filtering.
INFO - See 'cfgset search -h' for filtering options.
```

2. Invoke the configurator for the `cray_node_health` service.

```
smw# cfgset update -m interactive -l advanced -s cray_node_health p0_example
Service Configuration Menu (Config Set: p0_example, type: cle)
   cray_node_health        [ status: enabled ]  [ validation: valid ]

--------------------------------------------------------------------------------
  Selected      #      Settings                                    Value/Status
                                                                 (level=advanced)
--------------------------------------------------------------------------------
...
               27)     memory_plugins
                          desc: Default Memory                     [ OK ]

               28)     plugins
                          desc: Default Alps                       [ OK ]
                          desc: Plugin DVS Requests                [ OK ]
```

```
                            desc: Default Application             [ OK ]
                            desc: Default Reservation             [ OK ]
                            desc: Plugin Nvidia                   [ OK ]
                            desc: Plugin ugni                     [ OK ]
                            desc: Xeon Phi Plugin App Test        [ OK ]
                            desc: Xeon Phi Plugin Reservation     [ OK ]
                            desc: Xeon Phi Plugin Memory          [ OK ]
                            desc: Xeon Phi Plugin Alps            [ OK ]
                            desc: Plugin Sigcont                  [ OK ]
                            desc: Plugin Hugepage Check           [ OK ]
--------------------------------------------------------------------------------
...

Node Health Service Menu [default: save & exit - Q] $
```

3. Select the **plugins** setting using the index numbering.

   TIP: The configurator uses index numbering to identify configuration items. This numbering may vary; the value used in the examples may not be correct for all systems. The user must search the listing displayed on the screen to determine the correct index number for the service/setting being configured.

```
Node Health Service Menu [default: save & exit - Q] $ 28
```

4. Configure **plugins**.

```
Node Health Service Menu [default: configure - C] $ C
```

5. Add an entry.

   a. Enter **+**.

```
cray_node_health.settings.plugins
[<cr>=set 12 entries, +=add an entry, ?=help, @=less] $ +
```

   b. Set **desc** to `Plugin DataWarp`.

```
cray_node_health.settings.plugins.data.desc
[<cr>=set '', <new value>, ?=help, @=less] $ Plugin DataWarp
```

   c. Set **name** to `Plugin`.

```
cray_node_health.settings.plugins.data.Plugin DataWarp.name
[<cr>=set 'Plugin', <new value>, ?=help, @=less] $ <cr>
```

   d. Set **enabled** to `true`.

```
cray_node_health.settings.plugins.data.Plugin DataWarp.enabled
[<cr>=set 'false', <new value>, ?=help, @=less] $ true
```

   e. Set **command** to `datawarp.sh -v`.

```
cray_node_health.settings.plugins.data.Plugin DataWarp.command
[<cr>=set '', <new value>, ?=help, @=less] $ datawarp.sh -v
```

   f. Set **action** to `Admindown`.

```
cray_node_health.settings.plugins.data.Plugin DataWarp.command
[<cr>=set '', <new value>, ?=help, @=less] $ Admindown
```

g. Set **warntime** to `30`.

```
cray_node_health.settings.plugins.data.Plugin DataWarp.warntime
[<cr>=set '0', <new value>, ?=help, @=less] $ 30
```

h. Set **timeout** to `360`.

```
cray_node_health.settings.plugins.data.Plugin DataWarp.timeout
[<cr>=set '0', <new value>, ?=help, @=less] $ 360
```

i. Set **restartdelay** to `65`.

```
cray_node_health.settings.plugins.data.Plugin DataWarp.restartdelay
[<cr>=set '0', <new value>, ?=help, @=less] $ 65
```

j. Set **uid** to `0`.

```
cray_node_health.settings.plugins.data.Plugin DataWarp.uid
[<cr>=set '0', <new value>, ?=help, @=less] $ 0
```

k. Set **gid** to `0`.

```
cray_node_health.settings.plugins.data.Plugin DataWarp.gid
[<cr>=set '0', <new value>, ?=help, @=less] $ 0
```

l. Set **sets** to `Reservation`.

```
cray_node_health.settings.plugins.data.Plugin DataWarp.sets
[<cr>=set 'Application', <new value>, ?=help, @=less] $ Reservation
```

m. Accept the settings.

