XC™ Series DataWarp™ Installation and Administration Guide

(CLE 6.0.UP05)

S-2564

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 (S-2564) supports Cray software release Cray Linux Environment (CLE) 6.0.UP05, released October 2017.

Changes to this document since CLE 6.0.UP04 include the following new and updated topics:

- Content previously contained in *Create a Storage Pool* is now found in *Storage Pool Configuration Guidelines* on page 74, and *Create a Storage Pool* on page 75 has been rewritten as a procedure to create a storage pool comprised of homogeneous SSD hardware.
- *Create a Storage Pool Comprised of Non-homogeneous SSD Hardware* on page 85 is a new topic aimed at non-typical sites such as the Cray Data Center.
- The dwsd.yaml `equalize_fragments` setting is now enabled by default. This affects how instances are allocated. Details can be found in *Storage Pool Configuration Guidelines* on page 74 and *Why Does the Free Capacity Displayed by dwstat pools not Match the Quantity Capacity?* on page 41.
- With `equalize_fragments` set by default, the `dwpoolhelp` command is no longer needed and is deprecated, see *The dwpoolhelp Command* on page 117.
- *Over-provision an Intel P3608 SSD* on page 65 - this is now an optional procedure based on a site's preference for better performance and higher endurance or more storage capacity for users.
- The `xtiossdflash(8)` command now supports SamSung SSDs, which is reflected by a title and procedural changes in *Flash NVMe SSD Firmware* on page 94.
- Memory swapping may not benefit all applications as described in *Memory Swapping Caveats* on page 127, which has been added to the Troubleshooting section as a reference for administrators.

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 DVS Administration Guide* (S-0005)
2. *XC™ Series System Administration Guide* (S-2393)
3. *XC™ Series DataWarp™ User Guide* (S-2558)
4. *XC™ Series Software Installation and Configuration Guide* (S-2559)
5. *XC™ Series Configurator User Guide* (S-2560)

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

Host name and account in command prompts

The host name in a command prompt indicates where the command must be run. The account that must run the command is also indicated in the prompt.

- The `root` or super-user account always has the `#` character at the end of the prompt.
- Any non-`root` account is indicated with *`account@hostname`*`>`. A user account that is neither `root` nor `crayadm` is referred to as `user`.

`smw#`	Run the command on the SMW as `root`.
`cmc#`	Run the command on the CMC as `root`.
`sdb#`	Run the command on the SDB node as `root`.
`crayadm@boot>`	Run the command on the boot node as the `crayadm` user.
`user@login>`	Run the command on any login node as any non-`root` user.

`hostname#`	Run the command on the specified system as `root`.
`user@hostname>`	Run the command on the specified system as any non-`root` user.
`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.

Command prompt inside chroot

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

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

Directory path in command prompt

Example prompts do not include the directory path, because long paths can reduce the clarity of examples. 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 appear in this publication:

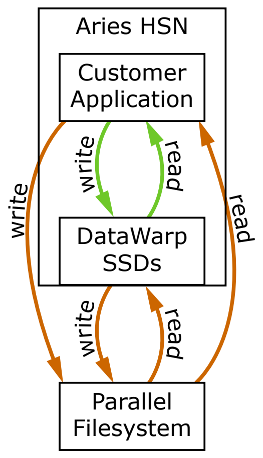
```
smw# cd /etc
smw# cd /var/tmp
smw# ls ./file
smw# su - crayadm
crayadm@smw> cd /usr/bin
crayadm@smw> ./command
```

2 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 Aries high speed network (HSN). I/O operations to this storage complete 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.

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



2.1 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.

2.2 DataWarp Limitations

For optimal use, it is important to understand DataWarp's limitations, including bugs, clarification on functionality, or warnings about the state of components.

SSD Write Protection

SSD write protection limitations include:

- I/O to swap files does not count against the SSD write protection policies.

- Violation of the write window policy is calculated on a per server basis. This can result in some processes seeing EROFS (Read-only file system) errors while others see no errors when interacting with the DataWarp file system.

Cache

Due to several problems identified with striped access mode for the cache file system, Cray advises customers not to use this feature at this time. However, load balanced access mode for the cache file system is functional.

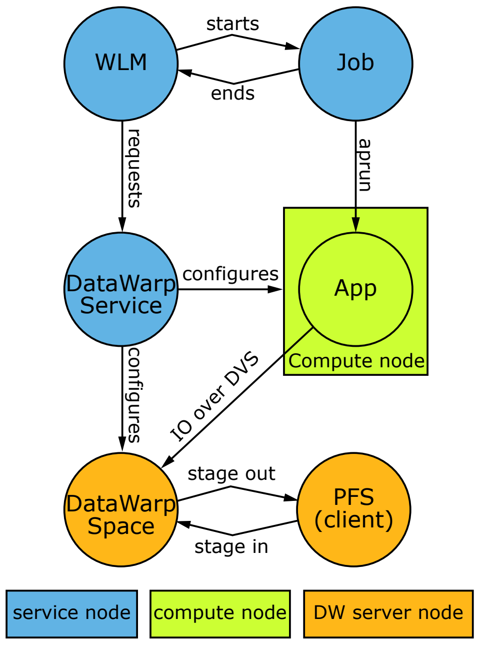
Scratch

The currently known caveats for scratch DataWarp are:

1. Scratch file system recovery
   - When a DataWarp server crashes, recovery is reliable when all file system servers are rebooted. If only the crashed DataWarp server is rebooted, the file system may not function properly.
2. Load balance
   - Scratch load balance functionality is not yet implemented.
3. Staging
   - Staging through the WLM interface is susceptible to timeouts. See *Stage In or Out Fails When Transferring a Large Number of Files* on page 125 for information on extending the timeout.
4. Striping
   - The libdatawarp `dw_set_stripe_configuration()` functionality is not implemented and returns ENOSYS.
5. Limits
   - Scratch file systems may have a directory depth up to 1000 directories deep. DataWarp service management tasks are subject to various resource limits, such as maximum open file descriptors. By default, the maximum number of open file descriptors per process is 1024. If a user creates a directory structure larger than 1000 directories deep, stage activity and teardown activity may fail. If deeper than 1000 directories deep is needed, the maximum number of open file descriptors per process limit can be increased on the DataWarp servers.

2.3 Overview of the DataWarp Process

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

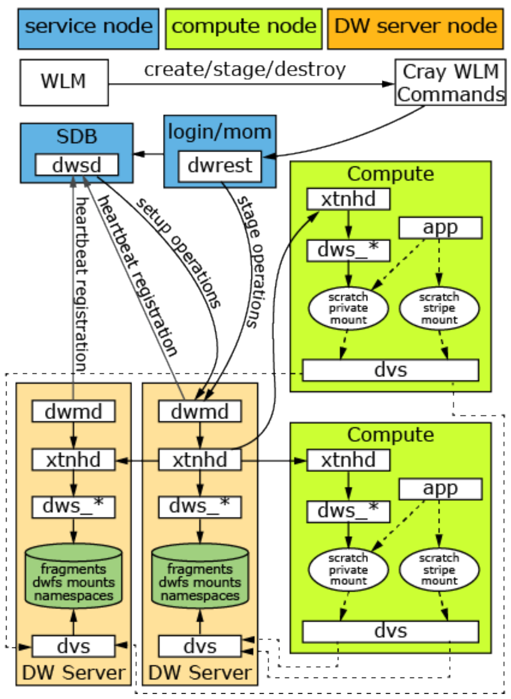


The diagram above provides a high level visual representation of the DataWarp process, as described here.

1. A user submits a job to a workload manager (WLM). 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 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 WLM requests the needed capacity and passes along other user-supplied configuration and staging requests.
3. The DataWarp service (DWS) dynamically assigns the storage and initiates the stage in process.
4. After this completes, the WLM acquires other resources needed for the batch job, such as compute nodes.
5. After the compute nodes are assigned, the WLM and DWS 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 WLM stages out files, if requested, and performs cleanup. First, the WLM releases the compute resources and requests that the DWS make the previously accessible DataWarp configuration inaccessible to the compute nodes. Next, the WLM requests that additional files, if any, are staged out. When this completes, the WLM tells the DWS 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



2.4 DataWarp Blade

The DataWarp blade is an I/O blade with the addition of solid-state drive (SSD) PCI cards, which allow the blade to run the DataWarp™ application I/O accelerator software. This software allocates storage dynamically in either private (dedicated) or shared modes.

The SSDs are half-height and half-length, off-the-shelf PCIe cards.

The DataWarp blade is a standard I/O blade with these characteristics:

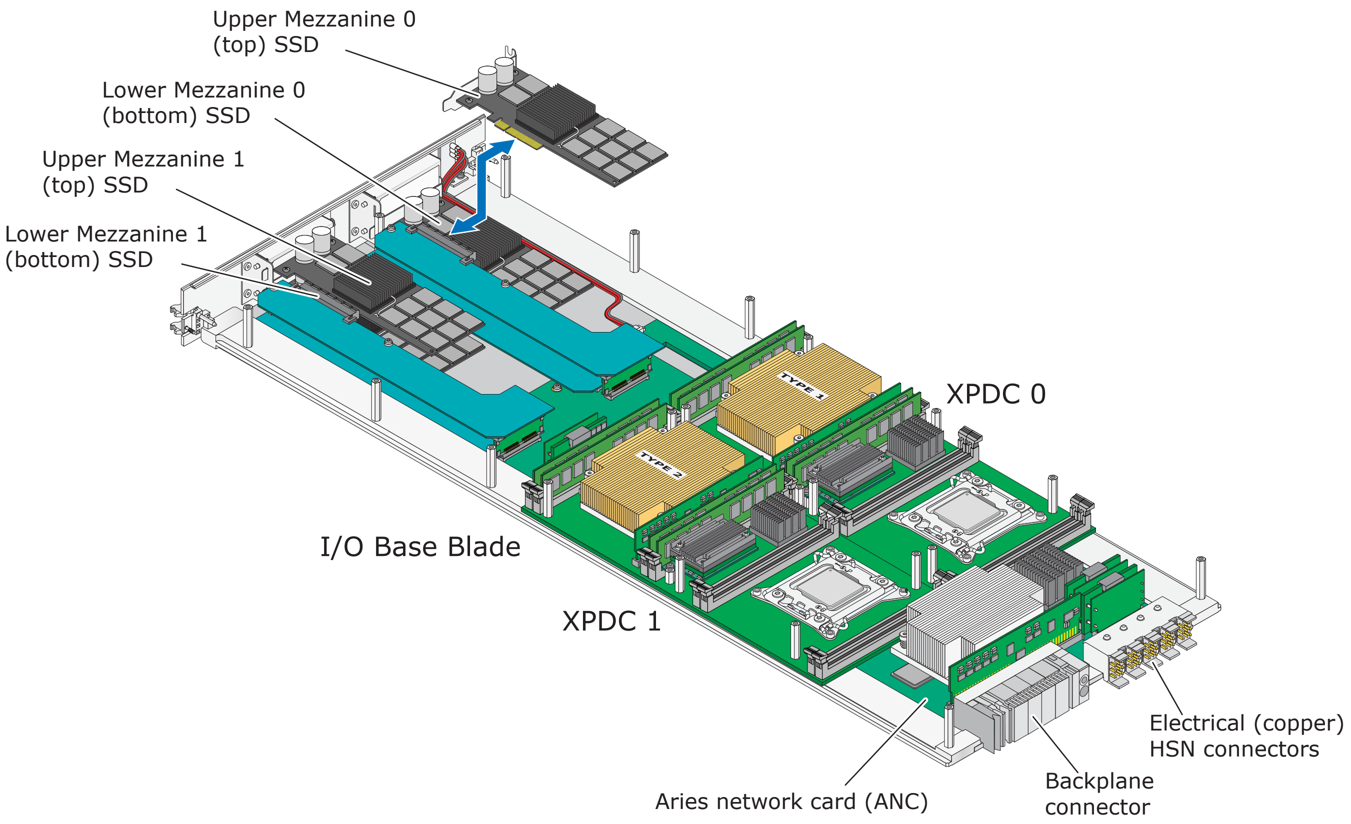
- Each node has a PDC with two Intel® Xeon® sockets, one of which is empty.
- Each node has two PCIe Gen3 x8 expansion slots.
- Both slots are populated with NVMe SSDs (Non-Volatile Memory Express host-controller interface). Note: SanDisk SSDs used on some early systems are not NVMe.

Either all Samsung or all Intel SSDs must be used on a node. Each Samsung SSD is a single device and therefore shows to the host as a single device. Each Intel SSD comprises two PCIe devices and therefore shows to the host as two devices.

At the lowest level, DataWarp uses the Linux logical volume manager to stripe all the local SSD devices into a single logical device. For Samsung there are two; and for Intel four devices show up on the PCIe buses. Each blade has a supported capacity of up to 6.4TB.

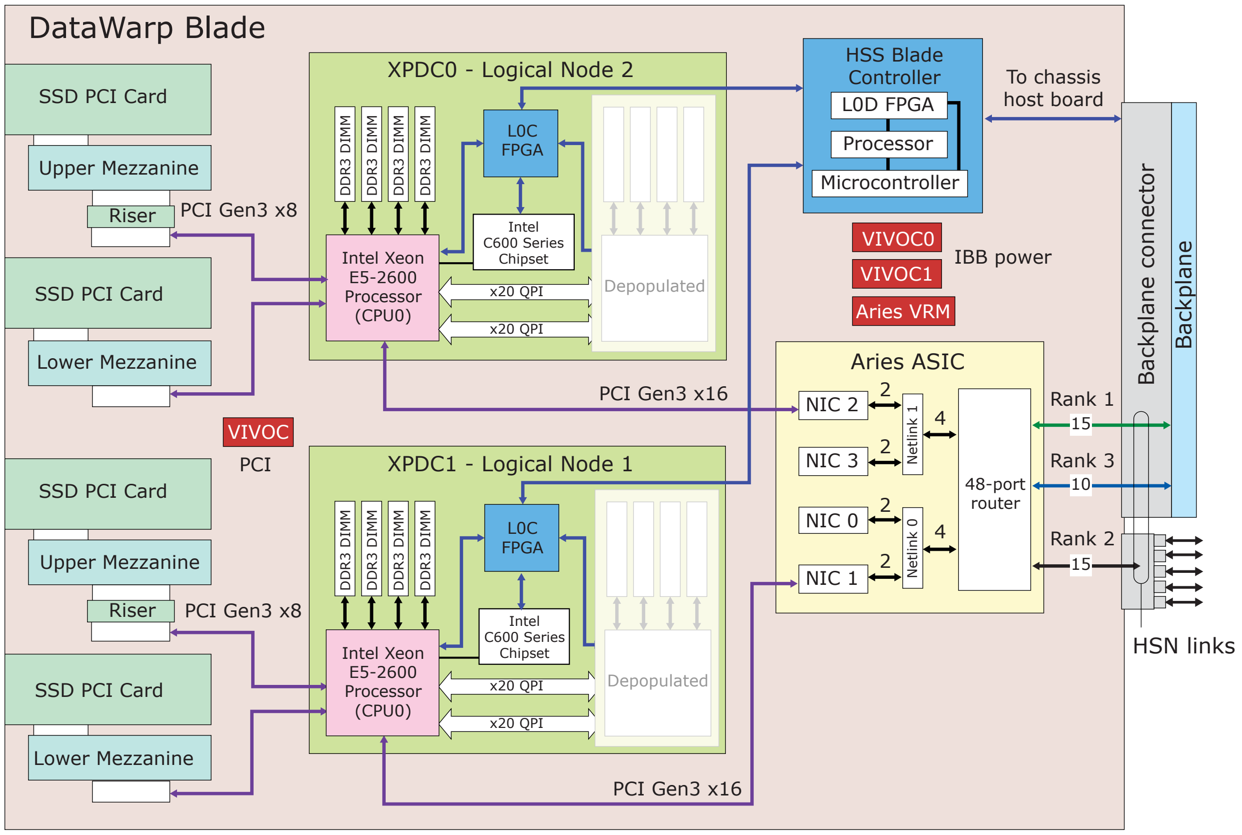
The DataWarp blade is physically laid out as shown in this figure.

Figure 4. DataWarp Blade Component Locations


Upper Mezzanine 0 (top) SSD
Lower Mezzanine 0 (bottom) SSD
Upper Mezzanine 1 (top) SSD
Lower Mezzanine 1 (bottom) SSD
TYPE 1
TYPE 2
XPDC 0
I/O Base Blade
XPDC 1
Electrical (copper) HSN connectors
Backplane connector
Aries network card (ANC)

The DataWarp blade functions logically as shown in the following figure. This is an I/O blade using the PCI cards to hold SSD cards. Details of components in this figure are described in *I/O Base Blade (IBB)* and *I/O Blade*.

Figure 5. DataWarp Blade Architecture



Reporting SSD Status

SSD modules and other components on the DataWarp blade are reported and identified as follows.

- `xtcheckssd`(8) command checks SSDs at boot and every 24 hours, with output to the console such as:

```
PCIe slot#:1,Name:INTEL SSDPECME040T4,SN:CVF8515300094P0DGN-1,Size:
4000GB,Remaining life:100%,Temperature:22(c)
```

- `xtdiagdata`(8) displays diagnostic records in two formats.

```
xtdiagdata ssd
```

- `capmc`(8) creates a JSON table of the SSDs.

```
module load capmc
capmc get_ssd_diags
```

- `xtcheckhss`(8) validates the health of HSS.
- `xthwinv`(8) gets information about ASIC, processor chip, and memory DIMMs on specified modules such as the DataWarp blade.

SSD products used in the DataWarp blade are made by Intel, Samsung, and Sandisk. When an SSD is replaced, the replacement must be from the same manufacturer and have the same size and performance as the one being replaced.

2.5 Identify Nodes with SSD Hardware

Identification of the nodes with SSD hardware is necessary to complete the DataWarp service installation. Use the `xtcheckhss` command if this information is not readily available.

The following example shows that the system has three nodes with SSD hardware, each with two PCIe expansion slots that are populated with Intel P3608 cards. Although not displayed as part of the `grep` output above, the column headers are as follows:

```
      Node Slot                                 Name Target Gen Trained Gen Target Width Trained Width
---------- ---- ------------------------------------- ---------- ----------- ------------ -------------
```

```
smw# xtcheckhss --nocolor --detail=f --pci |grep SSD
c0-0c0s4n1  2   Fusion-io_ioMemory3_Atomic_Series_SSD    Gen2       Gen2          x8           x8
c0-1c0s9n2  0/1 Intel_P3600_Series_SSD                   Gen3       Gen3          x4           x4
c0-1c0s9n2  0/1 Intel_P3600_Series_SSD                   Gen3       Gen3          x4           x4
c0-1c0s9n2  2   Intel_P3600_Series_SSD                   Gen3       Gen3          x4           x4
c0-1c0s9n2  2   Intel_P3600_Series_SSD                   Gen3       Gen3          x4           x4
c0-1c0s10n1 0/1 Samsung_SM1725_Series_SSD                Gen3       Gen3          x8           x8
c0-1c0s10n1 2   Samsung_SM1725_Series_SSD                Gen3       Gen3          x8           x8
c0-1c0s10n2 0/1 Intel_P3600_Series_SSD                   Gen3       Gen3          x4           x4
c0-1c0s10n2 0/1 Intel_P3600_Series_SSD                   Gen3       Gen3          x4           x4
c0-1c0s10n2 2   Intel_P3600_Series_SSD                   Gen3       Gen3          x4           x4
c0-1c0s10n2 2   Intel_P3600_Series_SSD                   Gen3       Gen3          x4           x4
c0-1c0s12n1 0/1 Intel_P3600_Series_SSD                   Gen3       Gen3          x4           x4
c0-1c0s12n1 0/1 Intel_P3600_Series_SSD                   Gen3       Gen3          x4           x4
c0-1c0s12n1 2   Intel_P3600_Series_SSD                   Gen3       Gen3          x4           x4
c0-1c0s12n1 2   Intel_P3600_Series_SSD                   Gen3       Gen3          x4           x4
```

3 Initial DataWarp Service (DWS) Installation

Prerequisites

This DataWarp installation procedure assumes the following:

- A Cray XC series system running CLE 6.0.UP04 with one or more nodes with SSD hardware
- This is an initial installation of CLE and not an update
- Identification (cname) of the nodes with SSD hardware (see *Identify Nodes with SSD Hardware* on page 15)

Additional Requirements

The following requirements can be implemented before or after the installation of DataWarp.

- A parallel file system (PFS) must be mounted in the same location on all compute nodes as well as all service nodes included in `managed_nodes_groups` 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.
- SanDisk/Fusion ioMemory3/SX300 SSD cards require firmware version 8.9.5. See *Update Fusion ioMemory Firmware* on page 68.

Other Important Things to Know

The DataWarp installation procedure includes running `cfgset update` to modify the attributes or content of a config set. Keep the following in mind while installing DataWarp.

⚠ **CAUTION: Boot failure possible if using cfgset under certain conditions.** The `cfgset create` and `cfgset update` commands always call pre- and post-configuration scripts. Some of these scripts require HSS daemons and other CLE services to be running. This can cause `cfgset` to fail if `xtbounce` is in progress or if the SMW is not connected to XC hardware. If `xtdiscover` is running, `cfgset` may hang or produce incorrect data that leads to system boot failure. In these situations, use `cfgset create` or `cfgset update` with the `--no-scripts` option to prevent running the scripts. When those situations are no longer in effect, run `cfgset update` without the `--no-scripts` option to ensure that all pre- and post-configuration scripts are run. Use the following command to see which HSS daemons are running:

```
smw# /etc/init.d/rsms status
```

Installation Overview

DataWarp is one of many services that store service configuration content in CLE configuration sets (config sets) on Cray systems. DataWarp can be configured when config sets are created during a fresh install or major upgrade, or it can be configured/reconfigured later by updating existing config sets during normal system operation (bearing in mind that some of the DataWarp module parameters are best set during initial system configuration). These procedures guide site administrators and staff in entering appropriate values for DataWarp configuration settings using the configurator. Whether sites enter values in an interactive configurator session or enter values in a configuration worksheet for bulk import, the configurator takes the supplied values and ensures that they become part of the config set being created or updated.