```
cray_node_health.settings.plugins
[<cr>=set 13 entries, +=add an entry, ?=help, @=less] $ <cr>
...

               28)     plugins
                          desc: Default Alps                    [ OK ]
                          desc: Plugin DVS Requests             [ OK ]
                          desc: Default Application             [ OK ]
                          desc: Default Reservation             [ OK ]
                          desc: Plugin Nvidia                   [ OK ]
                          desc: Plugin ugni                     [ OK ]
                          desc: Xeon Phi Plugin App Test        [ OK ]
                          desc: Xeon Phi Plugin Reservation     [ OK ]
                          desc: Xeon Phi Plugin Memory          [ OK ]
                          desc: Xeon Phi Plugin Alps            [ OK ]
                          desc: Plugin Sigcont                  [ OK ]
                          desc: Plugin Hugepage Check           [ OK ]
                          desc: Plugin DataWarp                 [ OK ]
```

n. Save the changes and exit the configurator.

```
Node Health Service Menu [default: save & exit - Q] $ Q
INFO - Configuration worksheets will be saved to
    - /var/opt/cray/imps/config/sets/p0_example/worksheets
INFO - Changelog will be written to
    - /var/opt/cray/imps/config/sets/p0_example/changelog/
changelog_2016-04-08T17:19:18.yaml
INFO - Running post-configuration scripts
```

```
INFO - Locally cloning ConfigSet 'p0_example' to 'p0_example-
autosave-2016-04-08T17:19:29'.
INFO - Successfully cloned to ConfigSet 'p0_example-
autosave-2016-04-08T17:19:29'.
INFO - Removed ConfigSet 'p0_example-autosave-2016-04-06T17:09:08'.
INFO - ConfigSet 'p0_example' has been updated.
INFO - Run 'cfgset search -s cray_node_health --level advanced p0_example'
to review the current settings.
```

6. Verify the settings.

```
smw# cfgset search -s cray_node_health -l advanced -t DataWarp p0_example
...
# 11 matches for 'DataWarp' from cray_node_health_config.yaml
#--------------------------------------------------------------------------
-
cray_node_health.settings.plugins.data.Plugin DataWarp.name: Plugin
cray_node_health.settings.plugins.data.Plugin DataWarp.enabled: True
cray_node_health.settings.plugins.data.Plugin DataWarp.command: datawarp.sh -v
cray_node_health.settings.plugins.data.Plugin DataWarp.action: Admindown
cray_node_health.settings.plugins.data.Plugin DataWarp.warntime: 30
cray_node_health.settings.plugins.data.Plugin DataWarp.timeout: 360
cray_node_health.settings.plugins.data.Plugin DataWarp.restartdelay: 65
cray_node_health.settings.plugins.data.Plugin DataWarp.uid: 0
cray_node_health.settings.plugins.data.Plugin DataWarp.gid: 0
cray_node_health.settings.plugins.data.Plugin DataWarp.sets: Reservation
cray_node_health.settings.plugins.data.Plugin DataWarp.command: datawarp.sh -v
smw#
```

   Correct any discrepancies before proceeding.

7. Reboot the system.

5.9 Deconfigure DataWarp

Prerequisites

- `root` privileges
- The system is not running

About this task

Follow this procedure to remove the DataWarp configuration from a system.

Procedure

1. Invoke the configurator in interactive mode to update the CLE config set.

```
smw# cfgset update -m interactive -s cray_dws p0_example
```

2. Disable the service by entering **E**.

```
Cray dws Menu [default: save & exit - Q] $ E
```

3. Save and exit the configurator.

```
Cray dws Menu [default: save & exit - Q] $ Q
```

4. Reboot the system.

5. Log in to an SSD-endowed node as `root`.

   This example uses `nid00349`.

6. Remove the data.

   a. Remove the Logical Volume Manager (LVM) volume group.

   ```
   nid00349# vgremove dwcache
   ```

   A confirmation prompt may appear:

   ```
   Do you really want to remove volume group "dwcache" containing 1 logical
   volumes? [y/n]:
   ```

   b. Answer `yes`.

   c. Identify the SSD block devices.

   ```
   nid00349# pvs
     PV           VG      Fmt  Attr PSize PFree
     /dev/nvme0n1 dwcache lvm2 a--  1.46t 1.46t
     /dev/nvme1n1 dwcache lvm2 a--  1.46t 1.46t
     /dev/nvme2n1 dwcache lvm2 a--  1.46t 1.46t
     /dev/nvme3n1 dwcache lvm2 a--  1.46t 1.46t
   ```

   d. Remove LVM ownership of devices. Specify all SSD block devices on the node.

   ```
   nid00349:# pvremove /dev/nvme0n1 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1
   Labels on physical volume "/dev/nvme0n1" successfully wiped
   Labels on physical volume "/dev/nvme1n1" successfully wiped
   Labels on physical volume "/dev/nvme2n1" successfully wiped
   Labels on physical volume "/dev/nvme3n1" successfully wiped
   ```

7. Repeat steps *5* on page 59 through *6* on page 59 for all SSD nodes listed in `managed_nodes`.

DataWarp is deconfigured.

5.10 Prepare to Replace a DataWarp SSD

Prerequisites

- DataWarp administrator privileges
- Knowledge of the configuration of the blade on which the failing SSD is located

About this task

SSDs may require replacement due to hardware failure or low remaining endurance. Before replacing an SSD, the DataWarp service (DWS) is instructed to temporarily stop using it for future usages, and any existing usages are cleaned up. After the SSD is replaced, it is initialized and DWS is informed that the new hardware is available for use.

> **IMPORTANT:** SSD replacement involves powering down a blade and physically removing it from a cabinet. Because a blade consists of more than one node, SSD replacement likely impacts more than just the SSD-endowed node. If the other nodes on the blade are used by DataWarp, which is the typical configuration, then DataWarp is told to stop using them as well. If the other nodes on the blade are not used by DataWarp, they are shut down gracefully in accordance with their respective software.

This procedure covers three node types:

1. Failing DWS-managed SSD nodes
2. Healthy DWS-managed nodes on the same blade as a failing SSD
3. Nodes not managed by DWS on the same blade as a failing SSD

Procedure

1. Log in to the `sdb` node and load the `dws` module.

   ```
   crayadm@sdb> module load dws
   ```

2. Drain the failing SSD node. This prevents the creation of new instances on the node and also removes the node's free capacity contribution from the pool.

   ```
   crayadm@sdb> dwcli update node --name hostname --drain
   ```

   Throughout these examples, `nid00350` is the failing SSD node and `nid00349` is located on the same blade.

   ```
   crayadm@sdb> dwcli update node --name nid00350 --drain
   ```

3. **WAIT** for all existing instances to be removed from the node.

   ```
   crayadm@sdb> watch -n 10 dwstat nodes
   ```

   a. If no instances or activations remain, proceed to step *4* on page 61.

   ```
   crayadm@sdb> watch -n 10 dwstat nodes
   Every 10.0s: dwstat nodes

       node     pool online drain  gran capacity insts activs
   nid00322 wlm_pool   true false 16MiB  7.28TiB     0      0
   nid00349 wlm_pool   true false 16MiB  7.28TiB     0      0
   nid00350 wlm_pool   true  true 16MiB  7.28TiB     0      0
   ```

   b. Determine the IDs of any persistent instances and remove them.

   This example shows that the site needs to wait or take action for one instance on `nid00350`.

   ```
   crayadm@sdb> watch -n 10 dwstat nodes
   Every 10.0s: dwstat nodes
   ```

```
    node     pool online drain  gran capacity insts activs
nid00322 wlm_pool   true false 16MiB  7.28TiB     0      0
nid00349 wlm_pool   true false 16MiB  7.28TiB     0      0
nid00350 wlm_pool   true  true 16MiB  7.28TiB     1      0

crayadm@sdb> dwstat fragments | grep nid00350
 frag state inst  capacity     node
88071  CA-- 2227 596.16GiB nid00350
88072  CA-- 2227 596.16GiB nid00350
88073  CA-- 2227 596.16GiB nid00350
88074  CA-- 2227 596.16GiB nid00350
crayadm@sdb> dwcli rm instance --id 2227
```

WAIT for the instance to be removed. An instance that cannot be removed is likely blocked by a reservation trying to copy data back out to the parallel file system (PFS). In which case, the reservation may need to be set to `--no-wait`. For further information, see *Registrations* on page 24.

4. Log in to the failing SSD node as root.

   This example uses `nid00350`.

5. Display and remove the logical volume(s).

   TIP: Use `-f` to force removal.