The initial installation process described in XC™ Series DataWarp™ Installation and Administration Guide modifies configuration worksheets and consists of the following tasks:

1. (Systems with Fusion IO SSD cards only) Integrate the driver software into the service node image; *Create a New Service Node Image for Fusion IO SSDs* on page 17.
2. Update `cray_node_groups`; *Update cray_node_groups for DataWarp* on page 19.
3. Verify that required services are enabled; *Ensure that cray_ipforward, cray_lnet, cray_munge, and cray_dw_wlm are Enabled* on page 21.
4. Update `cray_persistent_storage`; *Set Up DataWarp Persistent Storage* on page 26.
5. Configure `cray_dws` and validate the config set; *Configure the cray_dws Worksheet* on page 27.
6. System reboot.
7. (Optional) Enable and configure DataWarp accounting within the `cray_rur` service; *Enable and Configure Accounting* on page 31.
8. Post-boot configuration tasks; *Post-boot Configuration* on page 65.

3.1 Create a New Service Node Image for Fusion IO SSDs

Prerequisites

- A Cray XC series system with one or more Fusion IO (Sandisk) SSD cards installed
- Identification (cname) of nodes with SSD hardware (see *Identify Nodes with SSD Hardware* on page 15)

About this task

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

For more information about installation third-party software with a custom image, see *XC™ Series System Administration Guide (S-2393)*.

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_6.0up05_sles_12sp2_x86-64_ari
smw# recipe update --add-recipe service_cle_6.0up05_sles_12sp2_x86-64_ari \
fio-service_cle_6.0up05_sles_12sp2_x86-64_ari
smw# recipe update --add-coll datawarp-xtra_cle_6.0up05_sles_12sp2 fio-service_cle_6.0up05_sles_12sp2_x86-64_ari
smw# recipe update --add-repo passthrough-common_cle_6.0up05_sles_12sp2_x86-64 \
fio-service_cle_6.0up05_sles_12sp2_x86-64_ari
smw# recipe update --add-repo passthrough-common_cle_6.0up05_sles_12sp2_x86-64_updates \
fio-service_cle_6.0up05_sles_12sp2_x86-64_ari
smw# recipe update --add-repo common_cle_6.0up05_sles_12sp2_x86-64_ari \
fio-service_cle_6.0up05_sles_12sp2_x86-64_ari
smw# recipe update --add-repo common_cle_6.0up05_sles_12sp2_x86-64_ari_updates \
fio-service_cle_6.0up05_sles_12sp2_x86-64_ari
smw# recipe update --add-repo sle-server_12sp2_x86-64 fio-service_cle_6.0up05_sles_12sp2_x86-64_ari
smw# recipe update --add-repo sle-server_12sp2_x86-64_updates fio-service_cle_6.0up05_sles_12sp2_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_6.0up05_sles_12sp2_x86-64_ari"
        dest: "fio-service{note}_cle_{cle_release}-build{cle_build}{patch}_sles_12sp2-created{date}.cpio"
        nims_group: "fio-service"
  fio-service:
      - recipe: "fio-service_cle_6.0up05_sles_12sp2_x86-64_ari"
        dest: "fio-service{note}_cle_{cle_release}-build{cle_build}{patch}_sles_12sp2-created{date}.cpio"
        nims_group: "fio-service"
```

For example:

```
cray_image_groups:
  default:
      - recipe: "compute_cle_6.0up05_sles_12sp2_x86-64_ari"
        dest: "{compute{note}_cle_{cle_release}-build{cle_build}{patch}_sles_12sp2-created{date}.cpio"
        nims_group: "compute"
      - recipe: "login_cle_6.0up05_sles_12sp2_x86-64_ari"
        dest: "login{note}_cle_{cle_release}-build{cle_build}{patch}_sles_12sp2-created{date}.cpio"
        nims_group: "login"
      - recipe: "service_cle_6.0up05_sles_12sp2_x86-64_ari"
        dest: "service{note}_cle_{cle_release}-build{cle_build}{patch}_sles_12sp2-created{date}.cpio"
        nims_group: "service"
      - recipe: "fio-service_cle_6.0up05_sles_12sp2_x86-64_ari"
        dest: "fio-service{note}_cle_{cle_release}-build{cle_build}{patch}_sles_12sp2-created{date}.cpio"
        nims_group: "fio-service"
  fio-service:
      - recipe: "fio-service_cle_6.0up05_sles_12sp2_x86-64_ari"
        dest: "fio-service{note}_cle_{cle_release}-build{cle_build}{patch}_sles_12sp2-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, continue with the DataWarp installation; otherwise, execute the following command to instruct NIMS to use the new image.

```
smw# cnode update -p p0 --filter group=fio-service -i image_file
```

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

Next, configure the node groups required by DataWarp; *Update cray_node_groups for DataWarp* on page 19.

3.2 Update cray_node_groups for DataWarp

About this task

DataWarp requires:

- At least one node group containing only DataWarp service nodes
- At least one node group containing only DataWarp API gateway nodes (for most systems, `login_nodes`, the node group that contains all internal login nodes is acceptable)

These node groups are defined in `cray_node_groups` and referenced within the `cray_dws` service.

Procedure

1. Copy the cray_node_groups worksheet from the current config set to the CLE worksheet workarea.

   This ensures that the worksheet being edited reflects current configuration values.

   ```
   smw# cd /var/adm/cray/release/p0_worksheet_workarea
   smw# cp -a /var/opt/cray/imps/config/sets/p0/worksheets/\
   cray_node_groups_worksheet.yaml .
   ```

2. Determine if the `datawarp_nodes` node group is already defined within the `cray_node_groups` service.

   ```
   smw# cfgset search -s cray_node_groups -t datawarp p0
   ```

   The node group does not exist:

   ```
   # 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.
   ```

   The node group exists:

   ```
   # 3 matches for 'datawarp' from cray_node_groups_config.yaml
   #--------------------------------------------------------------------------------
   cray_node_groups.settings.groups.data.datawarp_nodes.description: Datawarp server nodes
   cray_node_groups.settings.groups.data.datawarp_nodes.members: c1-0c2s7n2, c1-0c2s7n1, c1-0c2s0n2
   cray_node_groups.settings.groups.data.datawarp_nodes.description: Datawarp server nodes
   ```

3. Display the `login_nodes` node group defined in the `cray_node_groups` service.

   ```
   smw# cfgset search -s cray_node_groups -t login_nodes p0
   # 4 matches for 'login_nodes' from cray_node_groups_config.yaml
   #--------------------------------------------------------------------------------
   cray_node_groups.settings.groups.data.login_nodes.description: Default node group which contains the
   login nodes for the configured system.
   cray_node_groups.settings.groups.data.login_nodes.members: c0-1c0s8n0, c0-1c0s8n1
   cray_node_groups.settings.groups.data.elogin_nodes.description: # (empty)
   cray_node_groups.settings.groups.data.elogin_nodes.members: elogin1, elogin2, elogin3
   ```

   If the nodes listed in steps *2* on page 19 and *3* on page 19 accurately reflect the system configuration, exit this procedure and continue with the DataWarp installation process.

4. Edit `cray_node_groups_worksheet.yaml`.

```
smw# vi cray_node_groups_worksheet.yaml
```

5. Proceed based on the results of the previous steps.

- Go to step *6* on page 20 to add a `datawarp_nodes` node group.
- Go to step *7* on page 20 to update an existing `datawarp_nodes` node group.
- Go to step *8* on page 20 if no changes are needed for the `datawarp_nodes` node group.

6. Add the `datawarp_nodes` node group if it does not already exist.

a. Copy the three commented lines under

```
** EXAMPLE 'groups' VALUE (with current defaults) **
```

and paste them in this location:

```
# NOTE: Place additional 'group' setting entries here, if desired.
#cray_node_groups.settings.groups.data.group_name.sample_key_a: null <--setting a multival key
#cray_node_groups.settings.groups.data.sample_key_a.description: ''
#cray_node_groups.settings.groups.data.sample_key_a.members: []
```

b. Uncomment the lines, and replace `sample_key_a` with `datawarp_nodes` in all lines.

c. Remove the `<-- setting a multival key` text at the end of the first line (note that the `null` value is required; do not remove or change it).

d. Set the `description` value to `DataWarp server nodes`.

e. Add DataWarp node names to the `members` list field. Add each cnode name on a separate line prefixed by a hyphen and space (`- `). For example:

```
# NOTE: Place additional 'group' setting entries here, if desired.
cray_node_groups.settings.groups.data.group_name.datawarp_nodes: null
cray_node_groups.settings.groups.data.datawarp_nodes.description: DataWarp server nodes
cray_node_groups.settings.groups.data.datawarp_nodes.members:
- c1-0c2s7n2
- c1-0c2s7n1
- c1-0c2s0n2
#*********** END Service Setting: groups **********
```

f. Proceed to step *8* on page 20.

7. Update pre-existing `datawarp_nodes.members` if not defined correctly.

Each cnode name must be on a separate line prefixed by a hyphen and space (`- `). For example:

```
cray_node_groups.settings.groups.data.datawarp_nodes.members:
- c1-0c2s7n2
- c1-0c2s7n1
- c1-0c2s0n2
```

8. Add an API gateway node group if not using the `login_nodes` node group. For most systems, the `login_nodes` node group is used.

a. Copy the three commented lines under

```
** EXAMPLE 'groups' VALUE (with current defaults) **
```

and paste them in this location:

```
# NOTE: Place additional 'group' setting entries here, if desired.
#cray_node_groups.settings.groups.data.group_name.sample_key_a: null <--setting a multival key
#cray_node_groups.settings.groups.data.sample_key_a.description: ''
#cray_node_groups.settings.groups.data.sample_key_a.members: []
```

b. Uncomment the lines, and replace `sample_key_a` with `dw_gateway_nodes` in all lines.

c. Remove the `<-- setting a multival key` text at the end of the first line (note that the `null` value is required; do not remove or change it).

d. Set the `description` field as `DataWarp API gateway nodes`.

e. Add DataWarp node names to the `members` list field. Add each cnode name on a separate line prefixed by a hyphen and space (-). For example:

```
# NOTE: Place additional 'group' setting entries here, if desired.
cray_node_groups.settings.groups.data.group_name.dw_gateway_nodes: null
cray_node_groups.settings.groups.data.dw_gateway_nodes.description: DataWarp API gateway nodes
cray_node_groups.settings.groups.data.dw_gateway_nodes.members:
- c0-1c0s8n0
#********** END Service Setting: groups **********
```

9. Save the changes, and upload the modified worksheet into the CLE config set.

```
smw# cfgset update --worksheet-path cray_node_groups_worksheet.yaml p0
INFO - Running pre-configuration scripts

INFO - Checking directory access

...

INFO - ConfigSet 'p0' has been updated.

INFO - Run 'cfgset search --level basic p0' to review the current settings.
```

Next: ensure that the `cray_ipforward`, `cray_lnet`, `cray_munge`, and `cray_dw_wlm` services are enabled.

3.3 Ensure that cray_ipforward, cray_lnet, cray_munge, and cray_dw_wlm are Enabled

About this task

The following configuration services must be enabled for DataWarp to function in a CLE 6.0 / SMW 8.0 system.

- cray_lnet
- cray_munge
- cray_dw_wlm
- cray_ipforward

This procedure provides two different methods to ensure that they are enabled. Use whichever method seems easiest.

Method 1 Use the configurator in interactive mode to check the status of these configuration services and enable them if needed.

Method 2 Use the configurator search function to check their status and use worksheets to enable them if needed.

Procedure

```
---------- METHOD 1: CONFIGURATOR IN INTERACTIVE MODE ----------

If using Method 2, skip this section and go to step 4 on page 23.
```

1. Invoke the configurator in interactive mode with level = advanced and state = all to view the list of services in the CLE config set p0.

   This example configurator screen shows the list of CLE services in config set p0. The necessary services are in bold and have a variety of statuses. The number of a service in the list can vary (the services are not always listed alphabetically), so if using the example commands that follow, substitute the correct number for the service being enabled.

```
smw# cfgset update -m interactive -l advanced -S all p0
```

```
Service Configuration List Menu (Config Set: p0, type: cle)

-------------------------------------------------------------------------------
  Selected    #    Service            Status (level=advanced, state=all)
-------------------------------------------------------------------------------
             1)   cray_alps          [ OK ]
             2)   cray_auth          [ OK ]
                 ...
             9)   cray_dw_wlm        valid, disabled, 6/7 unconfigured settings
                 ...
            17)   cray_ipforward     inheriting from global config
                 ...
            20)   cray_lnet          [ OK ]
                 ...
            27)   cray_munge         unconfigured service, 7/7 unconfigured settings
                 ...
-------------------------------------------------------------------------------
                 ...
Service List Menu [default: save & exit - Q] $
```

2. Ensure that cray_dw_wlm, cray_ipforward, cray_lnet, and cray_munge are enabled.

   Find each of the necessary services in the list of services.

   - If the status is `[ OK ]`, it is enabled. No change is needed.
   - If the status is disabled, enter the number of the service in the list to select it, then enter **E** to toggle it to enabled.

     ```
     Service List Menu [default: save & exit - Q] $ 9
     Service List Menu [default: configure - C] $ E
     ```

   - If the status is unconfigured, select it and toggle it to disabled, then select it and toggle it again to enabled.

```
Service List Menu [default: save & exit - Q] $ 27
Service List Menu [default: configure - C] $ E
Service List Menu [default: save & exit - Q] $ 27
Service List Menu [default: configure - C] $ E
```

- If the status of cray_ipforward is `inheriting from global config`, it must be checked in the global config set. Exit the configurator and continue to the next step to ensure that it is enabled.

When done checking all four necessary services, save any changes and exit the configurator. If there are no changes, enter **x** instead of **Q**.

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

3. (If cray_ipforward inherits from global) Ensure that cray_ipforward is enabled in the global config set.

a. Invoke the configurator in interactive mode with state = all to view the list of services in the global config set.

```
smw# cfgset update -m interactive -S all global
```

b. Ensure that cray_ipforward is enabled.

Find cray_ipforward in the list of global services.

- If the status is `[ OK ]`, it is enabled.
- If the status is disabled, select it (#4 in list in this example) and toggle it to enabled.

```
Service List Menu [default: save & exit - Q] $ 4
Service List Menu [default: configure - C] $ E
```

- If the status is unconfigured, select and toggle it twice to enable it.

```
Service List Menu [default: save & exit - Q] $ 4
Service List Menu [default: configure - C] $ E
Service List Menu [default: save & exit - Q] $ 4
Service List Menu [default: configure - C] $ E
```

c. Save any changes and exit the configurator. If there are no changes, enter **x** instead of **Q**.

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

If Method 1 used, the procedure is complete. Skip the steps for Method 2.

```
---------- METHOD 2: SEARCH + WORKSHEETS ----------
```

4. Determine whether the cray_lnet, cray_munge, and cray_dw_wlm services are enabled.

Use `cfgset search` to determine the status of these config services. Note that `-l advanced` must be added to the command to search for the status of cray_dw_wlm, because all settings in that service are level advanced.

```
smw# cfgset search -s cray_lnet --service-status p0 \
| grep lnet.enabled
cray_lnet.enabled: True

smw# cfgset search -s cray_munge --service-status p0 \
| grep munge.enabled
```

```
cray_munge.enabled: True

smw# cfgset search -s cray_dw_wlm --service-status -l advanced p0 \
| grep dw_wlm.enabled
cray_dw_wlm.enabled: True
```

In the example, these services are enabled. If any of these services is disabled (set to false), it can be enabled in a later step of this procedure.

5. Determine which config set to check for status of the cray_ipforward service.

Because the cray_ipforward service has both a global and CLE template, the CLE template can be configured to inherit settings from the global template. Therefore, to check the status of cray_ipforward, it is first necessary to determine which template governs the configuration of that service: global or CLE.

Search the cray_ipforward worksheet in the CLE config set (p0 in example) to find the `inherit` setting.

```
smw# grep inherit: /var/opt/cray/imps/config/sets/p0/worksheets/\
cray_ipforward_worksheet.yaml
```

- If the output of this command is `cray_ipforward.inherit: False`, then cray_ipforward does not inherit from global, so the CLE config set worksheet is the correct one to check to find the status of the service.
- If the output of this command is `cray_ipforward.inherit: True`, then cray_ipforward inherits from global, so the global config set worksheet is the correct one to check to find the status of the service.

6. Determine whether the cray_ipforward service is enabled.

a. (If cray_ipforward.inherit set to false) Determine whether cray_ipforward is enabled in the CLE config set (p0 in example).

```
smw# cfgset search -s cray_ipforward --service-status p0 \
| grep ipforward.enabled
cray_ipforward.enabled: true
```

In the example, cray_ipforward is enabled.

b. (If cray_ipforward.inherit set to true) Determine whether cray_ipforward is enabled in the global config set.

```
smw# cfgset search --service-status global | grep ipforward.enabled
cray_ipforward.enabled: true
```

In the example, cray_ipforward is enabled.

7. If any of the necessary services are disabled, copy current worksheets to the workareas for editing.

Copy CLE worksheets.

```
smw# cp -a /var/opt/cray/imps/config/sets/p0/worksheets/* \
/var/adm/cray/release/p0_worksheet_workarea/*
```

Copy global worksheets.

```
smw# cp -a /var/opt/cray/imps/config/sets/global/worksheets/* \
/var/adm/cray/release/global_worksheet_workarea/*
```

8. If cray_lnet is disabled, enable it.

a. Edit the worksheet.

```
smw# cd /var/adm/cray/release/p0_worksheet_workarea
smw# vi cray_lnet_worksheet.yaml
```

b. Enable cray_lnet.

If `cray_lnet.enabled` is commented out, uncomment it and ensure that it is set to `true`.

c. Import the updated worksheet to the config set.

```
smw# cfgset update --worksheet-path cray_lnet_worksheet.yaml p0
```

9. If cray_munge is disabled, enable it.

a. Edit the worksheet.

```
smw# cd /var/adm/cray/release/p0_worksheet_workarea
smw# vi cray_munge_worksheet.yaml
```

b. Enable cray_munge.

If `cray_munge.enabled` is commented out, uncomment it and ensure that it is set to `true`.

c. Import the updated worksheet to the config set.

```
smw# cfgset update --worksheet-path cray_munge_worksheet.yaml p0
```

10. If cray_dw_wlm is disabled, enable it.

a. Edit the worksheet.

```
smw# cd /var/adm/cray/release/p0_worksheet_workarea
smw# vi cray_dw_wlm_worksheet.yaml
```

b. Enable cray_dw_wlm.

If `cray_dw_wlm.enabled` is commented out, uncomment it and ensure that it is set to `true`.

c. Import the updated worksheet to the config set.

```
smw# cfgset update --worksheet-path cray_dw_wlm_worksheet.yaml p0
```

11. (If cray_ipforward.inherit set to false) If cray_ipforward is disabled, enable it in the CLE config set.

a. Edit the worksheet.

```
smw# cd /var/adm/cray/release/p0_worksheet_workarea
smw# vi cray_ipforward_worksheet.yaml
```

b. Enable cray_ipforward.

If `cray_ipforward.enabled` is commented out, uncomment it and ensure that it is set to `true`.

c. Import the updated worksheet to the config set.

```
smw# cfgset update --worksheet-path cray_ipforward_worksheet.yaml p0
```

12. (If cray_ipforward.inherit set to true) If cray_ipforward is disabled, enable it in the global config set.

a. Edit the worksheet.

```
smw# cd /var/adm/cray/release/global_worksheet_workarea
smw# vi cray_ipforward_worksheet.yaml
```

b. Enable cray_ipforward.

If `cray_ipforward.enabled` is commented out, uncomment it and ensure that it is set to `true`.

c. Import the updated worksheet to the config set.

```
smw# cfgset update --worksheet-path cray_ipforward_worksheet.yaml global
```

3.4 Set Up DataWarp Persistent Storage

Procedure

1. Copy the cray_persistent_data worksheet from the current config set to the CLE worksheet workarea.

This ensures that the worksheet being edited reflects current configuration values.

```
smw# cd /var/adm/cray/release/p0_worksheet_workarea
smw# cp -a /var/opt/cray/imps/config/sets/p0/worksheets/\
cray_persistent_data_worksheet.yaml .
```

2. Edit `cray_persistent_data_worksheet.yaml`.

```
smw# vi cray_persistent_data_worksheet.yaml
```

3. Ensure that cray_persistent_data is enabled.

If `cray_persistent_data.enabled` is commented out, uncomment it and ensure that it is set to `true`.

4. Configure persistent storage for DataWarp.

In the worksheet, copy the five lines below

```
# ** EXAMPLE 'mounts' VALUE (with current defaults) **
```

and paste them below

```
# NOTE: Place additional 'mounts' setting entries here, if desired.
```

```
# ** EXAMPLE 'mounts' VALUE (with current defaults) **
# cray_persistent_data.settings.mounts.data.mount_point.sample_key_a: null <-- setting a multival key
# cray_persistent_data.settings.mounts.data.sample_key_a.alt_storage_path: ''
# cray_persistent_data.settings.mounts.data.sample_key_a.options: ''
# cray_persistent_data.settings.mounts.data.sample_key_a.ancestor_def_perms: '0771'
# cray_persistent_data.settings.mounts.data.sample_key_a.client_groups: []
```

Uncomment the lines, replace `sample_key_a` with `/var/opt/cray/dws` in all lines, and remove the `<-- setting a multival key` text at the end of the first line (note that the `null` value is required; do not remove or change it). Change setting values as indicated in the example below.

```
# NOTE: Place additional 'mounts' setting entries here, if desired.
cray_persistent_data.settings.mounts.data.mount_point./var/opt/cray/dws: null
cray_persistent_data.settings.mounts.data./var/opt/cray/dws.alt_storage_path: ''
cray_persistent_data.settings.mounts.data./var/opt/cray/dws.options: rw
cray_persistent_data.settings.mounts.data./var/opt/cray/dws.ancestor_def_perms: '0755'
cray_persistent_data.settings.mounts.data./var/opt/cray/dws.client_groups:
- service_nodes

#************************ END Service Setting: mounts ************************
```

5. Save the changes, and upload the modified worksheet into the CLE config set.

```
smw# cfgset update --worksheet-path cray_persistent_data_worksheet.yaml p0
INFO - Running pre-configuration scripts

INFO - Checking directory access

...

INFO - ConfigSet 'p0' has been updated.

INFO - Run 'cfgset search --level basic p0' to review the current settings.
```

Next, configure the DataWarp service; *Configure the cray_dws Worksheet* on page 27.