```
nid00350# lvdisplay
  --- Logical volume ---
  LV Path                /dev/dwcache/s98i94f104o0
  LV Name                s98i94f104o0
  VG Name                dwcache
  LV UUID                910tio-RJXq-puYV-s3UL-yDM1-RoQl-HugeTM
  LV Write Access        read/write
  LV Creation host, time nid00350, 2016-02-22 13:29:11 -0500
  LV Status              available
  # open                 0
  LV Size                3.64 TiB
  Current LE             953864
  Segments               2
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     1024
  Block device           253:0

nid00350# lvremove /dev/dwcache
```

6. Display and remove the volume group(s).

```
nid00350# vgs
  VG       #PV #LV #SN Attr   VSize VFree
  dwcache    2   0   0 wz--n- 3.64t 3.64t
nid00350# vgremove dwcache
  Volume group "dwcache" successfully removed
```

7. Display and remove the physical volume(s).

```
nid00350# pvs
PV           VG        Fmt Attr PSize PFree
/dev/nvme0n1           lvm2 a--  1.46t 1.26t
/dev/nvme1n1           lvm2 a--  1.46t 1.26t
```

```
/dev/nvme2n1          lvm2 a--  1.46t 1.26t
/dev/nvme3n1          lvm2 a--  1.46t 1.26t

nid00350# pvremove /dev/nvme0n1 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1
  Labels on physical volume "/dev/nvme1n1" successfully wiped
  Labels on physical volume "/dev/nvme0n1" successfully wiped
  Labels on physical volume "/dev/nvme1n1" successfully wiped
  Labels on physical volume "/dev/nvme1n1" successfully wiped
```

The failing SSD is disabled and ready for replacement; however, the other node(s) on the blade must first be quiesced. Repeat the previous steps if there are multiple failing nodes.

8. Quiesce the non-failing nodes that share the blade with the failing SSD. Proceed based on node type:
   - DWS-managed SSD nodes: drain the nodes and remove all instances. **Do not** remove logical volumes, volume groups, physical volumes or labels.

     ```
     dwcli update node --name nid00349 --drain
     ```
   - Nodes not managed by DWS: refer to the software-specific documentation
9. Follow appropriate hardware procedures to power down the blade and replace the SSD node.

   Further software configuration is required after the SSD node is physically replaced, see *Complete the Replacement of an SSD Node* on page 62.

5.11 Complete the Replacement of an SSD Node

Prerequisites

- DataWarp administrator privileges
- Completion of *Prepare to Replace a DataWarp SSD* on page 59

About this task

SSDs may require replacement due to hardware failure or low remaining endurance. After replacing an SSD, a one-time manual device configuration that defines the Logical Volume Manager (LVM) structure is done, and then the DataWarp service (DWS) is informed that the new hardware is available for use.

> **IMPORTANT:** SSD replacement involves power cycling the blade on which the new SSD is located. Because a blade consists of more than one node, SSD replacement likely impacts more than just the SSD-endowed node. If the other nodes on the blade are used by DataWarp, which is the typical configuration, then DataWarp is told to enable them as well. If the other nodes on the blade are not used by DataWarp, they are enabled in accordance with their respective software.

This procedure covers three node types:

1. Newly-replaced SSD nodes for DataWarp
2. DWS-managed nodes on the same blade as a newly-replaced node
3. Nodes not managed by DWS on the same blade as a newly-replaced node

Procedure

1. Power up the blade and boot the nodes according to standard procedure.
2. Over-provision the new SSD if it is an Intel P3608; see *Over-provision an Intel P3608 SSD* on page 27.
3. Verify that the new SSD has the proper PCIe generation and width:
   - Intel P3608:
     - On-board PLX switch trains as Gen3 and x8 width
     - Each card has two x4 SSD devices connected by the PLX switch
   - Samsung SM1725 trains as Gen3 and x8 width
   - SX300 (ioMemory3) trains as Gen2 and x4 width
   - Fusion ioScale2 cards are not supported with CLE 6.0/SMW 8.0 and beyond

```
smw# xtcheckhss --nocolor --detail=f --pci
      Node Slot                   Name Target Gen Trained Gen Target Width Trained Width
---------- ---- ---------------------- ---------- ----------- ------------ -------------
...

c0-0c0s3n1    0 Intel_P3600_Series_SSD       Gen3        Gen3           x4            x4
c0-0c0s3n1    0 Intel_P3600_Series_SSD       Gen3        Gen3           x4            x4
c0-0c0s3n1    0             PLX_switch       Gen3        Gen3           x8            x8
c0-0c0s3n1    1 Intel_P3600_Series_SSD       Gen3        Gen3           x4            x4
c0-0c0s3n1    1 Intel_P3600_Series_SSD       Gen3        Gen3           x4            x4
c0-0c0s3n1    1             PLX_switch       Gen3        Gen3           x8            x8
...
```

4. Initialize the new SSD to define the LVM structure, see *Initialize an SSD* on page 31.
5. Set `--no-drain` for the new SSD node.

```
crayadm@sdb> dwcli update node --name hostname --no-drain
```

   In this example, `nid00350` is the new SSD node and `nid00349` is a DWS-managed node located on the same blade.

```
crayadm@sdb> dwcli update node --name nid00350 --no-drain
crayadm@sdb> dwstat nodes
    node     pool online drain  gran capacity insts activs
nid00322 wlm_pool   true false 16MiB  7.28TiB     0      0
nid00349 wlm_pool   true  true 16MiB  7.28TiB     0      0
nid00350 wlm_pool   true false 16MiB  7.28TiB     0      0
```

   The new SSD is enabled and its storage is added to the pool; however, the other nodes on the blade must also be enabled. Repeat the previous steps if there are multiple new DataWarp nodes.

6. Enable the nodes that share the blade with the new SSD. Proceed based on node type:
   - DWS-managed SSD nodes: set the nodes to `un-drain`.

     ```
     dwcli update node --name nid00349 --no-drain
     ```

   - Nodes not managed by DWS: refer to the software-specific documentation.

   This completes the process of replacing a DataWarp SSD node.

6 Troubleshooting

6.1 Where are the Log Files?

The DataWarp scheduler daemon (`dwsd`), manager daemon (`dwmd`), and RESTful service (`dwrest`) write to local log files as well as to log files managed by the Lightweight Log Manager (LLM). `logrotate` manages the following local log files on the specified nodes:

- API gateway:`/var/opt/cray/dws/log/dwmd.log`, `gunicorn.log`
- API gateway:`/var/log/nginx/access.log`, `error.log`
- Scheduler node:`/var/opt/cray/dws/log/dwsd.log`
- Managed nodes:`/var/opt/cray/dws/log/dwmd.log`

By default, `logrotate` runs as a daily cron job. For further information, see the `logrotate(8)` and `cron(8)` man pages.

The following LLM-managed log files are located on the SMW at `/var/opt/cray/log/p#-bootsession/dws`:

- `dwmd-`*`date`*: multiple daemons (one for every managed node) log to this file
- `dwsd-`*`date`*: one daemon logs to this file
- `dwrest-`*`date`*: multiple daemons (one for every API gateway) can log to this file

For further information about LLM, see the `intro_LLM(8)` man page.

6.2 Old Nodes in dwstat Output

The DataWarp Service (DWS) learns about node existence from two sources:

1. Heartbeat registrations between the `dwsd` process and the `dwmd` processes
2. Hostnames provided by workload managers as users are granted access to compute nodes as part of their batch jobs

The `dwsd` process on the sdb node stores the DWS state in its state file and controls the information displayed by `dwstat nodes`. On `dwsd` process restart, `dwsd` removes a node from its state file if the node meets the following criteria:

1. the node is not in a pool
2. there are no instances on the node

3. there are no activations on the node
4. the node does not belong to a session

If a node lingers in the `dwstat nodes` output longer than expected, verify the above criterion are met, and then restart the `dwsd` process on the sdb node: `service dwsd restart`.

6.3 Dispatch Requests

The DataWarp scheduler daemon, `dwsd`, is designed to dispatch requests to the `dwmd` processes as soon as there is work for the `dwmd` processes to perform. If the `dwsd` gets confused or has a bug, it may fail to dispatch a request at the appropriate time. If this is suspected, send `SIGUSR1` to the `dwsd` process on the sdb node, forcing it to look for tasks to perform.