3.5 Configure the cray_dws Worksheet

About this task

Several configuration parameters are required prior to booting and accessing the DataWarp nodes. The following steps correspond to the configuration settings available in the `cray_dws` worksheet, and step numbering reflects the order in which those settings appear there, unless noted otherwise.

CAUTION:

Configure DataWarp and all services only through the configurator or by placing/editing configuration files in the Simple Sync directory structure within the config set. Do not configure DataWarp by manually adding lines in any /etc files. In general, changes to those files are not persistent, and rebooting could result in loss of data.

Procedure

1. Copy the cray_dws worksheet from the current config set to the CLE worksheet workarea.

   This ensures that the worksheet being edited reflects current configuration values.

```
smw# cd /var/adm/cray/release/p0_worksheet_workarea
smw# cp -a /var/opt/cray/imps/config/sets/p0/worksheets/\
cray_dws_worksheet.yaml .
```

2. Edit `cray_dws_worksheet.yaml`.

```
smw# vi cray_dws_worksheet.yaml
```

3. Uncomment `cray_dws.enabled` and ensure that it is set to `true`.

   ```
   cray_dws.enabled: true
   ```

4. Set the managed nodes variable.

   Uncomment `#cray_dws.settings.service.data.managed_nodes_groups` and set it to a list of node groups that contain DataWarp nodes and have been defined in the `cray_node_groups` configuration.

   This example uses the `datawarp-nodes` node group defined in *Update cray_node_groups for DataWarp* on page 19.

   ```
   cray_dws.settings.service.data.managed_nodes_groups:
   - datawarp_nodes
   ```

5. Set the API gateway variable.

   Uncomment `cray_dws.settings.service.data.api_gateway_nodes_groups` and set it to a list of node groups that contain the DataWarp gateway nodes that have been defined in the `cray_node_groups` configuration. Typically this is the set of login nodes, but sites can define a subset of login nodes as gateway nodes. This example assumes that all nodes in the predefined `login_nodes` node group are gateway nodes.

   ```
   cray_dws.settings.service.data.api_gateway_nodes_groups:
   - login_nodes
   ```

6. (Optional) Set the external API gateway variable.

   This is necessary to provide native access to DataWarp commands from eLogin nodes.

   a. Uncomment `#cray_dws.settings.service.data.external_api_gateway_hostnames` and set it to a list of the fully qualified domain names (FQDN), also known as DNS Authoritative name records, of nodes specified in `api_gateway_nodes_groups`.

      ```
      cray_dws.settings.service.data.external_api_gateway_hostnames:
      - syslog1.us.cray.com
      - syslog2.us.cray.com
      ```

   b. Additionally, before executing DataWarp commands from eLogin nodes, the eLogin image config set must be defined with the MUNGE credential omitted from the exclusion list located at /etc/opt/cray/elogin/exclude_lists/elogin_cfgset_excludelist. This is accomplished by either deleting or commenting out the MUNGE entry prior to pushing the config set to the eLogin nodes. For details, see *XC™ Series eLogin Installation Guide* (S-2566), which is available at http://pubs.cray.com.

   `dwrest_cacheroot_whitelist` and `dwrest_cachemount_whitelist`: the next two steps for settings `dwrest_cacheroot_whitelist` and `dwrest_cachemount_whitelist` are switched. Although `dwrest_cacheroot_whitelist` is sequentially next in the worksheet, it introduces a potential security issue that does not exist if using `dwrest_cachemount_whitelist`. Therefore, Cray recommends using `dwrest_cachemount_whitelist`.

7. Set `dwrest_cachemount_whitelist`, if desired.

   `dwrest_cachemount_whitelist` is a list of PFS directories that users are allowed to use as mounted cache file systems. The PFS paths must be set up on all DataWarp service nodes for cache configurations to function.

Only the directories listed for `dwrest_cachemount_whitelist` are allowed for cache mount file systems whereas, `dwrest_cacheroot_whitelist` contains paths on which users are allowed to mount cache file systems. `dwrest_cachemount_whitelist` is more restrictive than `dwrest_cacheroot_whitelist`.

`dwrest_cachemount_whitelist` is not a required setting and can be defined with 0 entries. These two settings can be used either jointly or separately, but at least one must be defined for DataWarp caching to work.

If `dwrest_cacheroot_whitelist` and `dwrest_cachemount_whitelist` are both defined, the process used for determining if a user-specified cache directory is valid is as follows:

1. Is it an acceptable path given the value for `dwrest_cacheroot_whitelist`? If yes, the request succeeds; else,
2. Is it an acceptable path given the value for `dwrest_cachemount_whitelist`? If yes, the request succeeds; else,
3. The request fails.

a. Uncomment `#cray_dws.settings.service.data.dwrest_cachemount_whitelist` and set it to a list of PFS directories that users are allowed to use as mounted cache file systems.

```
cray_dws.settings.service.data.dwrest_cacheroot_whitelist:
- /pfs/path/user1
- /pfs/path/user2
```

8. Set `dwrest_cacheroot_whitelist`, if desired.

WARNING: Use of `dwrest_cacheroot_whitelist` introduces a potential security issue that depends on the permissions of the parent directories used for DataWarp cache. This issue and workarounds are described here.

`dwrest_cacheroot_whitelist` is a list of PFS path prefixes on which users are allowed to mount cache file systems. For example, if `dwrest_cacheroot_whitelist` is set as `/lus/users`, a user can mount the cache file system `/lus/users/seymour`. The PFS paths specified 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.

`dwrest_cacheroot_whitelist` is not a required setting and can be defined with 0 entries. For a more restrictive setting, sites can specify a list of specific directories as cache file systems by defining `dwrest_cachemount_whitelist` in the next step. These two settings can be used either jointly or separately, but at least one must be defined for DataWarp caching to work.

If `dwrest_cacheroot_whitelist` and `dwrest_cachemount_whitelist` are both defined, the process used for determining if a user-specified cache directory is valid is as follows:

1. Is it an acceptable path given the value for `dwrest_cacheroot_whitelist`? If yes, the request succeeds; else,
2. Is it an acceptable path given the value for `dwrest_cachemount_whitelist`? If yes, the request succeeds; else,
3. The request fails.

With the DataWarp caching file system, dwcfs, users can specify which PFS directory to cache. For example, a user may wish to cache `/lus/users/seymour/data` rather than all of `/lus/users`. When dwcfs is mounted and made available to compute nodes via DVS, users can only interact with the files in `/lus/users/seymour/data` or lower.

The security issue is possible when interacting with the cached version of `/lus/users/seymour/data`, because dwcfs only honors the file system permissions at `/lus/users/seymour/data` and lower, and not the permissions of the parent directories of the path being cached. Suppose a user does not have execute access to `/lus/users/seymour`, but the directory `/lus/users/seymour/data` is world readable, writeable, and executable. If interacting directly with the PFS, the user cannot access `/lus/users/seymour/data` or any of its files because of the file permissions set on the parent directory. But, if the same user requests `type=cache` access to `/lus/users/seymour/data`, the user can now access and modify files at this level or lower.

The following conditions are necessary for a user to exploit this issue:

- The user has access to DataWarp
- The user knows of the existence of `/lus/users/seymour/data`
- the dwrest_cacheroot_whitelist setting includes any of the following:
  - `/lus/`
  - `/lus/users/`
  - `/lus/users/seymour/`
  - `/lus/users/seymour/data`

To avoid this issue, use `dwrest_cachemount_whitelist` to define specific cache mount points.

a. Uncomment `#cray_dws.settings.service.data.dwrest_cacheroot_whitelist` and set it to a list of PFS paths on which users are allowed to mount cache file systems.

```
cray_dws.settings.service.data.dwrest_cacheroot_whitelist:
- /pfs/path/free
- /pfs/path/scratch
- /pfs/path/notbackedup
```

9. Set `allow_dws_cli_from_computes`, if desired.

`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.settings.service.data.allow_dws_cli_from_computes: false
```

10. Exit the editor, then import the complete DataWarp worksheet by updating the config set and specifying the worksheet path.

```
smw# cfgset update --worksheet-path cray_dws_worksheet.yaml' p0
```

When the config set is updated using the configurator, all of the pre-and post-configuration scripts are run.

11. 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
...
INFO - ConfigSet 'p0' is valid.
```

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

DWS is now enabled as part of CLE.

Next, post boot configuration procedures are necessary to define a functional DWS state; see *Post-boot Configuration* on page 65.

3.6 Enable and Configure Accounting

DataWarp accounting is enabled and configured through Cray's Resource Utilization Reporting service (`cray_rur`). RUR supports a plugin architecture, allowing many types of usage data to be collected while using the same software infrastructure. Cray provides data plugins (`dws`, `dws_job_server`, `dws_server`) for collecting DataWarp usage statistics. These plugins can be added to `cray_rur` at any time and does not require a system reboot. For the complete procedure, see *XC™ Series System Administration Guide*.

4 DataWarp Update Following CLE Update

Prerequisities

The DataWarp update procedures assume the following:

- System is currently running CLE 6.0.UP01, CLE 6.0.UP02, or CLE 6.0UP03
- SanDisk/Fusion ioMemory3/SX300 SSD cards require firmware version 8.9.5. See *Update Fusion ioMemory Firmware* on page 68.

The CLE software update brings in all new configuration templates. During the update process, the configurator runs in auto mode to merge the new content with CLE and global config sets already on the system. All config sets are updated and validated. Everything is in place for `cray_dws` to function properly in an updated environment, there are no additional software update procedures for DataWarp. That said, Cray recommends reviewing all DataWarp settings to verify that they are as expected. Use the following procedures to ensure that DataWarp is configured as desired.

Other Important Things to Know

The DataWarp installation procedure includes running `cfgset update` to modify the attributes or content of a config set. Keep the following in mind while installing DataWarp.

CAUTION: Boot failure possible if using cfgset under certain conditions. The `cfgset create` and `cfgset update` commands always call pre- and post-configuration scripts. Some of these scripts require HSS daemons and other CLE services to be running. This can cause `cfgset` to fail if `xtbounce` is in progress or if the SMW is not connected to XC hardware. If `xtdiscover` is running, `cfgset` may hang or produce incorrect data that leads to system boot failure. In these situations, use `cfgset create` or `cfgset update` with the `--no-scripts` option to prevent running the scripts. When those situations are no longer in effect, run `cfgset update` without the `--no-scripts` option to ensure that all pre- and post-configuration scripts are run. Use the following command to see which HSS daemons are running:

```
smw# /etc/init.d/rsms status
```

IMPORTANT:

For sites updating DataWarp from CLE 6.0.UP01 to CLE6.0.UP02 or above. Cray Field Notice (FN) #6121a describes a performance issue caused by the incorrect over-provisioning of Intel P3608 SSDs. Sites with P3608 SSDs that followed the over-provisioning procedure described in a DataWarp installation guide published prior to the dates listed below **must** complete the recommended fix described within the FN if this has not yet been done.

Corrected versions of the following DataWarp installation guides were available from the Cray release team and on the Cray Publications Portal at *http://pubs.cray.com* on October 17, 2016:

- *XC™ Series DataWarp™ Installation and Administration Guide (CLE 6.0.UP01) S-2564 Rev C*
- *DataWarp™ Installation and Configuration Guide S-2547-5204c*

4.1 Remove Existing Cache Configurations Before Initiating Updated DWS

Prerequisites

This procedure assumes the system is being updated from CLE 6.0.UP01. Skip this procedure for systems updating from CLE 6.0.UP02 or later.

About this task

DataWarp cache configurations created in CLE 6.0.UP01 are not usable in CLE 6.0.UP02 or later and must be removed. If a CLE 6.0.UP01 cache configuration is detected by a CLE 6.0.UP02 or later DataWarp manager daemon (`dwmd`), the configuration's fuse is blown and messages similar to the following are written to `dwmd.log`:

```
2017-05-13 16:06:01 (3519): <77> [testsys]: (cid:1,sid:1,stoken:xxyzz)
dws_realm_member ERROR:realm_member_create2 1 failed: Found old cache_dir setup
in /var/opt/cray/dws/mounts/fragments/1
...
2017-05-13 16:06:01 (3519): <77> [testsys]: RuntimeError: Found old cache_dir
setup in /var/opt/cray/dws/mounts/fragments/1
...
```

Procedure

1. Check for existing cache configurations.

   ```
   hostname# module load dws
   hostname# dwstat --cache configurations
   conf state inst  type  amode activs backing_path
     19 CA---   753 cache stripe      1   /usr/local
   ```

   Exit this procedure if there are no existing cache configurations.

2. Find the session corresponding to the instance for this configuration.

   ```
   hostname# dwstat instances
   inst state sess  bytes nodes             created expiration intact  label  public confs
    753 CA---   832 128GiB     1 2017-05-08T16:20:36      never   true I832-0 private     1
   ```

3. Remove the session.

   The command to remove a session is:

   ```
   dwcli rm session --id sid
   ```

   ```
   hostname# dwcli rm session --id 832
   ```

4.2 Verify DataWarp Service Update

Prerequisites

This procedure assumes:

- An XC series system with one or more nodes with SSD hardware
- CLE has been updated by following the instructions in XC™ Series Software Installation and Configuration Guide

About this task

During the CLE update procedure, the `cray_dws` service template was updated and a new template for `cray_dw_wlm` was added. This procedure verifies that all settings are as expected.

Procedure

1. Display and verify the `cray_dws` settings.

   ```
   smw# cfgset search -l advanced -s cray_dws p0
   ```

   The configurator displays the basic settings, any advanced settings that have been modified, and some advanced settings that are currently set to their default values.

   ```
   smw# cfgset search -l advanced -s cray_dws p0

   INFO - Checking services for valid YAML syntax
   INFO - Checking services for schema compliance
   # 11 matches for '.' from cray_dws_config.yaml
   #--------------------------------------------------------------------------------
   cray_dws.settings.service.data.managed_nodes_groups: datawarp_nodes
   cray_dws.settings.service.data.api_gateway_nodes_groups: login_nodes
   cray_dws.settings.service.data.external_api_gateway_hostnames: [ ] # (empty)
   cray_dws.settings.service.data.dwrest_cacheroot_whitelist: /lus/scratch
   cray_dws.settings.service.data.dwrest_cachemount_whitelist: [ ] # (empty)
   cray_dws.settings.service.data.allow_dws_cli_from_computes: false
   cray_dws.settings.service.data.lvm_issue_discards: 0
   cray_dws.settings.dwmd.data.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
   cray_dws.settings.dwsd.data.dwsd_conf: log_mask: 0x7,
   instance_optimization_default: bandwidth, scratch_limit_action: 0x3
   cray_dws.settings.dwrest.data.dwrest_conf: port: 2015
   cray_dws.settings.dwrestgun.data.dwrestgun_conf: max_requests=1024
   ```

2. Verify that `cray_dws` configuration is as expected, and correct any discrepencies before proceeding.

   a. Verify that `managed_nodes_groups` is defined as one or more node groups that contain the cnames of the DataWarp service nodes to be managed by DWS.

b. Verify that `api_gateway_nodes_groups` is defined either as `login_nodes`, the group of internal login nodes defined within `cray_node_groups`, or as a node group consisting of a subset of `login_nodes`.

c. Verify that `external_api_gateway_hostnames` is a list of fully-qualified domain names (FQDN) of internal login nodes with external connectivity selected to run the `dwrest` service.

d. Verify that `dwrest_cacheroot_whitelist` is defined as expected.

e. Verify `dwrest_cachemount_whitelist` is defined as expected.

f. Verify `allow_dws_cli_from_computes` is defined as expected.

3. Verify `cray_dws` advanced settings if any have been modified, and correct any discrepencies before proceeding.

Some sites choose to modify certain DWS advanced settings in response to their workload specifics. Those settings are displayed and Cray recommends verifying them.

TIP:

The configurator displays a handful of advanced settings, whether or not they have been modified, when invoked with `-l advanced`. For convenience, these settings and their default values are listed here for comparison.

All other displayed advanced settings have been modified by the site.

```
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
log_mask: 0x7
instance_optimization_default: bandwidth
scratch_limit_action: 0x3
port: 2015
max_requests=1024
```

Next, verify that other necessary services are enabled.

4.3 Verify Settings of Required Services

Prerequisites

This procedure assumes:

- An XC series system with one or more nodes with SSD hardware
- CLE has been updated by following the instructions in *XC™ Series Software Installation and Configuration Guide*

About this task

During the update of CLE, all service templates were updated. This procedure verifies that the services required by DWS are enabled and configured correctly, if applicable.

Procedure

1. Verify that `cray_ipforward`, `cray_lnet`, `cray_munge`, and `cray_dw_wlm` are enabled; *Ensure that cray_ipforward, cray_lnet, cray_munge, and cray_dw_wlm are Enabled* on page 21.

2. Verify that a persistent directory entry for DataWarp exists in the `mounts` setting of `cray_persistent_data`.

   ```
   smw# cfgset search -s cray_persistent_data -l advanced p0 |grep dws
   cray_persistent_data.settings.mounts.data./var/opt/cray/dws.alt_storage_path: # (empty)
   cray_persistent_data.settings.mounts.data./var/opt/cray/dws.options: # (empty)
   cray_persistent_data.settings.mounts.data./var/opt/cray/dws.ancestor_def_perms: 0771
   cray_persistent_data.settings.mounts.data./var/opt/cray/dws.client_groups: service_nodes
   ```

   If a persistent directory entry for DataWarp doesn't exist, it's likely that it was not configured in the previous version of CLE running on the system. Because of this, all storage pool information is lost. It is necessary to add `/var/opt/cray/dws` (or site-specific persistent directory) to the `mounts` setting, as described in *Set Up DataWarp Persistent Storage* on page 26.