```
sdb# kill -USR1 $(</var/opt/cray/dws/dwsd.pid)
sdb# tail -6 /var/opt/cray/dws/log/dwsd.log
2015-09-17 15:24:05 ========== Event on fd 4
2015-09-17 15:24:05 Caught signal User defined signal 1
2015-09-17 15:24:05 Alerting the task manager to wake up
2015-09-17 15:24:05 ========== Event on fd 7
2015-09-17 15:24:05 Finding tasks to spawn
2015-09-17 15:24:05 Nothing can be done right now
```

More likely than not, the `dwsd` cannot yet perform the action in question. Check if any nodes are not online (`dwstat nodes`) and if all prerequisites to the action are met. For example, the `dwsd` will not dispatch a request to create a configuration until after the corresponding instance has been created.

6.4 Recover After a Backwards-incompatible Upgrade

Prerequisites

- DataWarp administrator privileges (e.g., `root`, `crayadm`)

About this task

The DataWarp scheduler daemon, `dwsd`, relies on specific node and pool information, stored in state files, for correct operation. Occasionally, a DataWarp software upgrade may modify these state files such that the DataWarp service (DWS) is not backwards compatible with any state created by a previous DataWarp release. If this occurs, DataWarp does not come up correctly, and:

- The `dwcli` and `dwstat` commands fail and report a connection error to the `dwsd` daemon
- Messages similar to the following appear in both:
  - `sdb:/var/opt/cray/dws/log/dwsd.log`
  - `smw:/var/opt/cray/log/p#-`*`timestamp`*`/dws/dwsd-`*`timestamp`*

```
2015-11-13 15:51:07 State file is at v0.0
2015-11-13 15:51:07 ADMIN ALERT -> This version of dwsd expects state file v1.0, you have v0.0.
2015-11-13 15:51:07 ADMIN ALERT -> This version of dwsd is incompatible with the existing state
```

```
   file located at /var/opt/cray/dws/dwsd.db.  If you roll back to an
   older version of the DWS as well as underlying dependencies like DVS and kdwfs,
   you may be able to retrieve any existing data stored in your DataWarp instances.
   Otherwise, to get DWS working again, you can back up some DWS state (like pools)
   with the dwbackup tool and then later restore that state with the dwcli config restore
   tool.  See the DataWarp man pages and other documentation for further details.
2015-11-13 15:51:07 src/dwsd/context.c:dwsd_context_init():356 -> Unable to initialize sqlite
2015-11-13 15:51:07 Daemon ran for 2 seconds
2015-11-13 15:51:07 src/dwsd/dwsd.c:main():66 -> Context initialization failed.
2015-11-13 15:51:07 Shutting down
```

This procedure describes the steps to back up some DataWarp state (if not done prior to the software upgrade), remove the incompatible `dwsd.db` file, and restore the backed up state to an upgraded state file.

Procedure

1. Log in to the `sdb` node as `root`, and back up the DataWarp state if it was not backed up just prior to the software upgrade.

   ```
   sdb# module load dws
   sdb# dwbackup > /persistent/storage/my_dws_backup.json
   ```

2. Stop the `dwsd` service.

   ```
   sdb# service dwsd stop
   ```

3. Move the existing DataWarp state file.

   ```
   sdb# mv /var/opt/cray/dws/dwsd.db /var/opt/cray/dws/dwsd.db.old-$(date "+%Y%m%d%H%M%S")
   ```

4. Start the `dwsd` service.

   ```
   sdb# service dwsd start
   ```

5. Wait 600 seconds, or restart `dwmd` on all SSD-endowed nodes.

   ```
   sdb# module load pdsh
   sdb# pdsh -w dwnode1,dwnode2,... 'kill -USR1 $(</var/opt/cray/dws/dwmd.pid)'
   ```

   Where *dwnode#* is the hostname of any DataWarp server node.

6. Restore the backed up state to the upgraded state file.

   ```
   sdb# dwcli config restore </persistent/storage/my-dw-backup.json
   ```

6.5 Stage In or Out Fails When Transferring a Large Number of Files

The stage in and stage out operations associated with DataWarp scratch configurations have default timeout settings that are configured in both the NGINX webserver and the `dwrest` application runner, Gunicorn. If the staging of a large number of files fails, the default settings may need increasing. The following symptoms may indicate that timeouts have been hit:

- Stage in of a directory generates error 404 and job becomes `JobHeldAdmin`
- HTTP 500 errors appear in the NGINX logs (API gateway: `/var/log/nginx/access.log`, `error.log`)
- WLM user receives a stage in or out failure after `dwrest` is forcibly restarted

The default timeout settings (in seconds) are as follows:

- NGINX `proxy_read_timeout`: 645
- dwrestgun `timeout`: 600

The disparity between the timeouts is due to a 30 second grace period with a 15 second buffer when data is actively transferred back to clients.

TIP: Maintain at least a 30 second buffer for the NGINX timeout when modifying these default values.

These default values routinely allow for 3,000 to 10,000 files to stage in or out each minute with the transfer of 100,000 files typically not triggering the timeout. This is a general heuristic, and Cray recommends adjusting these timeouts based on both the worst-case anticipated load and stage in/out performance measurements from the system.

Modify the Default Settings

The following is an overview of the procedure to modify the NGINX and `dwrest` timeout settings.

1. Create a site-local Ansible play in `smw:/var/opt/cray/imps/config/sets/`*`configset`*`/ansible` to modify the default timeout setting for NGINX, a third-party package not directly affected by the Cray system configurator. Set *`numsecs`* as desired.