3. 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
   ...
   INFO - ConfigSet 'p0' is valid.
   ```

4. Reboot the system following the typical procedure in order to activate any changes to `cray_dws`, `cray_ipforward`, `cray_munge`, `cray_persistent_data`, or `cray_dw_wlm`.

   Reboot is only necessary if changes were made to the DWS service or any of the required services.

DataWarp is now enabled as part of CLE. Cray recommends verifying that the site's pool configurations are as expected.

5 DataWarp Concepts

For basic definitions, refer to *Terminology* on page 129.

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 and accessibility managed by whatever creates it. 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.

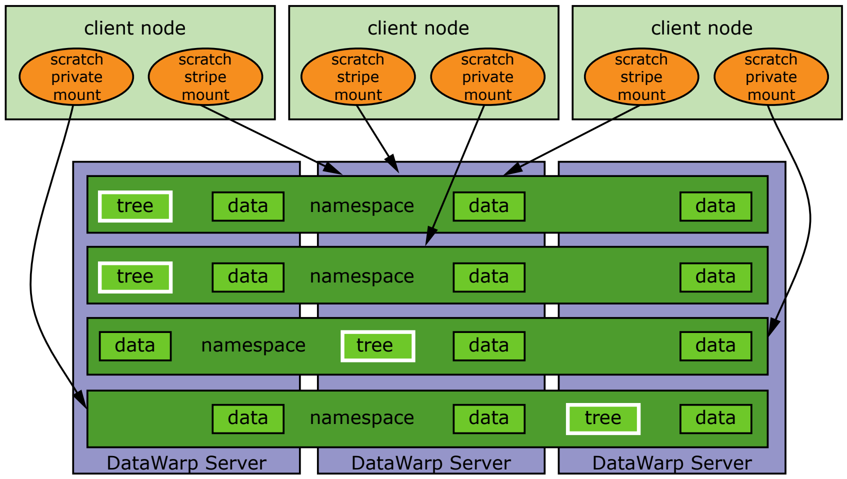
- **Private**: In private access mode individual files are also striped across multiple DataWarp server nodes (also aggregating both capacity and bandwidth *per file*), but the files are accessible only to the compute node that created them (e.g., /tmp). Private access is not supported for persistent 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*.

TIP: The default settings for scratch configuration access modes changed beginning with the CLE 6.0.UP01 release. This affects users whose systems have recently been upgraded from CLE 5.2.UP04 or CLE 6.0.UP00 to this release. The differences are pointed out in the following information and diagrams.

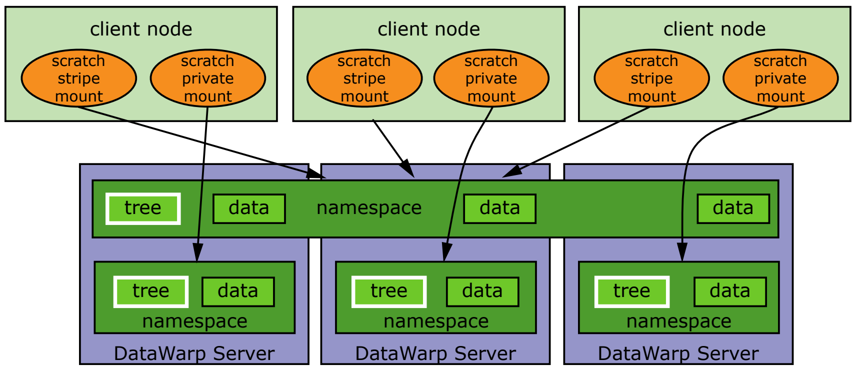
The following diagram shows a scratch private and scratch stripe mount point on each of three compute (client) nodes in a DataWarp installation configured with default settings; where `tree` represents which node manages metadata for the namespace, and `data` represents where file data may be stored. 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 6. 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 7. Scratch Configuration Access Modes (with `scratch_private_stripe=no`)



Cache Configuration I/O

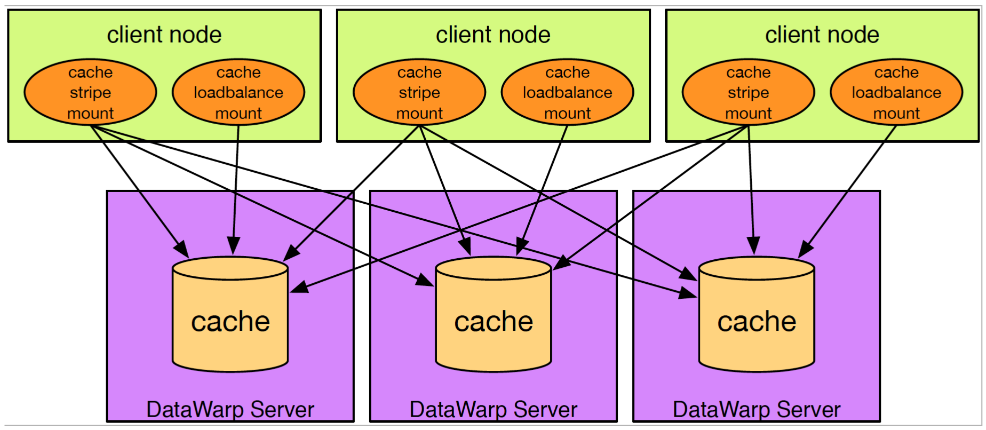
A cache configuration is accessed in one or more of the following ways:

- **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 8. Cache Configuration Access Modes

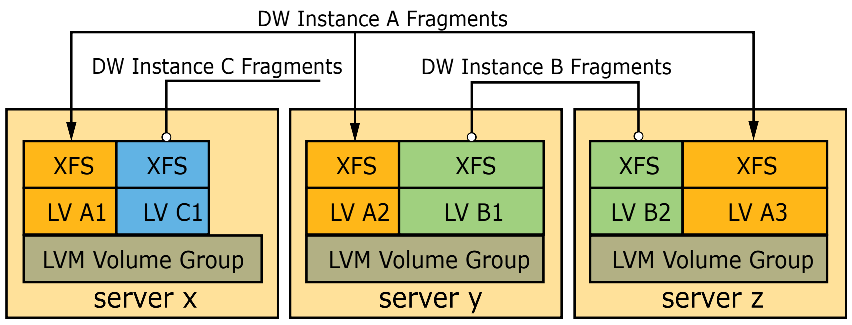


5.1 Instances and Fragments - a Detailed Look

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.

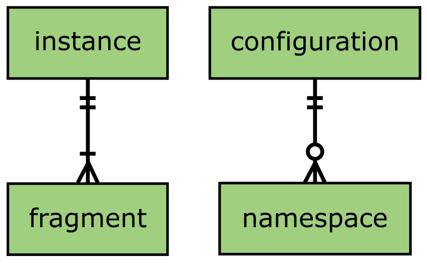
The following figure is an example of three DataWarp instances. DataWarp instance A consists of fragments that map to LVM logical volumes A1, A2, and A3 on servers x, y, z, respectively. DataWarp Instance B consists of fragments that map to LVM logical volumes y and z, respectively. DataWarp Instance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

Figure 9. 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 10. Instance-Fragment and Configuration-Namespace Relationships



5.2 Storage Pools

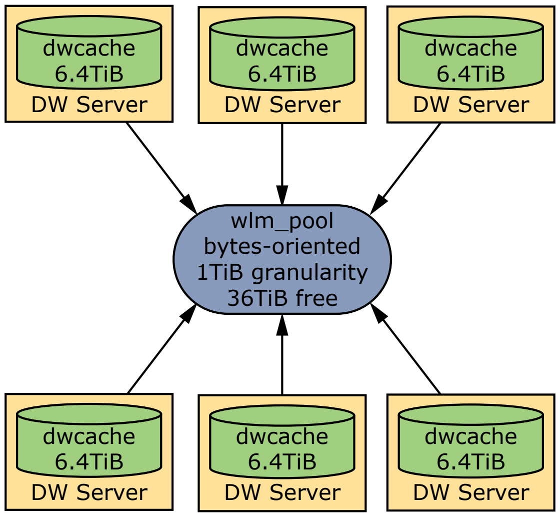
A storage pool groups nodes with storage together such that requests for space made against the pool are fulfilled from the nodes associated with the pool with a common allocation granularity. Pools have either byte or node allocation granularity (*`pool_AG`*). This release of DataWarp only supports 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 during SSD initialization) 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 11. Storage Pool Example



For an in depth look at pools, see *Storage Pool Configuration Guidelines* on page 74.

5.2.1 Why Does the Free Capacity Displayed by `dwstat pools` not Match the Quantity Capacity?

There are several reasons the free capacity displayed by `dwstat pools` may not match the quantity capacity:

- One or more nodes in the pool is offline
- One or more nodes in the pool is marked drain
- The dwsd.yaml `equalize_fragments` and `equalize_fragments_guarantee` settings are both enabled

In earlier versions, free capacity indicated how much storage capacity was available for use in that pool at that moment in time. However, when the dwsd.yaml `equalize_fragments` and `equalize_fragments_guarantee` settings are both enabled (default starting with CLE 6.0.UP05), free capacity takes on a new meaning, which is the largest capacity instance that can be requested using that pool at that moment in time. This difference leads to the following non-intuitive behavior when both settings are enabled.

- The free capacity may never equal the quantity capacity. For example, this happens if a pool has two nodes but the contributing capacity of each node is unequal.

- When an instance is created, the free capacity is reduced by up to the instance capacity size. In other words, after creating an instance, the free capacity may remain the same.
- When an instance is removed, the free capacity is increased by up to the instance capacity size. In other words, after removing an instance, the free capacity may remain the same.

5.3 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. By default, `--wait` is set, 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 `--no-wait` | `--haste` 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 wait
```

Instruct the DWS to abort the normal registration removal tasks by setting `--haste` with the following `dwcli` command:

```
user@login> dwcli update registration --id 2 --haste
update request for registrations entity with id = 2 accepted, "dwstat
registrations" for status
```

WARNING: Use of `--haste` 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. Although the DataWarp-integrated WLMs are exepected to automatically set `--haste` for registrations when the WLM-specific job removal force option is used, not all have implemented this feature. Therefore, it is necessary to manually set `--haste` using `dwcli` for registrations without a corresponding WLM batch job to force remove, and may also be necessary for some WLMs. For example:

```
login# dwcli update registration --id 1 --haste
update request for registrations entity with id = 1 accepted, "dwstat
registrations" for status
```

6 Advanced DataWarp Concepts

6.1 DVS Client-side Caching can Improve DataWarp Performance

With the advent of DataWarp and faster backing storage, the overhead of network operations has become an increasingly large portion of overall file system operation latency. DVS provides the ability to cache both read and write data on a client node while preserving close-to-open coherency and without contributing to out-of-memory issues on compute nodes. Instead of using network communication for all read/write operations, DVS can aggregate those operations and reuse data already read by or written from a client. This can provide a substantial performance benefit for these I/O patterns, which typically bear the additional cost of network latency:

- small reads and writes
- reads following writes
- multiple reads of the same data

Client-side Write-back Caching may not be Suitable for all Applications

CAUTION: Possible data corruption or performance penalty!

Using the page cache may not provide a benefit for all applications. Applications that require very large reads or writes may find that introducing the overhead of managing the page cache slows down I/O handling. Benefit can also depend on write access patterns: small, random writes may not perform as well as sequential writes. This is due to pages being aggregated for write-back. If random writes do not access sequential pages, then less-than-optimal-sized write-backs may have to be issued when a break in contiguous cache pages is encountered.

More important, successful use of write-back caching on client nodes requires a clear understanding and acceptance of the limitations of close-to-open coherency. It is important for site system administrators to ensure that users at their site understand how client-side write-back caching works before enabling it. Without that understanding, users could experience data corruption issues.

For detailed information about DVS client-side caching, see *XC™ Series DVS Administration Guide* (S-0005).

6.1.1 Client-side Caching Options

Although many workloads can benefit from client-side caching because it can reduce the frequency and necessity of network operations, others will be negatively affected due to the coherency characteristics of the implementation. Therefore, DWS includes both administrator-controlled default configuration settings and DataWarp job script command line options that enable users to opt in or opt out of client-side caching on a per-activation basis.

System Configuration Options

Two `dwsd` system-level configuration options (considered advanced settings) control the default values for the client-side caching attribute. They are:

`activation_cache_default`

Specifies the default for client-side caching on activation objects for activations that support the feature. This includes scratch and cache but excludes swap. For now, this only applies to stripe access modes. With client-side caching enabled, application performance can be greatly improved but applications must take special care to avoid having application processes on separate client nodes write to the same page.

Default: `off`.

`activation_private_cache_default`

Specifies the default for client-side caching on activation objects of private access mode configurations. With stripe access mode, multiple processes on separate nodes can write to the same page, which can lead to what appears to be data corruption. With private access mode, this is not possible since only one client node can ever access a particular file. Consequently the default for client-side caching for private access mode is separately configurable.

Default: `on`.

⚠ **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 on how to change these default settings, see *Modify DWS Advanced Settings* on page 96.

Client-side caching defaults are enumerated by activation type and access mode in the following table:

Table 1. Default Client-side Caching Configuration Settings

Activation Type	Access Mode	Default
scratch	stripe	`activation_cache_default`
scratch	private	`activation_private_cache_default`
cache	stripe	`activation_cache_default`
cache	ldbalance	always on

Note that read-only activations always have the client-side caching attribute enabled.

User-defined Options

Users opt in or out of client-side caching on a per-access mode basis via the `#DW jobdw` and `#DW persistentdw` job script commands. For example, the following command requests a scratch striped instance with client-side caching enabled and no more than 1000 files able to be created:

```
#DW jobdw type=scratch access_mode=striped(MFC=1000,client_cache=yes)
```

For further details, see *XC™ Series DataWarp™ User Guide* (S-2558).

6.2 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 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.

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.

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 change the method by which `dwsd` decides how to select space when creating instances, the `instance_optimization_default` setting has three valid options from which to choose. 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 `instance_optimization_default` setting is defined in `dwsd.yaml` as:

```
# When processing an instance-create request, dwsd decides how to select space
# across all of the servers that are under the control of the DWS. It first
# restricts selection to server nodes that belong to the pool specified in the
# instance create request itself and are up and responding. It then carves out
# space from each server according to some policy. When the instance create
# request does not specify a policy, the instance_optimization_default
# option is used as the default.

# Valid options:
```

```
#     bandwidth - pick as many server nodes as possible
#     interference - pick as few server nodes as possible, and pick nodes that are
#     unused when possible
#     wear - pick nodes based on the health of the underlying block devices. Note
#     that the other options only take device health into account to break ties.

# instance_optimization_default: bandwidth
```

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
```

6.2.1 The dwsd Configuration File

The DataWarp scheduler daemon (`dwsd`) runs on the SDB node and reads the configuration file `/etc/opt/cray/dws/dwsd.yaml` at startup and when it receives the SIGHUP signal. Keep in mind that the majority of the configuration options are considered advanced settings; see *Modify DWS Advanced Settings* on page 96.

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

CONFIGURATION OPTIONS

The configuration file `/etc/opt/cray/dws/dwsd.yaml` contains the following modifiable options:

`activation_cache_default`

Specifies the default for client-side caching on activation objects for activations that support the feature. This includes scratch and cache but excludes swap. For now, this only applies to stripe access types. With client-side caching enabled, application performance can be greatly improved but applications must take special care to avoid having application processes on separate client nodes write to the same page.

`activation_cache_default`: no

`activation_private_cache_default`

Specifies the default for client-side caching on activation objects of private access type configurations. With stripe access type, multiple processes on separate nodes can write to the same page, which can lead to what appears to be data corruption. With private access type, this is not possible since only one client node can ever access a particular file. Consequently the default for client-side caching for private access type is separately configurable.

`activation_private_cache_default`: yes

`cache_limit_action`

What action to take when one of the cache limits are exceeded. This action applies to all limits.

0x1: log only
0x2: error on filesystem operations
0x3: log and error

`cache_limit_action`: 0x3

`cache_max_file_size_default`

Specifies the maximum size (bytes) of any one file that may ever 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 will be emitted to the system console and an error will be reported back to the filesystem operation that triggered the limit.

A value of 0 means no limit. User requests may override this value.

`cache_max_file_size_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.

`cache_max_file_size_max`: 0

`cache_modified_threshold_default`

For cache configurations, the maximum number of dirty bytes (per file) before the file system begins writeback of the dirty bytes to the backing store/PFS.

`cache_modified_threshold_default`: 268435456

`cache_read_ahead_size_default`

For cache configurations, the number of bytes to be read ahead when read-ahead is enabled.

`cache_read_ahead_size_default`: 8388608

`cache_read_ahead_threshold_default`

For cache configurations, the number of bytes that must be read sequentially before a read-ahead starts.

`cache_read_ahead_threshold_default`: 25165824

`cache_stripe_size`

The stripe size for cache configurations. This must be a power of 2 (enforced) as well as a multiple of the PFS stripe size (unenforced). Most PFS configuration stripe sizes are a factor of the default value given here; therefore, it is unlikely this needs to change.

`cache_stripe_size`: 8388608

`cache_substripe_width`

The number of substripes for each cache stripe. Substriping improves performance when multiple client nodes try to interact with the same stripe on a server.

`cache_substripe_width`: 12

`cache_sync_on_close_default`

For cache configurations, this controls the behavior of POSIX `close()`. If yes, a file close will not return until all modified data has been flushed to the backing store/PFS.

`cache_sync_on_close_default`: no

`cache_sync_to_backing_store_default`

For cache configurations, this controls the behavior of POSIX `fsync()` and `fdatasync()`. If yes, these operations will not close until all data has been flushed to the backing store/PFS.

`cache_sync_to_backing_store_default`: no

`device_health_interval`

The minimum number of seconds before `dwsd` asks for a system-wide update on the health of all block devices being used in the DWS. A value of `0` means this action is disabled.

`device_health_interval`: 3600

`dwmd_heartbeat_watchdog`

The number of seconds a `dwmd` may neglect to heartbeat back to `dwsd` before `dwsd` considers the `dwmd` node to be offline. Although `dwsd` normally detects that a `dwmd` is down due to node failure or connect failures to the node, there are cases where a `dwmd` may hang or crash that are only detectable by a lack of heartbeats. A value of `0` disables the check.

`dwmd_heartbeat_watchdog`: 1800

`dwsd_host`

The hostname to which `dwmd` should connect when responding to tasks distributed by `dwsd`. On multi-interface hosts running `dwsd`, the primary hostname may not be appropriate for the remote `dwmd` to use. This option serves as an override to the primary hostname returned by `gethostname()`.

`dwsd_host`: result of `gethostname()`

`equalize_fragments`

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`: yes

`equalize_fragments_guarantee`

This option tweaks both `equalize_fragments` behavior and how pool "free" space is calculated for pools. This option is only valid when `equalize_fragments` is set to `yes`. When `equalize_fragments` and this option are set to `yes`, this option prevents the scheduler from creating an instance that is not comprised of fragments of equal size. Additionally, the amount of free capacity reported in pool information may be adjusted downward so as to reflect the maximum size a request may be while respecting the modified `equalize_fragments` behavior.

`equalize_fragments_guarantee`: yes

`expire_grace`

The length of time in seconds to allow an expired resource to linger before `dwsd` automatically requests its destruction.

`expire_grace`: 3600

`instance_optimization_default`

When processing an instance-create request, `dwsd` decides how to select space across all of the servers that are under the control of the DWS. It first restricts selection to server nodes that belong to the pool specified in the instance create request itself and are up and responding. It then carves out space from each server according to some policy. When the instance create request does not specify a policy, the `instance_optimization_default` option is used as the default.

Valid options:

1. `bandwidth` - pick as many server nodes as possible
2. `interference` - pick as few server nodes as possible, and pick nodes that are unused when possible
3. `wear` - pick nodes based on the health of the underlying block devices. Note that the other options only take device health into account to break ties.

`instance_optimization_default`: bandwidth

`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.

`instance_write_window_length_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.

`instance_write_window_length_max`: 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.

`instance_write_window_length_min`: 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.

`instance_write_window_multiplier_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.

`instance_write_window_multiplier_max`: 0

`log_mask`

A mask for the various types of messages logged by `dwsd`. Bits beyond warning (0x4) are subject to change.

0x0000001: Info
0x0000002: Error
0x0000004: Warning
0x0000080: General-purpose debugging
0x0000100: JSON RPCs
0x0000200: Encrypted messages
0x0000400: JSON notification messages
0x0000800: RCA debugging messages (noisy)

`log_mask`: 0x7

`max_failures`

The maximum number of times `dwsd` attempts to transition a resource to its non-destroyed state before the scheduler requires an API client interaction to try again through PATCH on fuse_blown attributes (`dwcli update resources --id` *`id`* `--replace-fuse`).

`max_failures`: 2

`resource_failure_cooldown`

Periodically, performing an action on a resource may fail (e.g., creating an LVM logical volume as part of fragment creation). When this occurs, it can be useful to wait a small period of time to retry the operation, rather than aggressively retrying in a tight loop. This parameter sets the minimum number of seconds before `dwsd` retries an operation.

`resource_failure_cooldown`: 60

`resource_rm_scan_interval`

Typically all nodes are up and responding and resources can be created and destroyed without error. If a node becomes unresponsive, the resource is created or destroyed once it comes back online. Occasionally, a node becomes unresponsive for an extended period of time (e.g., it crashes and is not rebooted for some time) and it is desirable to "forget" about the resource on the node. This parameter specifies the minimum number of seconds between scans where `dwsd` finds all down nodes (using an external data source) and deletes from them any resources that are intended to be destroyed.

`resource_rm_scan_interval`: 180

`scratch_data_subdirs`

The underlying XFS file system has points of serialization that can be overcome by distributing data into multiple subdirectories. This variable influences the number of subdirectories used for scratch file systems.

`scratch_data_subdirs`: 256

`scratch_limit_action`

What action to take when one of the scratch limits are exceeded; this action applies to all limits.

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

`scratch_limit_action`: 0x3

`scratch_metadata_subdirs`

The underlying XFS file system has points of serialization that can be overcome by distributing metadata into multiple subdirectories. This variable influences the number of subdirectories used for scratch file systems.

`scratch_metadata_subdirs`: 32

`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 is emitted to 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.

`scratch_namespace_max_file_size_default`: 0

`scratch_namespace_max_file_size_max`

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

`scratch_namespace_max_file_size_max`: 0

`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 is displayed 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.

`scratch_namespace_max_files_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.

`scratch_namespace_max_files_max`: 0

`scratch_private_striped`

Specifies whether the scratch private access type should behave in a striped manner. That is, every client node that activates the scratch private configuration will still have its own unique namespace, but each namespace will stripe across all servers in a dwfs realm. This gives any one private namespace access to all space in an instance, rather than being restricted to one fragment's worth of space.

`scratch_private_striped`: yes

`scratch_private_substripe_width`

The number of substripes to use for each scratch stripe (private access type). Substriping a stripe improves performance when multiple client nodes try to interact with the same stripe on a server. Because only one client node is intended to interact with a stripe at any one time with private access type, substripes are not generally useful with scratch private.

`scratch_private_substripe_width`: 1

`scratch_stripe_size`

The stripe size for scratch configurations; this must be a power of 2 (enforced) as well as a multiple of the PFS stripe size (unenforced). Most PFS configuration stripe sizes are a factor of the default value here, so it is unlikely this needs to be changed.

`scratch_stripe_size`: 8388608

`scratch_substripe_size`

The substripe size to use for scratch configurations; this must be a power of 2 (enforced). If the value is less than `scratch_stripe_size`, then each complete read/write of a stripe is mapped to two or more substripe files, using round robin across the substripe files as necessary. If the value is equal to `scratch_stripe_size`, then each complete read/write of a stripe is mapped to exactly one substripe file. If the value is greater than `scratch_stripe_size`, then each complete read/write of a stripe is mapped to a subset of one substripe file.

`scratch_substripe_size`: 8388608

`scratch_substripe_width`

The number of substripes for each scratch stripe (stripe access type). Substriping a stripe improves performance when multiple client nodes try to interact with the same stripe on a server.

`scratch_substripe_width`: 12

`trusted_uids`

The dwsd only trusts messages from the listed numeric uids. This should be the UIDs of `dwmd` (`root`) and the API gateway (non-`root`).

`trusted_uids`: [0, *`nginx_uid`*]

6.2.2 The dwmd Configuration File

The DataWarp management daemon (`dwmd`) master process reads the configuration file `/etc/opt/cray/dws/dwmd.yaml` at startup and when it receives the `SIGHUP` signal. Keep in mind that the majority of the configuration options are considered advanced settings; see *Modify DWS Advanced Settings* on page 96.

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

CONFIGURATION OPTIONS

There are four ways to specify `dwmd` configuration settings. They are, in order of precedence:

1. environment variables
2. `dwmd` command line options
3. configuration file changes via the configurator
4. default `dwmd` settings

Not all configuration settings are meant to be modified by a site. The following subset of configuration options found in `dwmd.yaml` represent those options that an experienced DataWarp administrator might modify. To implement changes to the configuration file, send a `SIGHUP` to `dwmd`.

allow_scc

If specified, dwcfs allows the `dwc_set_stripe_configuration` API to be used and will query the PFS for persistent attributes when a file is first opened. If not specified (default), dwcfs does not attempt to use PFS extended attributes, which avoids the associated overhead in those cases where the PFS does not support or is not configured to allow the use of extended attributes. Only available for dwcfs; value must be [`0` | `1`].

`allow_scc`: `0`

`dvs_mnt_opt`

An optional DVS mount option string added to the option string for `mount -t dwfs [-o option[,option]...]`. This option string must be a comma-separated list of valid DVS options.

`dvs_mnt_opt`: `""`

`dwfs_mnt_opt`

An optional DWfs mount option string added to the option string for `mount -t dwfs [-o option[,option]...]`. This option string must be a comma-separated list of valid DWfs options.

`dwfs_mnt_opt`: `""`

`hb_aggr_delay`

The minimum initial delay interval (in seconds) before `dwmd` goes into aggressive heartbeat mode after the first occurrence of detecting a task failure.

`hb_aggr_delay`: `10`

`hb_val`

The heartbeat to `dwsd` interval in seconds.

`hb_val`: `600`

`hb_num_aggr`

The number of aggressive heartbeats to repeat.

`hb_num_aggr`: `5`

`hb_val_aggr`

The aggressive heartbeat interval used when task failures are detected.

For example, when a task fails, `dwmd` changes the heartbeat delay from `hb_val` to `hb_val_aggr` for `hb_num_aggr` times. If `hb_val_aggr=5`, `hb_val=600`, and `hb_val_aggr=30`, `dwmd` sends a heartbeat every 30 seconds for five times before returning to the normal heartbeat (600 seconds), unless another failure occurs and resets the aggressive heartbeat counter.

`hb_val_aggr`: `30`

`max_utilization`

The percentage (1 to 100) of the capacity of the underlying file system that dwcfs will use for application file data. XFS can fail catastrophically if the file system fills up under a heavy load, and this avoids filling the file system with application data. Only available for dwcfs; value is a percentage.

`max_utilization`: `98`

`substripe_type`

Namespace substripe type; value must be [`DEFERRED` | `NONDEFERRED`].

`substripe_type`: `DEFERRED`

`trusted_uids`

A comma-separated list of trusted numeric UIDs; DataWarp only trusts messages from UIDs within this list. This list must include the UIDs of `dwsd` (`root`) and the API gateway.

Note that the CLE installer sets this option, therefore, it does not need to be changed.

`trusted_uids: [0, nginx_uid]`

`vgck_panic`

Flag for panic node when volume group test fails; value must be [`yes` | `no`].

`vgck_panic: yes`

`xfs_mkfs_opt:`

An optional `mkfs.xfs` command option string added to command. This option string must be a space-separated list of valid `mkfs.xfs` options. Note that `-f` is always added by default.

`xfs_mkfs_opt: "-K"`

`xfs_mnt_opt:`

An optional XFS mount option string added to the option string for the `mount [-o option[,option]...]` command to mount fragments. This option string must be a comma-separated list of valid XFS mount options.

`xfs_mnt_opt: "nodiscard"`

6.2.3 The dwrest Configuration File

The DataWarp RESTful gateway daemon (`dwrest`) reads the configuration file `/etc/opt/cray/dws/dwrest.yaml` at startup and when it receives the `SIGHUP` signal. Keep in mind that the majority of the configuration options are considered advanced settings; see *Modify DWS Advanced Settings* on page 96.

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

CONFIGURATION OPTIONS

The configuration file `/etc/opt/cray/dws/dwrest.yaml` contains the following modifiable options:

`admin_mountroot_blacklist`

A comma-separated list of mount paths on which an administrator cannot create activations. This list is to prevent mistakes by an administrator accidentally mounting over `/tmp` as an example.

While an admin may choose to mount to subdirectories within this blacklist, e.g., `/`, depending upon the mount point it may or may not be advisable to do so.

One may choose to change this to allow these directories to be overlaid, however doing so may put DWS or the system into an unknown state.

`admin_mountroot_blacklist: /sys/fs/cgroup, /var/tmp, /dev/shm, /usr/sbin, /usr/bin, /sbin, /bin, /.snapshots, /root, /boot, /usr, /etc, /tmp, /proc, /sys, /dev, /run, /`

`admins`

A comma-separated list of UIDs of MUNGE administration. MUNGE is used to provide user authentication/identification in environments where uids are consistent across machines.

`admins: []`

`user_mountroot_whitelist`

List of allowed mount paths on which a user can create activations. Note, all paths are required to be fully rooted or `dwrest` will not start. The default of an empty list means that users cannot create activations as there is no valid mount path on which to create them.

Additionally, users cannot mount to the whitelist mount itself. That is, if `/one/two` is in the whitelist and a user tries to mount to `/one/two`, it fails. An activation must be a subdirectory of a mount path in the whitelist. For example `/one/two/three`.

`user_mountroot_whitelist: []`

6.2.4 The dwrestgun Configuration File

The `dwrestgun.conf` file, found on login nodes, contains the configuration options for interfacing with Gunicorn. Gunicorn (Green Unicorn) is a Python WSGI HTTP Server for UNIX. For `dwrest`, `nginx` communicates with Gunicorn, which handles the running of multiple instances of `dwrest` in a number of Unix processes. Keep in mind that the majority of the configuration options are considered advanced settings; see *Modify DWS Advanced Settings* on page 96.

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

CONFIGURATION OPTIONS

The configuration file `/etc/opt/cray/dws/dwrestgun.conf` contains the following modifiable options:

`backlog`

Maximum number of pending connections – this refers to the number of clients that can be waiting to be served. Exceeding this number results in the client getting an error when attempting to connect. It should only affect servers under significant load.

`backlog=128`

`graceful_timeout`

Timeout for graceful workers restart – after receiving a restart signal, workers have this much time to finish serving requests. Workers still alive after the timeout (starting from the receipt of the restart signal) are force killed.

`graceful_timeout=30`

`max_requests`

Maximum number of requests a worker will process before restarting – any value greater than zero will limit the number of requests a worker will process before automatically restarting. This is a simple method to help limit the damage of memory leaks. If this is set to zero (the default), the automatic worker restarts are disabled.

```
max_requests=1024
```

`max_requests_jitter`

Maximum jitter to add to the `max_requests` setting – the jitter causes the restart per worker to be randomized by`randint(0, max_requests_jitter)`. This is intended to stagger worker restarts to avoid all workers restarting at the same time.

```
max_requests_jitter=10
```

`timeout`

Workers silent for more than this many seconds are killed and restarted.

```
timeout=600
```

Some `dwrestgun.conf` options are not listed here because they are not meant to be modified. Additionally, there are Gunicorn settings that, if added to `dwrestgun.conf`, would be valid, but their use is discouraged by Cray. For further details on all Gunicorn configuration options, see *http://docs.gunicorn.org/en/19.3/settings.html#config-file*.

6.3 DataWarp Accounting

The CLE Resource Utilization Reporting (RUR) administration tool includes three plugins (`dws`, `dws_server`, `dws_job_server`) that collect DataWarp accounting data. These plugins must be enabled within the RUR framework in order to collect data. The data is written in JSON format to `/var/opt/cray/log/partition-current/messages-`*`date`* on the SMW.

Usage data is collected separately from compute and storage nodes, and is then combined during post processing. For systems using ALPS, RUR is the primary collection mechanism for DataWarp accounting data. Systems with SLURM must use a plugin defined by SchedMD. RUR is called from both job prologue and epilogue scripts, and by ALPS `apsys`. This provides both job and application scope accounting records, with more complete records for job scope. Job scope records do not include compute node data, only server data.

6.3.1 The dws Data Plugin

The `dws` plugin collects the following DataWarp utilization statistics from compute nodes, if presesnt. This usage data is available within the scope of an application, not the scope of a job. The data is written in JSON dictionary format. Additional DataWarp usage data is available through the `dws_server` and `dws_job_server` plugins.

`bytes_read`	Number of bytes read
`bytes_written`	Number of bytes written
`files_created`	Number of files created
`inodes_created`	Number of inodes created, including files, directories and links
`max_read_offset`	Maximum byte offset read
`max_write_offset`	Maximum byte offset written

RUR dws output

This example shows dws data as written to /var/opt/cray/log/*partition*-current/messages-*date* on the SMW:

```
2017-02-03T13:27:32.662733-05:00 c0-1c0s8n0 RUR 16521
p0-20161006t064726 [RUR@34] uid: 12345, apid: 1140449, jobid:
1127.sdb, cmdname: /bin/hostname, plugin:dws {"token": "1127.sdb",
"mountpoint": "/var/opt/cray/dws/mounts/batch/1127.sdb/ss",
"inodes_created": 407, "files_created": 405, "bytes_read":
11207405004, "bytes_written": 6712208222, "max_read_offset":
4096024126, "max_write_offset": 21772241122}
```

6.3.2 The dws_job_server Data Plugin

The dws_job_server plugin collects utilization statistics from DataWarp servers, if present. This usage information is within the scope of a job, not the scope of an application. The data is written in JSON dictionary format. DataWarp server usage information within the scope of an application is available through the dws_server plugin. Compute node usage of DataWarp is available through the dws plugin.

Type scratch File Systems

The following data is collected for type=scratch file systems.

Per realm:

dwtype	Type of DataWarp file system (scratch)
namespace_count	Number of namespaces within the realm
realm_id	Realm ID
server_count	Number of servers in the realm
server_hostname	Server hostname

Per fragment:

bytes_read	Number of bytes read from this fragment
bytes_written	Number of bytes written to this fragment
capacity_used	Amount of file system capacity used
capacity_max	Maximum capacity of fragment
files_created	Number of files created in this realm
fs_capacity	Capacity of file system to which this fragment belongs
max_window_write	Maximum amount of data written during a write window period of time
write_high_water	The largest amount of data written
write_limit	Maximum bytes allowed to be written per fragment
write_moving_avg	The average amount of data written during a write window period of time

Per namespace:	`namespace_id`	Namespace ID
	`bytes_read`	Number of bytes read from this namespace
	`bytes_written`	Number of bytes written to this namespace
	`files_create_threshold`	Maximum number of files allowed to be created within this namespace
	`file_size_limit`	Maximum size (bytes) of one file
	`files_created`	Number of files created within this namespace
	`max_offset_read`	Maximum byte offset read
	`max_offset_written`	Maximum byte offset written
	`num_data_created`	Total number of data files created on all DataWarp servers
	`stage_bytes_read`	Number of staged bytes read
	`stage_bytes_written`	Number of staged bytes written
	`stripe_size`	Size of each stripe (bytes)
	`stripe_width`	Number of stripes in this namespace
	`substripe_size`	Size of each substripe (bytes)
	`substripe_width`	Number of substripes in per stripe

RUR `dws_server` / `dws_job_server` output for `type=scratch` file systems

This example shows data as written to `/var/opt/cray/log/partition-current/messages-date` on the SMW:

```
uid: 12345, apid: 416746, jobid: 21268, cmdname: xio_p,
plugin:dws_server {{"realm": {"server_count": 1, "fragments": [{
{"capacity_used": 128648781824, "fs_capacity": 3296920076288,
"capacity_max": 128648781824, "max_window_write": 86400,
"files_created": 258, "write_high_water": 3407329284614,
"write_moving_avg": 3407329284614, "bytes_read": 3298534883328,
"write_limit": 32985348833280, "bytes_written": 3407329284614,
"server_hostname": "c0-0c1s1n1"], "namespaces": [{ "namespace_id":
9324, "stripe_width": 1, "stripe_size": 8388608, "bytes_read":
3298534883328, "substripe_width": 12, "stage_bytes_read": 0,
"substripe_size": 8388608, "max_offset_read": 1099511627776,
"files_created": 258, "bytes_written": 3407329284614,
"files_create_threshold": 0, "file_size_limit": 0,
"num_data_created": 258, "stage_bytes_written": 0,
"max_offset_written": 1099511627776 }], "realm_id": 3704}}
```

Type `cache` File Systems

The following data is collected for `type=cache` file systems.

Per realm:	`dwtype`	Type of DataWarp file system (cache)

	`pfs_path`	Backing path
	`realm_id`	Realm ID
	`server_count`	Number of servers in the realm
Per fragment:	**`capacity_highwater`**	Maximum number of bytes used in underlying file system
	`fs_capacity`	Capacity of file system to which this fragment belongs
	`max_offset_read`	Maximum byte offset read
	`max_offset_threshold`	Maximum byte offset allowed to be read/written
	`max_offset_written`	Maximum byte offset written
	`pfs_read`	Number of bytes read from the PFS
	`pfs_written`	Number of bytes written to the PFS
	`window_write_bytes`	Number of bytes written during a write window period of time
	`window_write_seconds`	Number of seconds in a write window period of time
	`cache_read`	Number of DataWarp storage bytes read
	`cache_written`	Number of DataWarp storage bytes written

RUR `dws_server` / `dws_job_server` output for `type=cache` file systems

This example shows data as written to `/var/opt/cray/log/partition-current/messages-date` on the SMW:

```
uid: 12345, apid: 416742, jobid: 21266, cmdname: fdfa1,
{plugin:dws_server "realm": {"fragments": [{ "server_hostname":
"c0-0c1s1n2", "window_write_bytes": 82165273, "fs_capacity": 0,
"capacity_highwater": 0, "window_write_seconds": 21600000,
"cache_written": 82165273, "max_offset_read": 183609,
"max_offset_written": 8388608, "pfs_read": 82165273,
"max_offset_threshold": 0, "pfs_written": 0, "cache_read":
6741137 }, { "server_hostname": "c0-0c1s1n1", "window_write_bytes":
101616737, "fs_capacity": 3856795508736, "capacity_highwater":
2657973817344, "window_write_seconds": 21600000, "cache_written":
101616737, "max_offset_read": 95786, "max_offset_written": 8388608,
"pfs_read": 101616737, "max_offset_threshold": 0, "pfs_written": 0,
"cache_read": 6586671 }], "server_count": 2, "realm_id": 1902,
"pfs_path": "/lus/peel" }}}
```

6.3.3 The dws_server Data Plugin

The `dws_server` plugin collects utilization statistics from DataWarp servers, if present. This usage information is within the scope of an application, not the scope of a job. The data is written in JSON dictionary format. DataWarp server usage information within the scope of a job is available through the `dws_job_server` plugin. Compute node usage of DataWarp is available through the `dws` plugin.

Type `scratch` File Systems

The following data is collected for `type=scratch` file systems.

Per realm:

`dwtype`	Type of DataWarp file system (scratch)
`namespace_count`	Number of namespaces within the realm
`realm_id`	Realm ID
`server_count`	Number of servers in the realm
`server_hostname`	Server hostname

Per fragment:

`bytes_read`	Number of bytes read from this fragment
`bytes_written`	Number of bytes written to this fragment
`capacity_used`	Amount of file system capacity used
`capacity_max`	Maximum capacity of fragment
`files_created`	Number of files created in this realm
`fs_capacity`	Capacity of file system to which this fragment belongs
`max_window_write`	Maximum amount of data written during a write window period of time
`write_high_water`	The largest amount of data written
`write_limit`	Maximum bytes allowed to be written per fragment
`write_moving_avg`	The average amount of data written during a write window period of time

Per namespace:

`namespace_id`	Namespace ID
`bytes_read`	Number of bytes read from this namespace
`bytes_written`	Number of bytes written to this namespace
`files_create_threshold`	Maximum number of files allowed to be created within this namespace
`file_size_limit`	Maximum size (bytes) of one file
`files_created`	Number of files created within this namespace
`max_offset_read`	Maximum byte offset read
`max_offset_written`	Maximum byte offset written
`num_data_created`	Total number of data files created on all DataWarp servers
`stage_bytes_read`	Number of staged bytes read
`stage_bytes_written`	Number of staged bytes written
`stripe_size`	Size of each stripe (bytes)
`stripe_width`	Number of stripes in this namespace
`substripe_size`	Size of each substripe (bytes)

`substripe_width`	Number of substripes in per stripe

RUR `dws_server` / `dws_job_server` output for `type=scratch` file systems

This example shows data as written to `/var/opt/cray/log/partition-current/messages-date` on the SMW:

```
uid: 12345, apid: 416746, jobid: 21268, cmdname: xio_p,
plugin:dws_server {{"realm": {"server_count": 1, "fragments": [{
{"capacity_used": 128648781824, "fs_capacity": 3296920076288,
"capacity_max": 128648781824, "max_window_write": 86400,
"files_created": 258, "write_high_water": 3407329284614,
"write_moving_avg": 3407329284614, "bytes_read": 3298534883328,
"write_limit": 32985348833280, "bytes_written": 3407329284614,
"server_hostname": "c0-0c1s1n1"], "namespaces": [{ "namespace_id":
9324, "stripe_width": 1, "stripe_size": 8388608, "bytes_read":
3298534883328, "substripe_width": 12, "stage_bytes_read": 0,
"substripe_size": 8388608, "max_offset_read": 1099511627776,
"files_created": 258, "bytes_written": 3407329284614,
"files_create_threshold": 0, "file_size_limit": 0,
"num_data_created": 258, "stage_bytes_written": 0,
"max_offset_written": 1099511627776 }], "realm_id": 3704}}
```

Type cache File Systems

The following data is collected for `type=cache` file systems.

Per realm:	**`dwtype`**	Type of DataWarp file system (cache)
	`pfs_path`	Backing path
	`realm_id`	Realm ID
	`server_count`	Number of servers in the realm
Per fragment:	**`capacity_highwater`**	Maximum number of bytes used in underlying file system
	`fs_capacity`	Capacity of file system to which this fragment belongs
	`max_offset_read`	Maximum byte offset read
	`max_offset_threshold`	Maximum byte offset allowed to be read/written
	`max_offset_written`	Maximum byte offset written
	`pfs_read`	Number of bytes read from the PFS
	`pfs_written`	Number of bytes written to the PFS
	`window_write_bytes`	Number of bytes written during a write window period of time
	`window_write_seconds`	Number of seconds in a write window period of time
	`cache_read`	Number of DataWarp storage bytes read
	`cache_written`	Number of DataWarp storage bytes written

RUR `dws_server` / `dws_job_server` output for `type=cache` file systems

This example shows data as written to `/var/opt/cray/log/partition-current/messages-date` on the SMW:

```
uid: 12345, apid: 416742, jobid: 21266, cmdname: fdfa1,
{plugin:dws_server "realm": {"fragments": [{ "server_hostname":
"c0-0c1s1n2", "window_write_bytes": 82165273, "fs_capacity": 0,
"capacity_highwater": 0, "window_write_seconds": 21600000,
"cache_written": 82165273, "max_offset_read": 183609,
"max_offset_written": 8388608, "pfs_read": 82165273,
"max_offset_threshold": 0, "pfs_written": 0, "cache_read":
6741137 }, { "server_hostname": "c0-0c1s1n1", "window_write_bytes":
101616737, "fs_capacity": 3856795508736, "capacity_highwater":
2657973817344, "window_write_seconds": 21600000, "cache_written":
101616737, "max_offset_read": 95786, "max_offset_written": 8388608,
"pfs_read": 101616737, "max_offset_threshold": 0, "pfs_written": 0,
"cache_read": 6586671 }], "server_count": 2, "realm_id": 1902,
"pfs_path": "/lus/peel" }}}
```

7 Post-boot Configuration

About this task

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

Procedure

1. (Optional) Over-provision Intel P3608 cards; *Over-provision an Intel P3608 SSD* on page 65.
2. Flash the firmware for all Fusion IO cards; *Update Fusion ioMemory Firmware* on page 68.
3. Initialize all DataWarp-designated SSDs for use with the DataWarp service (DWS); *Initialize an SSD* on page 70.
4. Create a storage pool; *Storage Pool Configuration Guidelines* on page 74.
5. Assign desired nodes with space to a storage pool; *Assign Nodes to a Storage Pool* on page 77.
6. Verify the configuration; *Verify the DataWarp Configuration* on page 78.

7.1 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

IMPORTANT: This procedure is optional and is only valid for Intel P3608 SSDs. The examples provided are based on the 4TB drives, but this procedure also works for the 1.6TB drives.

Over-provisioning is the process of increasing the spare area on an SSD. This provides extra capacity for the SSD controller to move data around without having to re-write multiple blocks of data as the drive fills up. This results in better performance and higher endurance, but with the tradeoff of less capacity for users. Sites can choose to over-provision or not.

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

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

Procedure

1. Log on to an Intel P3608 SSD-endowed node as `root`, then determine the SSD model number.

```
ssd# module load linux-nvme-ctl
ssd# nvme id-ctrl /dev/nvme0 |grep mn
mn      : INTEL SSDPECME040T4Y
```

2. Shut down the DataWarp manager daemon (`dwmd`).

```
ssd# systemctl stop dwmd
```

3. 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, 2017-02-22 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. Clear 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
```

4. Reconfigure the device based on the model number determined in step *1* on page 66 and the corresponding over-provision value from the following table.

Table 2. Over-provision values for supported Intel P3608 models

Model Number	Size (TB)	Over-provision Value (bytes)	HEX
SSDPECME016T4Y	1.6	1250259487	0x4a85721f
SSDPECME040T4	4.0	3125623327	0xba4d3a1f
SSDPECME040T4Y	4.0	3125623327	0xba4d3a1f

```
nid00350# nvme set-feature device -n 1 -f 0XC1 -v op_value
set-feature:193(Unknown), value:00000000
```

TIP: For the remainder of this procedure, the examples assume 4TB SSDs; values will be different for 1.6TB SSDs.

```
nid00350# nvme set-feature /dev/nvme0 -n 1 -f 0XC1 -v 3125623327
set-feature:193(Unknown), value:00000000
nid00350# nvme set-feature /dev/nvme1 -n 1 -f 0XC1 -v 3125623327
set-feature:193(Unknown), value:00000000
nid00350# nvme set-feature /dev/nvme2 -n 1 -f 0XC1 -v 3125623327
set-feature:193(Unknown), value:00000000
nid00350# nvme set-feature /dev/nvme3 -n 1 -f 0XC1 -v 3125623327
set-feature:193(Unknown), value:00000000
```

5. Confirm the change based on the SSD model number and values in *Over-provision values for supported Intel P3608 models* on page 67. Note that 0xba4d3a1f = 3125623327.

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

```
nid00350# nvme get-feature /dev/nvme0 -n 1 -f 0XC1 --sel=0
get-feature:193(Unknown), value:0xba4d3a1f
nid00350# nvme get-feature /dev/nvme1 -n 1 -f 0XC1 --sel=0
get-feature:193(Unknown), value:0xba4d3a1f
nid00350# nvme get-feature /dev/nvme2 -n 1 -f 0XC1 --sel=0
get-feature:193(Unknown), value:0xba4d3a1f
nid00350# nvme get-feature /dev/nvme3 -n 1 -f 0XC1 --sel=0
get-feature:193(Unknown), value:0xba4d3a1f
```

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

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

7. Log in to the Intel P3608 SSD-endowed node as `root`, and confirm that `SIZE` = `1600319143936` bytes 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.