```
# Datawarp site local play.
- name: Datawarp site local
  hosts: localhost
  roles:
  vars:
    run_after:
    - early
    - dws

  tasks:
    - name: fixup dwrest.conf
      lineinfile: >
        dest=/etc/nginx/conf.d/dwrest.conf
        regexp="^#?\s*proxy_read_timeout="
        line="                proxy_read_timeout=numsecs"
      when: not ansible_local.cray_system.in_init and
            ansible_local.cray_system.hostid in cray_dws.settings.service.data.api_gateway_nodes

    - name: restart nginx
      service: name=nginx state=restarted
      when: not ansible_local.cray_system.in_init and
            ansible_local.cray_system.hostid in cray_dws.settings.service.data.api_gateway_nodes
```

2. Follow the procedure in *Modify DWS Advanced Settings* on page 41 to modify the default `dwrestgun` setting: `timeout` and to activate the changes and run the Ansible play.

   IMPORTANT: Use the format `timeout=`*`secs`* if defining it for the first time; that is, if it is a non-displayed setting (as explained in the procedure).

7 Diagnostics

7.1 SEC Notification when 90% of SSD Life Expectancy is Reached

When a DataWarp SSD reaches 90% of its life expectancy, a message is written to the console log file. If enabled, the Simple Event Correlator (SEC) monitors system log files for significant events such as this and sends a notification (either by email, IRC, writing to a file, or some user-configurable combination of all three) that this has happened. The notification for nearing the end of life of an SSD is as follows:

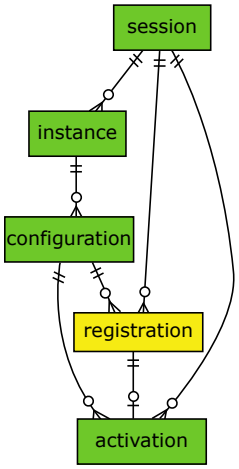
```
Mon 8/17/2015 3:17 PM
SEC: 15:17  sitename-systemname: Low SSD Life Remaining 8% c3-0c0s2n1 PCIe slot
-1
 Please contact your Cray support personnel or sales representative for SSD card
replacements.

 12 hours -- skip repeats period, applies on a per SSD basis.

 System:    sitename-systemname, sn9000
 Event:     Low ioMemory SSD Life Remaining (8%) c3-0c0s2n1 PCIe faceplate slot:
Unknown (only one slot is populated in this node)
 Time:      15:17:04 in logged string.
 Mon Aug 17 15:17:05 2015  --  Time when SEC observed the logged string.

 Entire line in log file:
 /var/opt/cray/log/p0-20150817t070336/console-20150817
 -----
 2015-08-17T15:17:04.871808-05:00 c3-0c0s2n1 PCIe slot#:-1,Name:ioMemory
SX300-3200,SN:1416G0636,Size:3200GB,Remaining life: 8%,Temperature:41(c)

 SEC rule file:
 --------------
 /opt/cray/sec/default/rules/aries/h_ssd_remaining_life.sr