7.2 Update Fusion ioMemory Firmware

Prerequisites

- A Cray XC series system with one or more nodes with Fusion IO SSD hardware
- 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. For ioMemory3/SX300 cards, CLE 6.0.UP02 and beyond requires the use of the SLES12 version of SanDisk/Fusion driver – VSL 4.2.5. This driver requires firmware version 8.9.5.

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.

- After reflashing, the firmware level cannot be reverted to a previous level, and therefore, the drives are no longer usable with pre-CLE6.0.UP02 releases.

For further details, see *Fusion ioMemory™ VSL® 4.2.x 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.x 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.x-date.fff
   ```

4. Update the firmware.

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

   For example:

   ```
   nid00078# fio-update-iodrive /usr/share/fio/firmware/fusion_4.2.x-date.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 old_level to new_level
     Updated CONTROLLER from old_level to new_level
     Updated SMPCTRL from old_level to new_level
     Updated NCE from old_level to new_level

   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.

7.3 Initialize an SSD

Prerequisites

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

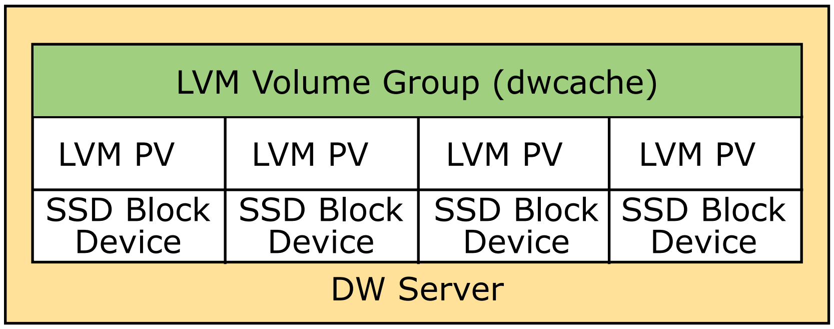
About this task

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_groups` setting of 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 12. 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 130.

Procedure

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

   This example uses `nid00350`.

2. Stop the `dwmd` service.

```
nid00350# systemctl stop dwmd
```

3. Identify the SSD block devices.

   NVMe SSDs:

```
nid00350# lsblk
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
```

   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 loop /var/opt/cray/impsdistribution/squash/
loop1     7:1    0  65536  0 loop /var/opt/cray/impsdistribution/squash/
```

4. (Intel P3608 SSDs only) Proceed to step *6* on page 72 if this SSD was over-provisioned per the *Over-provision an Intel P3608 SSD* on page 65 procedure.

5. (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, 2017-02-22 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
```

6. Initialize each physical device for later use by LVM. Note that Cray currently sells systems with 1, 2, 4, or 6 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 \
/dev/nvme4n1 /dev/nvme5n1
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
```

7. 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
```

8. Start the `dwmd` service.

```
nid00350# systemctl start dwmd
```

9. 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    -  online  fill 8MiB  3.64TiB     0      0
```

FIO SSDs:

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

7.4 Configure and Create a Storage Pool

7.4.1 Storage Pool Configuration Guidelines

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 may 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 during SSD initialization) 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.

Choosing a pool allocation granularity that meets the above requirements will result in a functioning, but perhaps non-optimal, DataWarp environment. 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.

How DataWarp Interacts with Pool Allocation Granularity

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 instances comprised of equal-size fragments only.

> **TIP:** This default behavior is new in CLE 6.0.UP05 as both `dwsd.yaml` configuration settings `equalize_fragments` and `equalize_fragments_guarantee` are set to `yes`. In previous releases the `equalize_fragments` default configuration setting in `dwsd.yaml` was `no`.

The `equalize_fragments_guarantee` setting prevents the scheduler from ever creating an instance that is not comprised of fragments of equal size. By default, it is set to `yes`, but is only applicable when `equalize_fragments` is also set to `yes`. Because `equalize_fragments_guarantee` has implications for how free space is calculated, the amount of free capacity reported in pool information may be adjusted downward so as to reflect the maximum size of a request. For more information, see *Why Does the Free Capacity Displayed by dwstat pools not Match the Quantity Capacity?* on page 41.

When `equalize_fragments` is not enabled, 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.

Recommendations

Taken together, Cray recommends the following:

- For performance reasons, create pools that only contain nodes with homogeneous SSD hardware.
- Keep the `equalize_fragments` and `equalize_fragments_guarantee` options set to `yes` (default).
- Use the smallest possible pool allocation granularity, which is typically 16MiB or 24MiB, depending on system hardware

Example: 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 16MiB
create request for pools entity with name = wlm_pool accepted, "dwstat pools" for
status
```

Verify the pool was created.

```
crayadm@login> dwstat pools
    pool  unit quantity free   gran
wlm_pool bytes        0    0  16MiB
```

7.4.2 Create a Storage Pool

Prerequisites

This procedure assumes that the pool being created will only include nodes with homogeneous SSD hardware. Although not recommended due to performance issues, it is possible to create a storage pool comprised of non-homogeneous SSD nodes; see *Create a Storage Pool Comprised of Non-homogeneous SSD Hardware* on page 85.

About this task

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 during SSD initialization) 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.

For further information, see *Storage Pool Configuration Guidelines* on page 74.

Procedure

1. Identify the list of nodes with homogeneous SSDs to be added to the pool being created.

   Use these commands to determine the type of SSD hardware available on the DataWarp nodes:

   - `dwstat nodes` - displays a table of current DataWarp nodes

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

    node     pool online drain  gran capacity insts activs
nid00001        -  online  fill 16MiB  5.82TiB     0      0
nid00002        -  online  fill 16MiB  5.82TiB     0      0
nid00425        -  online  fill  8MiB  5.82TiB     0      0
nid00428        -  online  fill  8MiB  2.91TiB     0      0
```

   - `xtssdconfig` - displays SSD configuration information

```
   node_id                model_number     ssd_id  ...    size           serial_num
--------------------------------------------------------------------------------------
 c0-0c0s0n1        INTEL SSDPECME040T4Y 0x80860953  ...  4000GB CVF86242006Y4P0DGN-1
 c0-0c0s0n1        INTEL SSDPECME040T4Y 0x80860953  ...  4000GB CVF86242006Y4P0DGN-2
 c0-0c0s0n1        INTEL SSDPECME040T4Y 0x80860953  ...  4000GB CVF8632200354P0DGN-1
 c0-0c0s0n1        INTEL SSDPECME040T4Y 0x80860953  ...  4000GB CVF8632200354P0DGN-2
 c0-0c0s0n2        INTEL SSDPECME040T4Y 0x80860953  ...  4000GB CVF86242003N4P0DGN-1
 c0-0c0s0n2        INTEL SSDPECME040T4Y 0x80860953  ...  4000GB CVF86242003N4P0DGN-2
 c0-0c0s0n2        INTEL SSDPECME040T4Y 0x80860953  ...  4000GB CVF8624200724P0DGN-1
 c0-0c0s0n2        INTEL SSDPECME040T4Y 0x80860953  ...  4000GB CVF8624200724P0DGN-2
c0-1c0s10n1 SAMSUNG MZPKK3T2HAJL-00003 0x144da821  ...  3200GB       S2SDNAAGA00008
c0-1c0s10n1 SAMSUNG MZPKK3T2HAJL-00003 0x144da821  ...  3200GB       S2SDNAAGA00019
c0-1c0s11n0        INTEL SSDPECME040T4Y 0x80860953  ...  4000GB CVF86322000R4P0DGN-1
c0-1c0s11n0        INTEL SSDPECME040T4Y 0x80860953  ...  4000GB CVF86322000R4P0DGN-2
```

- `rtr -Im` - displays nid/cname mapping

```
crayadm@smw> rtr -Im p0
NID    NIC-Addr   Node          Aries            Blue  Black  Green
----   --------   ----------    ------------     ----  -----  -----
0         0       c0-0c0s0n0    c0-0c0s0a0        0      0      0
1         1       c0-0c0s0n1    c0-0c0s0a0        0      0      0
2         2       c0-0c0s0n2    c0-0c0s0a0        0      0      0
...
425       553     c0-1c0s10n1   c0-1c0s10a0       1      0     10
...
428       556     c0-1c0s11n0   c0-1c0s11a0       1      0     11
...
```

In this example, nodes 1, 2, and 428 have Intel SSDs, and node 425 has a Samsung SSD.

2. Select a node from the group of nodes chosen for the pool and extract its node allocation value from `dwstat nodes` (`gran` column).

For example, if creating a pool that will include nid00001, nid00002, and nid00428, select one of the nodes, say nid00348. Its node allocation is 8MiB.

3. Determine the pool's allocation granularity by picking the first multiple of the node granularity found in step *2* on page 77 that is greater than or equal to 16MiB.

In this example, the node allocation of nid00001 is 8MiB. The first multiple of the 8MiB that is greater than or equal to 16MiB, is 16MiB. Therefore, the pool's allocation granularity will be defined as 16MiB.

4. Log on to a service node and create a storage pool with allocation granularity = 16MiB.

```
crayadm@login> module load dws
crayadm@login> dwcli create pool --name wlm_pool --granularity 16MiB
create request for pools entity with name = wlm_pool accepted, "dwstat pools"
for status
```

5. Verify the pool was created.

```
crayadm@login> dwstat pools
    pool  unit quantity free  gran
wlm_pool bytes        0    0 16MiB
```

Next: assign a node to the pool.

7.5 Assign Nodes 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
- A node can only be associated with a storage pool if the node's granularity is a factor of the pool's allocation granularity, see *Storage Pool Configuration Guidelines* on page 74

About this task

Follow this procedure to associate SSD-endowed nodes 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 SSD-endowed nodes with a storage pool.

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

   Note that the `update` action has an extended parsing capability that allows multiple inputs for `--name`. The syntax allows for whitespace-delimited inputs, comma-separated inputs, `pdsh`-style range inputs, or any combination of these. See the `dwcli(8)` man page for complete details.

   For example, to associate nodes `nid00001`, `nid00002`, and `nid00428` with a storage pool called `wlm_pool`:

   ```
   crayadm@login> dwcli update node --name nid0000[1-2] nid00428 --pool wlm_pool
   update request for nodes entity with name = nid00001 accepted, "dwstat nodes" for status
   update request for nodes entity with name = nid00002 accepted, "dwstat nodes" for status
   update request for nodes entity with name = nid00428 accepted, "dwstat nodes" for status
   ```

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

4. Verify the nodes are associated with the pool.

   ```
   crayadm@login> dwstat nodes
       pool units quantity     free  gran
   wlm_pool bytes 25.48TiB 14.55TiB 16MiB

       node     pool online drain  gran capacity insts activs
   nid00001 wlm_pool online  fill 16MiB  5.82TiB     0      0
   nid00002 wlm_pool online  fill 16MiB  5.82TiB     0      0
   nid00425        -  online  fill  8MiB  5.82TiB     0      0
   nid00428 wlm_pool online  fill  8MiB  2.91TiB     0      0
   ```

7.6 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 as `crayadm`, then 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 online  fill 16MiB     1TiB     1      0
nid00066 wlm_pool online  fill 16MiB     1TiB     0      0
nid00069 wlm_pool online  fill 16MiB     1TiB     0      0
nid00070 wlm_pool online  fill 16MiB     1TiB     0      0
nid00004        - online  fill 16MiB  3.64TiB     0      0
nid00005        - online  fill 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 Nodes to a Storage Pool* on page 77.
   - 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.

8 DataWarp Administrator Tasks

8.1 Check the Status of DataWarp Resources

Prerequisites

The `dws` module must be loaded:

```
> module load dws
```

The `dwstat` command

To check the status of various DataWarp resources, invoke the `dwstat` command, which has the following format:

```
dwstat [-h] [unit_options] [RESOURCE [RESOURCE]...]
```

Where:

unit_options

Includes a number of options that determine the SI or IEC units with which output is displayed. See the `dwstat(1)` man page for details.

RESOURCE

May be: `activations`, `all`, `configurations`, `fragments`, `instances`, `most`, `namespaces`, `nodes`, `pools`, `registrations`, or `sessions`.

By default, `dwstat` displays the status of pools:

```
> dwstat
    pool units quantity    free   gran
wlm_pool bytes        0       0   1GiB
 scratch bytes  7.12TiB 2.88TiB 128GiB
```

In contrast, `dwstat all` reports on all resources for which it finds data:

```
> dwstat all
    pool units quantity     free   gran
wlm_pool bytes 14.38TiB 13.88TiB 128GiB

  sess state token     creator owner             created expiration nodes
     4 CA---  1527 MOAB-TORQUE  1226 2017-05-19T21:16:12      never     0
     7 CA---  1534 MOAB-TORQUE  1226 2017-05-19T23:53:17      never     0
   138 CA---  1757 MOAB-TORQUE   827 2017-05-29T14:46:09      never     0
   139 CA---  1759 MOAB-TORQUE 10633 2017-05-29T16:06:26      never    32
  inst state sess  bytes nodes             created expiration intact  label  public confs
     4 CA---    4 128GiB     1 2017-05-19T21:16:12      never intact   I4-0 private     1
     7 CA---    7 128GiB     1 2017-05-19T23:53:17      never intact   I7-0 private     1
   138 CA---  138 128GiB     1 2017-05-29T14:46:09      never intact I138-0 private     1
   139 CA---  139 128GiB     1 2017-05-29T16:06:26      never intact I139-0 private     1

  conf state inst    type activs
     4 CA---    4 scratch      0
     7 CA---    7 scratch      0
   138 CA---  138 scratch      0
   139 CA---  139 scratch      0

  reg state sess conf wait
    4 CA---    4    4 wait
    7 CA---    7    7 wait
  137 CA---  139  139 wait

  frag state  nst capacity     node
    10  CA--    4   128GiB nid00350
    15  CA--    7   128GiB nid00350
   180  CA--  138   128GiB nid00350
   181  CA--  139   128GiB nid00350

   ns state conf frag span
    4  CA--    4   10    1
    7  CA--    7   15    1
  138  CA--  138  180    1
  139  CA--  139  181    1

    node     pool online drain  gran capacity insts activs
nid00322 wlm_pool online  fill  8MiB  5.82TiB     0      0
nid00349 wlm_pool online  fill  4MiB  1.46TiB     0      0
nid00350 wlm_pool online  fill 16MiB  7.28TiB     4      0

did not find any cache configurations, swap configurations, activations
```

For further information, see the `dwstat(1)` man page.

8.2 Check SSD Health and Remaining Life

The `xtcheckssd` command queries the health of one or more SSDs (both FusionIO and NVMe), including remaining life expectancy. The command is located in `/opt/cray/ssd/bin`, and must be run as `root` on an SSD service node or from a login node. `xtcheckssd` runs as a daemon or as a one-time command. It reports output to:

- the console (when not run as a daemon)
- the SMW, via the `/dev/console` log
- the CLE system log (`syslog`) via the RCA event `ec_rca_diag`

Examples

1. Report on all SSDs for a node:

```
nid00065# /opt/cray/ssd/bin/xtcheckssd
PhysLoc:IBB:Slot2,Name:INTEL SSDPECME040T4Y,SN:CVF86242003T4P0DGN-1,Size:
2000GB,Remaining life:100%,Temperature:31(c)
PhysLoc:IBB:Slot2,Name:INTEL SSDPECME040T4Y,SN:CVF86242003T4P0DGN-2,Size:
```

```
2000GB,Remaining life:100%,Temperature:31(c)
PhysLoc:IBB:Slot3,Name:INTEL SSDPECME040T4Y,SN:CVF86242005T4P0DGN-1,Size:
2000GB,Remaining life:100%,Temperature:29(c)
PhysLoc:IBB:Slot3,Name:INTEL SSDPECME040T4Y,SN:CVF86242005T4P0DGN-2,Size:
2000GB,Remaining life:100%,Temperature:29(c)
xtcheckssd-terminate. exit code: 0 Normal program termination
```

2. Run `xtcheckssd` as a deamon generating a report every 24 hours.

```
nid00433# /opt/cray/ssd/bin/xtcheckssd -d
```

3. Run `xtcheckssd` for multiple nodes, originating from a login node.

```
login# module load pdsh
login# pdsh -w nid00073,nid00421 /opt/cray/ssd/bin/xtcheckssd
nid00073: PhysLoc:IBB:Slot2,Name:INTEL
SSDPECME040T4,SN:CVF85156006N4P0DGN-1,Size:2000GB,Remaining life:
100%,Temperature:25(c)
nid00073: PhysLoc:IBB:Slot2,Name:INTEL
SSDPECME040T4,SN:CVF85156006N4P0DGN-2,Size:2000GB,Remaining life:
100%,Temperature:26(c)
nid00073: PhysLoc:IBB:Slot3,Name:INTEL
SSDPECME040T4,SN:CVF85153001J4P0DGN-1,Size:2000GB,Remaining life:
100%,Temperature:24(c)
nid00073: PhysLoc:IBB:Slot3,Name:INTEL
SSDPECME040T4,SN:CVF85153001J4P0DGN-2,Size:2000GB,Remaining life:
100%,Temperature:25(c)
nid00421: PhysLoc:IODC0:J9500,Name:ioMemory PX600-2600,SN:1410G0497,Size:
2600GB,Remaining life:96%,Temperature:29(c)
nid00421: PhysLoc:IODC0:J9500,Name:ioMemory PX600-2600,SN:1410G0369,Size:
2600GB,Remaining life:100%,Temperature:31(c)
nid00421: xtcheckssd-terminate. exit code: 0 Normal program termination
```

See the `pdsh(1)` and `xtcheckssd(8)` man pages for further details.

8.3 Remove Nodes From a Storage Pool

Prerequisites

- DataWarp administrator privileges

About this task

Nodes that no longer exists or are no longer DataWarp server nodes should be removed from the pool to which they are 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   6.0TiB  6.0TiB  128GiB

    node     pool online drain  gran capacity insts activs
nid00065 wlm_pool online  fill 16MiB     1TiB     0      0
nid00066 wlm_pool online  fill 16MiB     1TiB     0      0
nid00067 wlm_pool online  fill 16MiB     1TiB     0      0
nid00068 wlm_pool online  fill 16MiB     1TiB     0      0
nid00069 wlm_pool online  fill 16MiB     1TiB     0      0
nid00070 wlm_pool online  fill 16MiB     1TiB     0      0
nid00004        - online  fill 16MiB  3.64TiB     0      0
nid00005        - online  fill 16MiB  3.64TiB     0      0
```

3. Remove the desired nodes from its pool.

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

Note that the `update` action has an extended parsing capability that allows multiple inputs for `--name`. The syntax allows for whitespace-delimited inputs, comma-separated inputs, `pdsh`-style range inputs, or any combination of these. See the `dwcli(8)` man page for complete details.

For example, remove nodes `nid00068`, `nid00069`, and `nid00070`.

```
crayadm@login> dwcli update node --name nid000[68-70] --rm-pool
update request for nodes entity with name = nid00068 accepted, "dwstat nodes"
for status
update request for nodes entity with name = nid00069 accepted, "dwstat nodes"
for status
update request for nodes entity with name = nid00070 accepted, "dwstat nodes"
for status
```

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

4. Verify the change.

```
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 online  fill 16MiB     1TiB     0      0
nid00066 wlm_pool online  fill 16MiB     1TiB     0      0
nid00067 wlm_pool online  fill 16MiB     1TiB     0      0
nid00004        - online  fill 16MiB  3.64TiB     0      0
nid00005        - online  fill 16MiB  3.64TiB     0      0
```

8.4 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 online  fill 16MiB  3.64TiB     0      0
   nid00065 wlm_pool online  fill 16MiB     1TiB     0      0
   nid00066 wlm_pool online  fill 16MiB     1TiB     0      0
   nid00069 wlm_pool online  fill 16MiB     1TiB     0      0
   nid00070 wlm_pool online  fill 16MiB     1TiB     0      0
   nid00005        - online  fill 16MiB  3.64TiB     0      0
   ```

3. Reassign the desired node(s) to the desired pool.

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

   Note that the `update` action has an extended parsing capability that allows multiple inputs for `--name`. The syntax allows for whitespace-delimited inputs, comma-separated inputs, `pdsh`-style range inputs, or any combination of these. See the `dwcli(8)` man page for complete details.

   To reassign node `nid00069` and `nid00070` to pool `pvt_pool`:

   ```
   crayadm@login> dwcli update node --name nid000[69-70] --pool pvt_pool
   update request for nodes entity with name = nid00069 accepted, "dwstat nodes"
   for status
   update request for nodes entity with name = nid00070 accepted, "dwstat nodes"
   for status
   ```

   Node `nid00069` and `nid00070` are assigned to the pool: `pvt_pool`, resulting in increased storage capacity for `pvt_pool` and decreased capacity for `wlm_pool`.

4. Verify the change.

   ```
   crayadm@login> dwstat pools nodes
       pool units quantity    free    gran
   pvt_pool bytes  5.64TiB 5.64TiB   16GiB
   wlm_pool bytes   2.0TiB  2.0TiB  128GiB

       node     pool online drain  gran capacity insts activs
   nid00004 pvt_pool online  fill 16MiB  3.64TiB     0      0
   nid00069 pvt_pool online  fill 16MiB     1TiB     0      0
   nid00070 pvt_pool online  fill 16MiB     1TiB     0      0
   nid00065 wlm_pool online  fill 16MiB     1TiB     0      0
   ```

```
nid00066 wlm_pool online  fill 16MiB     1TiB     0       0
nid00005        - online  fill 16MiB  3.64TiB     0       0
```

8.5 Create a Storage Pool Comprised of Non-homogeneous SSD Hardware

Prerequisites

Task prerequisites.

About this task

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 during SSD initialization) 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.

For further information, see *Storage Pool Configuration Guidelines* on page 74.

For performance reasons, Cray recommends that storage pools only contain nodes with homogeneous SSD hardware. However, in certain facilities this may not be feasible. This procedure describes how to create a storage pool such that its allocation granularity is compatible with different SSD hardware types. Note that it is unnecessary to identify the different types of SSD hardware .

Procedure

1. View the DataWarp nodes and write down unique values of the `gran` column for nodes that will be added to the same pool.

   ```
   crayadm@login> module load dws
   crayadm@login> dwstat nodes
       node     pool online drain  gran capacity insts activs
   nid00001        - online  fill 16MiB  5.82TiB     0       0
   nid00422        - online  fill  8MiB  2.91TiB     0       0
   nid00446        - online  fill 12MiB  2.18TiB     0       0
   ```

   For example, assume the pool to be created will contain all three nodes. The unique node granularities are 16MiB, 8MiB, and 12MiB.

2. Determine the least common multiple (LCM) of the chosen node allocation granularities.

   Use *http://www.wolframalpha.com/* to assist with calculating the LCM for the selected node allocation granularities. In the search box, enter

   ```
   lcm gran1 gran2 ...
   ```

   and press `enter`.

   Using the example above, the LCM of granularities 8, 12, and 16 is 48.

3. Determine the minimum pool allocation granularity based on the LCM as follows:

   - If the calculated LCM is greater than or equal to 16MiB, then the LCM value is the minimum pool allocation granularity. Proceed to step *4* on page 86.
   - If the calculated LCM is less than 16MiB, the value must be adjusted upward to determine the minimum pool allocation granularity. Choose the first multiple of the LCM that is greater than or equal to 16MiB.

     For example, if the calculated LCM is 12MiB, the first multiple of 12 is 24, 24 is greater than 16MiB, therefore 24 is the minimum pool allocation granularity.

4. Select the pool allocation granularity.

   Cray recommends using the minimum pool allocation granularity (determined in step *3* on page 86).

   - For this example, the recommended pool allocation granularity is 48MiB.

   Other valid pool allocation granularities are the minimum pool allocation granularity plus any multiple of the calculated LCM.

   - Example 1: If the calculated LCM is 48MiB and the minimum pool allocation granularity is 48MiB, valid values are 48MiB, 96MiB, 144MiB, etc.
   - Example 2: If the calculated LCM is 12MiB and the minimum pool allocation granularity is 24MiB, valid values 24Mib, 36Mib, 48Mib, etc.

5. Create a storage pool with allocation granularity = 48MiB.

   ```
   crayadm@login> dwcli create pool --name wlm_pool --granularity 48MiB
   create request for pools entity with name = wlm_pool accepted, "dwstat pools"
   for status
   ```

6. Verify the pool was created.

   ```
   crayadm@login> dwstat pools
       pool  unit quantity free  gran
   wlm_pool bytes        0    0 48MiB
   ```

Next: assign a node to the pool.

8.6 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
2017-05-20 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']
2017-05-20 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
2017-05-20 12:39:41 (11964): <76> [os-172-30-196-191]: Traceback (most recent call last):
2017-05-20 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
2017-05-20 12:39:41 (11964): <76> [os-172-30-196-191]:     % (dwfs_id, realm_id))
2017-05-20 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
2017-05-20 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']
2017-05-20 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
2017-05-20 12:39:41 (11964): <76> [os-172-30-196-191]: Traceback (most recent call last):
2017-05-20 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
2017-05-20 12:39:41 (11964): <76> [os-172-30-196-191]:     % (dwfs_id, realm_id))
2017-05-20 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.

8.7 Drain Storage Nodes

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@login> module load dws
crayadm@login> dwstat nodes pools
    node     pool online drain gran capacity insts activs
nid00022 wlm_pool online  fill 8MiB  3.64TiB     0      0
nid00023 wlm_pool online  fill 8MiB  3.64TiB     0      0
nid00024 wlm_pool online  fill 8MiB  3.64TiB     0      0
nid00025 wlm_pool online  fill 8MiB  3.64TiB     0      0

    pool units  quantity     free   gran
wlm_pool bytes  14.56TiB 14.56TiB 128GiB
```

2. Drain one or more storage nodes.

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

Note that the `update` action has an extended parsing capability that allows multiple inputs for `--name`. The syntax allows for whitespace-delimited inputs, comma-separated inputs, `pdsh`-style range inputs, or any combination of these. See the `dwcli(8)` man page for complete details.

For example, to drain nodes `nid00024` and `nid00025`:

```
crayadm@login> dwcli update node --name nid00024,nid00025 --drain
update request for nodes entity with name = nid00024 accepted, "dwstat nodes"
for status
update request for nodes entity with name = nid00025 accepted, "dwstat nodes"
for status
crayadm@login> dwstat nodes pools
     node    pool online drain gran capacity insts activs
nid00022 wlm_pool online  fill 8MiB  3.64TiB     0      0
nid00023 wlm_pool online  fill 8MiB  3.64TiB     0      0
nid00024 wlm_pool online drain 8MiB  3.64TiB     0      0
nid00025 wlm_pool online drain 8MiB  3.64TiB     0      0

    pool units quantity    free   gran
wlm_pool bytes  7.28TiB 7.28TiB 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@login> dwstat fragments
frag state inst  capacity     node
 102  CA--    47 745.19GiB nid00024
```

4. (Optional) Remove that instance.

```
crayadm@login> dwcli rm instance --id 47
rm request for instance entity with id = 47 accepted, "dwstat instances" for
status
```

   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.

   For example, unset the drain for `nid00024` only.

```
crayadm@login> dwcli update node --name nid00024 --fill
update request for nodes entity with name = nid00024 accepted, "dwstat nodes"
for status
crayadm@login> dwstat nodes pools
     node    pool online drain gran capacity insts activs
nid00022 wlm_pool online  fill 8MiB  3.64TiB     0      0
nid00023 wlm_pool online  fill 8MiB  3.64TiB     0      0
nid00024 wlm_pool online  fill 8MiB  3.64TiB     0      0
nid00025 wlm_pool online drain 8MiB  3.64TiB     0      0

    pool units quantity     free   gran
wlm_pool bytes 10.92TiB 10.92TiB 128GiB
```

8.8 Do Not Quiesce a DataWarp Node

WARNING: DVS, which provides software infrastructure for DataWarp storage, includes a quiesce capability that enables administrators to temporarily suspend traffic to a DVS-projected file system, allowing them to safely repair a file system or take a DVS server node out of service. This capability **must never be used** on DataWarp service nodes because of the risk for data corruption. Instead, use the *Drain Storage Nodes* on page 87 procedure to prevent new instance activity on a node.

8.9 Examples Using dwcli

The `dwcli` command provides a command line interface to act upon DataWarp resources. This is primarily an administration command, although a user can initiate some actions using it. With full WLM support, a user does not have a need for this command. For complete details, see the `dwcli(8)` man page.

The `dws` module must be loaded to execute DataWarp commands.

```
> module load dws
```

EXAMPLE: Create a pool

Only an administrator can execute this command.

```
# dwcli create pool --name wlm-pool --granularity 16MiB
create request for pools entity with name = wlm-pool accepted, "dwstat pools" for
status
```

EXAMPLE: Assign nodes to the pool

Only an administrator can execute this command.

```
login# dwcli update node --name nid000[30-32] --pool wlm-pool
update request for nodes entity with name = nid00030 accepted, "dwstat nodes" for
status
update request for nodes entity with name = nid00031 accepted, "dwstat nodes" for
status
update request for nodes entity with name = nid00032 accepted, "dwstat nodes" for
status
```

EXAMPLE: Create a scratch session, instance, configuration, and activation

Only an administrator can create a session or instance.

```
login# dwcli create session --expiration 1462815522 --creator crayadm --token \
ok-scratch --owner 12345 --hosts $(dwcli ls nodes)
125
login# dwcli create instance --expiration 1462815522 --public --session 125 \
--pool wlm_pool --capacity 1099511627776 --label my-scratch
324
login# dwcli create configuration --type scratch --access-mode stripe \
--root-permissions 0755 --instance 324 --group 513
112
login# dwcli create activation --mount /mnt/dw/my_mount --caching \
--configuration 112 --session 125
231
```

EXAMPLE: Create a swap session, instance, configuration, and activation

Only an administrator can create a session or instance.

```
login# dwcli create session --expiration 1462815522 --creator crayadm --token \
ok-swap --owner 12345 --hosts nid00030
126
login# dwcli create instance --expiration 1462815522 --public --session 126 \
--pool wlm_pool --capacity 1099511627776 --label my-swap
325
login# dwcli create configuration --type swap --swap-size 4GiB --instance 325 \
--group 513
113
login# dwcli create activation --configuration 113 --session 126
232
```

EXAMPLE: Create a cache session, instance, configuration, and activation

Only a DataWarp administrator can create a session or instance.

```
login# dwcli create session --expiration 1462815522 --creator crayadm --token \
ok-cache --owner 12345 --hosts nid00030
127
login# dwcli create instance --expiration 1462815522 --public --session 127 \
--pool wlm_pool --capacity 1099511627776 --label my-cache
326
login# dwcli create configuration --type cache --access-mode stripe \
--backing-path /lus/scratch --instance 326
114
login# dwcli create activation --mount /mnt/dw/my_mount2 --configuration 114 \
--session 127
323
```

EXAMPLE: View results from above create commands

```
login# dwstat most
$dwstat most
    pool units quantity f   ree   gran
wlm_pool bytes  8.59TiB 2.15TiB 200GiB

sess state      token creator owner             created expiration nodes
 125 CA--- ok-scratch crayadm 12345 2017-05-18T11:48:45   12:38:42     1
 126 CA--- ok-swap    crayadm 12345 2017-05-18T11:48:45   12:38:43     1
 127 CA--- ok-cache   crayadm 12345 2017-05-18T11:48:46   12:38:43     1

inst state sess  bytes nodes  created            expiration intact      label public confs
 324 CA---  125 200GiB     1 11:48:46   2017-05-18T12:38:43 intact my-scratch public     1
 325 CA---  126 200GiB     1 11:48:46   2017-05-18T12:38:43 intact    my-swap public     1
 326 CA---  127 200GiB     1 11:48:48   2017-05-18T12:38:44 intact   my-cache public     1

conf state inst    type activs
 112 CA---  324 scratch      1
 113 CA---  325    swap      1
 114 CA---  326   cache      1

reg state sess conf wait
110 CA---  125  113 wait
111 CA---  126  114 wait
112 CA---  127  115 wait

activ state sess conf nodes cache             mount
  321 CA---  125  112     1   yes  /mnt/dw/my_mount
  322 CA---  126  113     1    no                 -
  333 CA---  127  114     1    no /mnt/dw/my_mount2
```

EXAMPLE: Set multiple registrations to --haste

Directs DWS to not wait for associated configurations to finish asynchronous activities such as waiting for all staged out data to finish staging out to the PFS.

```
login# dwcli update registration --id 1,4-6 --haste
update request for registrations entity with id = 1 accepted, "dwstat
registrations" for status
update request for registrations entity with id = 4 accepted, "dwstat
registrations" for status
update request for registrations entity with id = 5 accepted, "dwstat
registrations" for status
update request for registrations entity with id = 6 accepted, "dwstat
registrations" for status
```

EXAMPLE: Remove a pool

Only an administrator can execute this command.

```
login# dwstat pools
  pool units quantity    free  gran
canary bytes  3.98GiB 3.97GiB 16MiB
# dwcli rm pool --name canary
# dwstat pools
no pools
```

EXAMPLE: Remove multiple sessions

Only an administrator or the session owner can execute this command.

```
login# dwstat sessions
sess state token creator owner             created expiration nodes
   1 D----    ok    test 12345 2017-05-18T16:31:24    expired     1
   2 D----    ok    test 12345 2017-05-18T16:31:24    expired     1
   3 D----    ok    test 12345 2017-05-18T16:31:24    expired     1
$ dwcli rm session --id 1-3
rm request for sessions entity with id = 1 accepted, "dwstat sessions" for status
rm request for sessions entity with id = 2 accepted, "dwstat sessions" for status
rm request for sessions entity with id = 3 accepted, "dwstat sessions" for status
```

EXAMPLE: Fuse replacement

Only an administrator or the session owner can execute this command.

```
login# dwstat instances
inst state sess bytes nodes             created expiration intact           label public confs
   1 D-F-M    1 16MiB     1 2017-05-18T17:47:57    expired  false canary-instance public     1
login# dwcli update instance --replace-fuse --id 1
update request for instances entity with id = 1 accepted, "dwstat instances" for status
login# dwstat instances
inst state sess bytes nodes             created expiration intact           label public confs
   1 D---M    1 16MiB     1 2017-05-18T17:47:57    expired  false canary-instance public     1
```

EXAMPLE: Stage in a directory, query immediately, then stage list

```
user@login> dwcli stage in --session $session --configuration $configuration --dir=/tld/. \
--backing-path=/tmp/demo/
path   backing-path nss ->c ->q ->f <-c <-q <-f <-m
/tld/. -              1   3   1   -   -   -   -   -

user@login> dwcli stage query --session $session --configuration $configuration
path  backing-path nss ->c ->q ->f <-c <-q <-f <-m
/.    -              1   4   -   -   -   -   -   -
/tld/ -              1   4   -   -   -   -   -   -

user@login> dwcli stage list --session $session --configuration $configuration
path                   backing-path                nss ->c ->q ->f <-c <-q <-f <-m
/tld/filea             /tmp/demo/filea               1   1   -   -   -   -   -   -
/tld/fileb             /tmp/demo/fileb               1   1   -   -   -   -   -   -
/tld/subdir/subdirfile /tmp/demo/subdir/subdirfile   1   1   -   -   -   -   -   -
/tld/subdir/subfile    /tmp/demo/subdir/subfile      1   1   -   -   -   -   -   -
```

EXAMPLE: Stage a file in afterwards, stage list, then query

Note the difference in the `stage query` output.

```
user@login> dwcli stage in --session $session --configuration $configuration --file /dwfsfile \
--backing-path /tmp/demo/filea
path      backing-path    nss ->c ->q ->f <-c <-q <-f <-m
```

```
/dwfsfile /tmp/demo/filea   1   1   -   -   -   -   -   -

user@login> dwcli stage list --session $session --configuration $configuration
path                   backing-path                nss ->c ->q ->f <-c <-q <-f <-m
/dwfsfile              /tmp/demo/filea               1   1   -   -   -   -   -   -
/tld/filea             /tmp/demo/filea               1   1   -   -   -   -   -   -
/tld/fileb             /tmp/demo/fileb               1   1   -   -   -   -   -   -
/tld/subdir/subdirfile /tmp/demo/subdir/subdirfile   1   1   -   -   -   -   -   -
/tld/subdir/subfile    /tmp/demo/subdir/subfile      1   1   -   -   -   -   -   -


user@login> dwcli stage query --session $session --configuration $configuration
path      backing-path    nss ->c ->q ->f <-c <-q <-f <-m
/.        -                 1   5   -   -   -   -   -   -
/tld/     -                 1   4   -   -   -   -   -   -
/dwfsfile /tmp/demo/filea   1   1   -   -   -   -   -   -
```

EXAMPLE: Terminate stage operations

To terminate a directory stage:

```
user@login> dwcli stage terminate -c 4 -s 4 -d /testdir/.
path       backing-path nss ->c ->q ->f <-c <-q <-f <-m
/testdir/. -              1   -   -   -   -   -   -   -

user@login> dwcli stage terminate -c 4 -s 4 -d /.
path backing-path nss ->c ->q ->f <-c <-q <-f <-m
/.   -              1   -   -   -   -   -   -   -
```

And similarly to terminate a file stage:

```
user@login> dwcli stage terminate -c 5 -s 5 -f /testdir/file
request accepted
user@login> dwcli stage terminate -c 5 -s 5 -f /testdir/file
Unexpected error codes -22 found in dwmd reply: {u'-22': [4]}
```

Where the "unexpected error" message indicates the file no longer exists.

EXAMPLE: Back up and restore the DataWarp configuration

Only a DataWarp administrator can execute these `dwcli` commands. For this example, the following configuration is assumed:

```
login# dwstat nodes pools
         node   pool online drain gran capacity insts activs
datawarp12-s5 backup online  fill 8MiB  3.99GiB     0      0

  pool units quantity    free  gran
backup bytes  3.98GiB 3.98GiB 16MiB
  demo bytes        0       0 16MiB
```

1. Backup the configuration.

   ```
   login# dwcli config backup > backup.json
   ```

2. Remove the `demo` pool.

   ```
   login# dwcli rm pool --name demo
   ```

3. Restore the configuration as captured by the backup.

```
login# dwcli config restore < backup.json
note: recreating pool(s) 'demo'
note: creating pool demo with granularity=16MiB and units=bytes
pool add progress [===========] 1/1 100% done
pool chk progress [===========] 1/1 100% done
node chk progress [===========] 1/1 100% done
```

4. Verify the configuration.

```
login# dwstat nodes pools
         node   pool online drain gran capacity insts activs
datawarp12-s5 backup online  fill 8MiB  3.99GiB     0      0

  pool units quantity    free  gran
backup bytes  3.98GiB 3.98GiB 16MiB
  demo bytes        0       0 16MiB
```

8.10 Manage Access to DataWarp Nodes

Each workload manager (WLM) has its own commands for managing DataWarp, and it is beyond the scope of this administration guide to document all cases. Therefore, WLM examples are provided with the caveat that they may be out of sync with changes made by the WLM vendors, although it is Cray's intent to keep them up-to-date. For detailed information, refer to the documentation for the site-specific WLM.

Access to DataWarp SSD nodes is managed through the various WLMs. In Slurm, for example, access to DataWarp nodes is controlled by either the `AllowUsers` or `DenyUsers` option within the burst buffer configuration file `/opt/slurm/default/etc/burst_buffer.conf`. These options are mutually exclusive. By default, all users have access to the DataWarp nodes.

```
boot# xtopview
default/:/# vi /opt/slurm/default/etc/burst_buffer.conf
```

The format of either option is a colon-delimited list of usernames or UIDs. For example:

```
DenyUsers=username1:username2...
```

8.11 Flash NVMe SSD Firmware

Prerequisites

- A Cray XC series system with one or more SSD cards installed
- Ability to log in as `root`
- Access to an image flash file appropriate for the specific type of SSD (location provided by Cray)

About this task

This procedure, typically only done at Cray's recommendation, ensures that the firmware of any SSD cards is up-to-date with an image flash file. The `xtiossdflash` command compares the current flash version to the image flash file and flashes the device (up or down) only if the two are different. For further information, see the `xtiossdflash(8)` man page.

```
boot# module load ssd-flash
boot# man xtiossdflash
```

Procedure

1. Log on to the boot node as `root`, then load the `pdsh` and `ssd-flash` modules.

   ```
   smw:# ssh boot
   boot:# module load pdsh ssd-flash
   ```

2. Copy the firmware image to all target nodes to be flashed.

   ```
   boot:# scp ssd_fw_image_file target:/location
   ```

   Where:

   ssd_fw_image_file

   Specifies the SSD flash image

   target

   Specifies a single node with SSDs

   location

   Specifies the location on the node to place the image

   For example:

   ```
   boot:# scp /P3608_FW/FW1B0_BL133 c0-1c0s9n2:/tmp
   ```

3. Flash the firmware.

   ```
   boot:# xtiossdflash -f -i /location/ssd_fw_image_file target
   ```

   Where:

   /location/ssd_fw_image_file

   Specifies the path (on the target node) to the SSD flash image

   target

   Specifies a single node with SSDs, a comma-separated list of nodes with SSDs, or the keyword `all_service`

   For example:

   ```
   boot:# xtiossdflash -f -i /tmp/FW1B0_BL133 c0-1c0s9n2
   ```

   If the firmware updates successfully, the message `Successfully flashed` is displayed.

   If the firmware does not update successfully, one of the following messages is displayed.

   - `No devices available` - No `/dev/nvmeX` devices were found.
   - `The file /root/8DV101B0_8B1B0133_signed.bin does not exist. Skipping.` - The firmware image flash file could not be found.
   - `/dev/nvmeX already flashed to firmware version <version>. Skipping.` - The device is already flashed to the firmware selected.

- `Failure to download firmware to /dev/nvmeX` - Firmware could not be loaded to an SSD.
- `Invalid firmware slot` - The specified slot on the device is not valid.
- `Invalid firmware image` - Specified firmware is not valid for the SSD type.

If applicable, rectify the problem and try again.

4. Load the new firmware:

- For Intel SSDs:

  Reboot the target node(s) to load the new firmware.

  ```
  boot# xtbootsys --reboot target
  ```

  For example:

  ```
  boot# xtbootsys --reboot c0-1c0s9n2
  ```

- For Samsung SSDs:

  1. Power cycle the blade on which the target node is located.

  2. Reboot the node.

  ```
  boot# xtbootsys --reboot target
  ```

  For example:

  ```
  boot# xtbootsys --reboot c0-1c0s10n1
  ```

5. Verify the flash version.

The firmware version displayed is for example purposes only and should not be expected.

```
boot# /opt/cray/ssd-flash/bin/xtiossdflash -v c0-1c0s9n2
c0-1c0s9n2: <nvme_flash>: /dev/nvme3: Model = INTEL SSDPECME040T4Y , FW Version = 8DV101B0
c0-1c0s9n2: <nvme_flash>: /dev/nvme2: Model = INTEL SSDPECME040T4Y , FW Version = 8DV101B0
c0-1c0s9n2: <nvme_flash>: /dev/nvme1: Model = INTEL SSDPECME040T4Y , FW Version = 8DV101B0
c0-1c0s9n2: <nvme_flash>: /dev/nvme0: Model = INTEL SSDPECME040T4Y , FW Version = 8DV101B0
```

8.12 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 46 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.

2. Proceed based on the setting to be modified:
   - To modify a displayed setting, proceed to step *3* on page 97.
   - To modify a non-displayed setting, proceed to step *4* on page 98.
   - To reset a displayed setting to its default value, proceed to step *5* on page 98.

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 99.

4. Modify a non-displayed setting:

This example modifies 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: no
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: no
...
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 99.