 Note:
 -----
 The skip repeats period is a period during which any repeats of this event
type that occur will not be reported by SEC. It begins when the first  message
that triggered this email was observed by SEC.
```

For detailed information about configuring SEC, see the *Configure Cray SEC Software* publication.

8 Terminology

The following diagram shows the relationship between the majority of the DataWarp service terminology using Crow's foot notation. A session can have 0 or more instances, and an instance must belong to only one session. An instance can have 0 or more configurations, but a configuration must belong to only one instance. A registration belongs to only one configuration and only one session. Sessions and configurations can have 0 or more registrations. An activation must belong to only one configuration, registration and session. A configuration can have 0 or more activations. A registration is used by 0 or no activations. A session can have 0 or more activations.

Figure 11. DataWarp Component Relationships



Activation An object that represents making a DataWarp configuration available to one or more client nodes, e.g., creating a mount point.

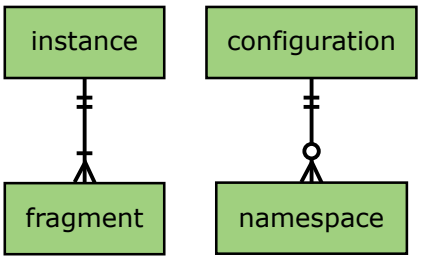
Client Node A compute node on which a configuration is activated; that is, where a DVS client mount point is created. Client nodes have direct network connectivity to all DataWarp server nodes. At least one parallel file system (PFS) is mounted on a client node.

Configuration A configuration represents a way to use the DataWarp space.

Fragment A piece of an instance as it exists on a DataWarp service node.

The following diagram uses Crow's foot notation to illustrate the relationship between an instance-fragment and a configuration-namespace. One instance has one or more fragments; a fragment can belong to only one instance. A configuration has 0 or more namespaces; a namespace can belong to only one configuration.

Figure 12. Instance/Fragment ↔ Configuration/Namespace Relationship



Instance — A specific subset of the storage space comprised of DataWarp fragments, where no two fragments exist on the same node. An instance is essentially raw space until there exists at least one DataWarp instance configuration that specifies how the space is to be used and accessed.

DataWarp Service — The DataWarp Service (DWS) manages access and configuration of DataWarp instances in response to requests from a workload manager (WLM) or a user.

Fragment — A piece of an instance as it exists on a DataWarp service node

Job Instance — A DataWarp instance whose lifetime matches that of a batch job and is only accessible to the batch job because the `public` attribute is not set.

Namespace — A piece of a scratch configuration; think of it as a folder on a file system.

Node — A DataWarp service node (with SSDs) or a compute node (without SSDs). Nodes with space are server nodes; nodes without space are client nodes.

Persistent Instance — A DataWarp instance whose lifetime matches that of possibly multiple batch jobs and may be accessed by multiple user simultaneously because the `public` attribute is set.

Pool — Groups server nodes together so that requests for capacity (instance requests) refer to a pool rather than a bunch of nodes. Each pool has an overall quantity (maximum configured space), a granularity of allocation, and a unit type. The units are either bytes or nodes (currently only bytes are supported). Nodes that host storage capacity belong to at most one pool.

Registration — A known usage of a configuration by a session.

Server Node — An IO service blade that contains two SSDs and has network connectivity to the PFS.

Session — An intagible object (i.e., not visible to the application, job, or user) used to track interactions with the DWS; typically maps to a batch job.

9 Prefixes for Binary and Decimal Multiples

Multiples of bytes						
SI decimal prefixes				IEC binary prefixes		
Name	**Symbol**	**Standard SI**	**Binary Usage**	**Name**	**Symbol**	**Value**
kilobyte	kB	10^3	2^{10}	kibibyte	KiB	2^{10}
megabyte	MB	10^6	2^{20}	mebibyte	MiB	2^{20}
gigabyte	GB	10^9	2^{30}	gibibyte	GiB	2^{30}
terabyte	TB	10^{12}	2^{40}	tebibyte	TiB	2^{40}
petabyte	PB	10^{15}	2^{50}	pebibyte	PiB	2^{50}
exabyte	EB	10^{18}	2^{60}	exbibyte	EiB	2^{60}
zettabyte	ZB	10^{21}	2^{70}	zebibyte	ZiB	2^{70}
yottabyte	YB	10^{24}	2^{80}	yobibyte	YiB	2^{80}

For a detailed explanation, including a historical perspective, see *http://physics.nist.gov/cuu/Units/binary.html*.