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: no
...
```

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, type: cle)

  cray_dws        [ status: enabled ]  [ validation: valid ]

----------------------------------------------------------------------------------------------------------
  Selected      #      Settings                             Value/Status (level=advanced)
----------------------------------------------------------------------------------------------------------
                       service
               1)        managed_nodes_groups               ['datawarp_nodes']
               2)        api_gateway_nodes_groups           ['login_nodes']
               3)        external_api_gateway_hostnames     (none)
               4)        dwrest_cacheroot_whitelist         /lus/scratch
               5)        dwrest_cachemount_whitelist        (none)
               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
```

8.13 Configure SSD Protection Settings

Prerequisites

- Ability to log in as `root`
- Read *Modify DWS Advanced Settings* on page 96

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 (S-2558)*.

> **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 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 96.

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
```

8.14 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.

8.15 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`*

```
2017-05-13 15:51:07 State file is at v0.0
2017-05-13 15:51:07 ADMIN ALERT -> This version of dwsd expects state file v1.0, you have v0.0.
2017-05-13 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.
2017-05-13 15:51:07 src/dwsd/context.c:dwsd_context_init():356 -> Unable to initialize sqlite
2017-05-13 15:51:07 Daemon ran for 2 seconds
2017-05-13 15:51:07 src/dwsd/dwsd.c:main():66 -> Context initialization failed.
2017-05-13 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# systemctl stop dwsd
```

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# systemctl start dwsd
```

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
```

8.16 In the Event of DataWarp Database Corruption

Prerequisites

- DataWarp administrator privileges

About this task

The DataWarp scheduler daemon, `dwsd`, relies on a state database for accurate node and pool information. Although a rare event, it is possible for this file to become corrupt. If this happens, administrator intervention is required.

Database Corruption Symptoms

A corrupt `dwsd` state database will generate varying error messages, dependent on the type of corruption, in `sdb:/var/opt/cray/dws/log/dwsd.log`. The most common is `SQLite error: database disk image is malformed`, as shown in this excerpt:

```
2017-05-24 08:50:20 Attempting to listen on port 2015
2017-05-24 08:50:20 src/dwsd/log.c:sqlite_errorlog_cb():168 -> SQLite error 11:
database corruption at line 52076 of [ea3317a480]
2017-05-24 08:50:20 src/dwsd/log.c:sqlite_errorlog_cb():168 -> SQLite error 11:
database corruption at line 52115 of [ea3317a480]
2017-05-24 08:50:20 src/dwsd/log.c:sqlite_errorlog_cb():168 -> SQLite error 11:
statement aborts at 5: []
2017-05-24 08:50:20 src/dwsd/log.c:sqlite_errorlog_cb():168 -> SQLite error 11:
database disk image is malformed
```

```
2017-05-24 08:50:20 src/dwsd/sqlite.c:sqlite_init():131 -> SQLite error:
database disk image is malformed
2017-05-24 08:50:20 src/dwsd/context.c:dwsd_context_init():356 -> Unable to
initialize sqlite
2017-05-24 08:50:20 Daemon ran for 0 seconds
2017-05-24 08:50:20 src/dwsd/dwsd.c:main():66 -> Context initialization failed.
2017-05-24 08:50:20 Shutting down
```

Additionally, commands such as `dwstat` may fail:

```
sdb# module load dws
sdb dwstat
cannot communicate with dwsd daemon at sdb port 2015
[Errno 111] Connection refused
```

Corrective Steps

If a current backup of the `dwsd` state database exists, it is possible to recover some of the configuration data. Otherwise, with no backup, it is necessary to re-create pools and the node-pool associations. In either case, **no user data is recoverable**. Follow this procedure after determining that the DataWarp database is corrupt.

Cray recommends backing up the state file at regular intervals, see *Back Up and Restore DataWarp State Data* on page 103.

Procedure

1. Stop the `dwsd` service.

   ```
   sdb# service dwsd stop
   ```

2. Remove the DataWarp database file.

   ```
   sdb# rm /var/opt/cray/dws/dwsd.db
   ```

3. Start the `dwsd` service.

   ```
   sdb# service dwsd start
   ```

4. 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.

5. Proceed based on the availability of a `dwsd` state file backup:

   - If a current backup exists, either via `cron` or done manually, restore the DataWarp state file from the backup:

     ```
     sdb# dwcli config restore < path_to_backup_file
     ```

   - If a current backup does not exist, recreate the pools and the node-pool associations.

     ```
     sdb# dwcli create pool -n pool_name -g granularity
     sdb# dwcli update node -n hostname -p pool_name
     ```

8.17 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
   ```

   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
Service Configuration Menu (Config Set: p0, 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/worksheets
INFO - Changelog will be written to
    - /var/opt/cray/imps/config/sets/p0/changelog/
changelog_2016-04-08T17:19:18.yaml
INFO - Running post-configuration scripts
INFO - Locally cloning ConfigSet 'p0' to 'p0-autosave-2016-04-08T17:19:29'.
INFO - Successfully cloned to ConfigSet 'p0-autosave-2016-04-08T17:19:29'.
INFO - Removed ConfigSet 'p0-autosave-2016-04-06T17:09:08'.
INFO - ConfigSet 'p0' has been updated.
INFO - Run 'cfgset search -s cray_node_health --level advanced p0' to review
the current settings.
```

6. Verify the settings.

```
smw# cfgset search -s cray_node_health -l advanced -t DataWarp p0
...
# 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.

8.18 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
   ```

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 112 through *6* on page 112 for all SSD nodes listed within the node group(s) defined as `managed_nodes_groups`.

DataWarp is deconfigured.

8.19 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
update request for nodes entity with name = nid00350 accepted, "dwstat nodes"
for status
```

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 114.

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

    node     pool online drain  gran capacity insts activs
nid00322 wlm_pool online  fill 16MiB  7.28TiB     0      0
nid00349 wlm_pool online  fill 16MiB  7.28TiB     0      0
nid00350 wlm_pool online  fill 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 online  fill 16MiB  7.28TiB     0      0
nid00349 wlm_pool online  fill 16MiB  7.28TiB     0      0
nid00350 wlm_pool online  fill 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
rm request for instances entity with id = 2227 accepted, "dwstat instances"
for status
```

   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 `--haste`. For further information, see *Registrations* on page 42.

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, 2017-02-22 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/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
```

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
     update request for nodes entity with name = nid00349 accepted, "dwstat
     nodes" for status
     ```

   - 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 115.

8.20 Complete the Replacement of an SSD Node

Prerequisites

- DataWarp administrator privileges
- Completion of *Prepare to Replace a DataWarp SSD* on page 113

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. (Optional) Over-provision the new SSD if it is an Intel P3608; see *Over-provision an Intel P3608 SSD* on page 65.

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 70.

5. Set `--fill` for the new SSD node.

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

   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 --fill
   update request for nodes entity with name = nid00350 accepted, "dwstat nodes"
   for status
   crayadm@sdb> dwstat nodes
       node     pool online drain  gran capacity insts activs
   nid00322 wlm_pool online   fill 16MiB  7.28TiB     0       0
   nid00349 wlm_pool online   fill 16MiB  7.28TiB     0       0
   nid00350 wlm_pool online   fill 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 not drain.

     ```
     dwcli update node --name nid00349 --fill
     update request for nodes entity with name = nid00349 accepted, "dwstat
     nodes" for status
     ```

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

   This completes the process of replacing a DataWarp SSD node.

8.21 The `dwpoolhelp` Command

Beginning with CLE 6.0.UP05 `equalize_fragments` is enabled by default; therefore, `dwpoolhelp` is deprecated and will be removed in a future release.

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.

When `equalize_fragments` is not enabled, 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.

TIP:

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. If `equalize_fragments` is enabled, there is no need to use `dwpoolhelp`.

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. The following example shows `dwpoolhelp` output sorted by amount of waste per node. See the dwpoolhelp(8) man page for further details.

Example: 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:

1. Determine node capacity and allocation granularity.

```
crayadm@login> dwstat -b nodes
    node      pool online drain    gran       capacity insts activs
nid00028         - online  fill 8388608 4000795590656     0      0
nid00029         - online  fill 8388608 4000795590656     0      0
nid00089         - online  fill 8388608 4000795590656     0      0
nid00090         - online  fill 8388608 4000795590656     0      0
```

2. 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) = 1024
Reducing granules per node to 1023 to account for interaction with node granularity
Maximum 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
        ...
```

3. Sort on 'Waste per pool,' as there is too much output to weed through.

```
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
        ...
```

4. Create a pool with 185 granularities per node.

```
crayadm@login> dwcli create pool --name wlm_pool2 --granularity 21625831424
create request for pools entity with name = wlm-pool2 accepted, "dwstat pools"
for status
```

5. 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
```

9 Troubleshooting

9.1 Where are the Log Files?

The DataWarp scheduler daemon (`dwsd`), manager daemon (`dwmd`), and RESTful service (`dwrest`) write to local log files, the console log, and 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.

9.2 What Does this Log Message Mean?

DataWarp daemons and the utilities they invoke write log messages for many different reasons, and not all are of interest or concern. Additionally, a message can occur during a transient condition and not be interesting, but become interesting during certain non-transient conditions. It is important to keep this in mind when browsing through a log file.

9.2.1 Low SSD Life Remaining

When a DataWarp SSD reaches 90% of its life expectancy, a message is written to the console log file.

```
Mon 03/17/2017 3:17 PM
SEC: 15:17 sitename-systemname: Low SSD Life Remaining 8% c3-0c0s2n1 PCIe slot -1
```

9.2.2 SSD Protection Limits Exceeded

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. See *Configure SSD Protection Settings* on page 100 for further details.

If the SSD protection settings are configured to log SSD overuse events (default setting), then a message is written to the console log file when an SSD protection threshold is exceeded. The message varies slightly depending on the configuration type (scratch or cache) as well as the type of violation, but always includes the following information:

`sid`	Session ID
`stoken`	Session token When a WLM is controlling DataWarp, the session token is the batch job identifier.
`rid`	Registration ID, or range of IDs (when same session/stoken registration gets converted into a single label)

Examples: SSD threshold messages written to the console log file

For a scratch configuration, when the bytes written threshold is exceeded:

```
[Thu Apr 20 22:13:13 2017] kdwfs: KDWFS protection limit(s) exceeded!
[Thu Apr 20 22:13:13 2017] kdwfs: Write threshold (16777216) exceeded. label=sid:
1;stoken:"WLM.111" rid:1;sid:1;stoken:"WLM.111"
```

For a scratch configuration, when the file size threshold is exceeded:

```
[Thu Apr 20 22:14:44 2017] kdwfs: KDWFS protection limit(s) exceeded!
[Thu Apr 20 22:14:44 2017] kdwfs: File size threshold (1) exceeded. label=sid:
4;stoken:"WLM.234" rid:7;sid:4;stoken:"WLM.234"
```

For a scratch configuration, when the file creation threshold is exceeded:

```
[Thu Apr 20 22:15:05 2017] kdwfs: KDWFS protection limit(s) exceeded!
[Thu Apr 20 22:15:05 2017] kdwfs: File creation threshold (1) exceeded.
label=sid:12;stoken:"WLM.432" rid:4;sid:12;stoken:"WLM.432"
```

For a persistent scratch instance created by one user and used by two others, when the bytes written threshold is exceeded:

```
[Thu Apr 20 22:17:57 2017] kdwfs: KDWFS protection limit(s) exceeded!
[Thu Apr 20 22:17:57 2017] kdwfs: Write threshold (34359738368) exceeded.
label=sid:28;stoken:"WLM.557" rid:100-110;sid:87:stoken:"WLM.732" rid:133;sid:
92:stoken:"WLM.759"
```

For a cache configuration when the bytes written threshold is exceeded:

```
[Thu Apr 20 22:23:27 2017] kdwcfs: KDWCFS protection limit(s) exceeded!
[Thu Apr 20 22:23:17 2017] kdwcfs: Write threshold (16777216) exceeded.
label=sid:57;stoken:"WLM.18482" rid:22-31;sid:57;stoken:"WLM.18482"
```

For a cache configuration when the file size threshold is exceeded:

```
[Thu Apr 20 22:23:27 2017] kdwcfs: KDWCFS protection limit(s) exceeded!
[Thu Apr 20 22:23:17 2017] kdwcfs: File size threshold (1) exceeded. label=sid:
28;stoken:"WLM.26636" rid:2-11;sid:28;stoken:"WLM.26636"
```

9.2.3 dwmd Daemon Triggers a Crash

The `dwmd` daemon triggers a server node panic and writes a message to the console log when it detects a faulty LVM logical volume, which indicates the likelihood of bad hardware.

```
2017-04-04T15:01:53.663992-06:00 c0-1c0s1n0 DataWarp dwmd daemon triggering a
crash after detecting a failed LVM volume group.  Check for failing hardware!
```

Fix: Be suspicious of the hardware on the mentioned node. If an SSD comes online with no issues after reboot, then it is possible that a problem was incorrectly detected. If it does not come online, then it must be re-seated and/or replaced. Continue to monitor the node for some time to see if a pattern emerges. Perhaps the device is faulty in specific situations, e.g., after a heavy load. Contact Cray support for further information.

9.2.4 MUNGE Authentication Error

Scenario: Execution of `dwstat` failed with the error: `You must be authenticated to request this resource`. At the same time, the MUNGE authentication service wrote the following message to a DataWarp log file:

```
MUNGE decrypt error: Rewound credential
```

DWS components depend on MUNGE authentication, which requires client/server clock times to be in sync (within a range). This message may indicate that the time discrepancy between the client and server node exceeded the threshold.

Fix: Sync client/server clock times. There is no need to restart any services.

9.3 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 5/22/2017 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 May 22 15:17:05 2017  --  Time when SEC observed the logged string.

 Entire line in log file:
 /var/opt/cray/log/p0-20150817t070336/console-20150817
 -----
 2017-05-22T15:17:04.871808-05:00 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 XC™ Series SEC Configuration Guide.

9.4 Why Do `dwcli` and `dwstat` Fail?

The `dwcli` and `dwstat` commands fail for a variety of reasons, some of which are described here.

1. Both commands fail if the DataWarp service is not configured or not up and running.

   ```
   user@login> dwstat
   Cannot determine gateway via libdws_thin
   fatal: Cannot find a valid api host to connect to or no config file found.
   ```

   Fix: Ensure that the DataWarp service is up and running.

2. Both commands fail if the `dws` module is not loaded.

   ```
   user@login> dwstat
   If 'dwstat' is not a typo you can use command-not-found to lookup the package
   that contains it, like this:
   cnf dwstat
   ```

   Fix: load the module and try again.

   ```
   user@login> module load dws
   > dwstat
          pool units quantity      free   gran
      wlm_pool bytes 53.12TiB 16.74TiB   1GiB
   ```

3. Both commands fail if the DataWarp scheduler daemon goes offline.

```
user@login> dwstat
cannot communicate with dwsd daemon at sdb-hostname port 2015
[Errno 111] Connection refused
```

TIP:

One reason the scheduler daemon `dwsd` may go offline is if DataWarp state files are upgraded such that the DWS is not backwards compatible with any state file from the previous release. This should only be a concern immediately following an upgrade.

Fix: A backup of the most recent state file prior to upgrade must be restored to the upgraded format. For details, see *Back Up and Restore DataWarp State Data* on page 103.

4. Both commands fail when SSL certificate verification fails.

```
user@login> dwstat all
Connecting to https://c1-0c0s0n2:81 yielded fatal error:
[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:581)
```

TIP: One reason SSL certification may fail is if SMW HA was recently installed on a system already running DataWarp. The installation creates a new certificate chain, thereby invalidating any client certificates that were generated by the prior non-HA installation. To remedy, the host certificates must be re-created. See *XC™ Series SMW HA Installation Guide (S-0044)*.

5. Both commands fail if the DataWarp configuration option `allow_dws_cli_from_computes` is set to `false` and one of the following is true:

- the command is executed from a batch script
- the command is executed from a compute node

Both commands output an error message similar to the following:

```
Connecting to https://dwrest-nodename yielded fatal error:
[SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:581)
```

Fix: to have this functionality, the system administrator must change the configuration setting and restart DataWarp.

6. Both commands fail when there is a MUNGE authentication issue.

```
user@login> dwstat
You must be authenticated to request this resource.
```

Fix: DWS components depend on MUNGE authentication, which requires client/server clock times to be in sync (within 120 seconds). This message may indicate that the time discrepancy between the client and server node exceeded the threshold. Check client/server clock times, and sync if necessary. There is no need to restart any services.

7. Depending on the options and actions invoked, `dwcli` can fail when `dwmd` is not functional.

```
user@login> dwcli stage in -c 1 -s 1 --backing-path /etc/lvm/ --dir /test
cannot communicate with backend dwmd daemon at datawarp port 49214
[Errno 111] Connection refused
```

Fix: If the DataWarp service is up and running, attempt to start `dwmd`.

```
smw# systemctl start dwmd
```

9.5 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
2017-05-17 15:24:05 ========== Event on fd 4
2017-05-17 15:24:05 Caught signal User defined signal 1
2017-05-17 15:24:05 Alerting the task manager to wake up
2017-05-17 15:24:05 ========== Event on fd 7
2017-05-17 15:24:05 Finding tasks to spawn
2017-05-17 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.

9.6 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 96 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).

9.7 Staging Failure Might be Caused by Insufficient Space

A `#DW stage_in` or `#DW stage_out` job script request can fail if there is insufficient space to complete the request. If a user reports a hung job or that a stage in/out request has failed, it could be due to insufficient space to fulfill the request. If this occurs, the job goes into a hold state and an error message is written to the `dwmd` log file on the SMW.

DataWarp Stage Out Failure

At the end of a batch job, the DWS transitions any files that are marked for deferred stage out to actually staging out. If there is insufficient space in the PFS to accommodate the data, the `#DW stage_out` command fails. Output from `dwcli stage query` will be similar to the following:

```
> dwcli stage query -s $sid -c $cid -f /demo
 path  backing-path nss ->c ->q ->f <-c <-q <-f <-m
/demo     /tmp/demo   1   -   -   -   -   -   1   -
```

Additionally, a message similar to:

```
__udwfs_activate_deferred_stage failed: -28
```

will be written to `smw:/var/opt/cray/log/$PARTITION-current/dws/dwmd-$DATE`.

If the DataWarp session, configuration, and instance do not expire, then the job and all of its DataWarp resources will remain until the system fails or an administrator intervenes.

When sufficient space is available, the stage out request can be submitted manually. Monitor the PFS to validate that it has enough space to complete the request. Cray also recommends validating that enough inodes are available, although this is much less likely to occur.

```
> df /pfs_mount; df -hi /pfs_mount
```

When sufficient space is available, manually resubmit the staging request via `dwcli stage out`. For example:

```
> dwcli stage out --configuration $configuration --session $session \
--backing-path /pfs/path --dir /dwfs/dir
```

After the data is staged out, either the user or an administrator must remove the WLM job following the WLM-specific procedures.

DataWarp Stage In Failure

A `#DW stage_in` request will fail if the DataWarp instance requested is not large enough for the amount of data being transferred. If this happens output from `dwcli stage query` will be similar to the following:

```
> dwcli stage query -s $sid -c $cid -f /demo
 path  backing-path nss ->c ->q ->f <-c <-q <-f <-m
/demo      /tmp/demo   1   -   -   1   -   -   -   -
```

and a message similar to:

```
 __udwfs_activate_deferred_stage failed: -28
```

will be written to `sdb:/var/opt/cray/dws/log/dwsd.log`.

The user or an administrator must remove the WLM job following the WLM-specific procedures, after which the user can resubmit the job with a DataWarp allocation large enough to cover the requirements of files needing to be staged in.

9.8 Memory Swapping Caveats

When swapping is enabled, CLE uses disk space to augment RAM. This may allow programs that would otherwise not fit within RAM, or even those that would fit, to run more efficiently by allowing CLE to keep the most frequently used memory contents in RAM. Swapping is used for memory that is not backed by a file, for example, heap allocations, program data and bss segments, or anonymous memory mapped by `mmap()`.

However, not all programs reap the same benefits when swapping is enabled. Swapping primarily benefits applications that do not use the communication runtime. Some operations and performance optimizations prevent memory from being swapped. For example, I/O operations usually prevent the memory that contains the I/O buffer from being swapped while the I/O is being performed. This includes file I/O and network I/O. When the I/O operation is complete, the memory is generally free to be swapped.

Additionally, hugepages cannot be swapped. Hugepages are used to improve the performance of network codes. They are automatically used by the programming environment runtime for some data structures described in the table. They are also used for application memory when one of the Cray hugepages modules is loaded (see the `intro_hugepages(1)` man page).

Many parallel programming models also prevent some amount of memory from being swapped. The restrictions frequently stem from performance optimizations and the use of network I/O. The table *Swapping restrictions by parallel programming model* on page 128 summarizes the programming models and their operations, using default settings, that prevent swapping.

Table 3. Swapping restrictions by parallel programming model

Parallel Programming Model	Operations that Restrict Swapping
MPI	● Internal MPI buffers used for eager messages and small message mailboxes. Allocated during `MPI_Init()` or program execution and released during `MPI_Finalize()`. ● User buffers for large off-node transfers. During the transfer, and while in-use by the MPI buffer cache. ● User buffers for large on-node transfers. During the transfer, and while in-use by the MPI buffer cache.
DMAPP and SHMEM	● Program data, bss and symmetric heap: during the entire program execution ● Buffers for transfers: during the transfer
PGAS	Program data, bss, often the stack, private and shared heap: during the entire program execution

9.9 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.

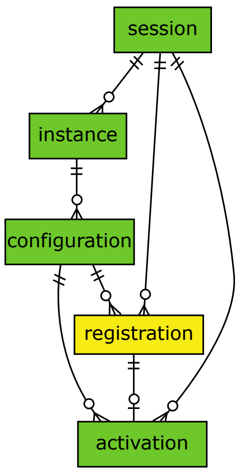
```
sdb# systemctl restart dwsd
```

10 Supplemental Information

10.1 Terminology

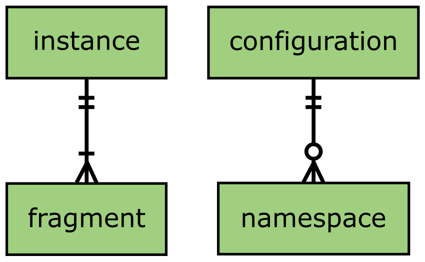
The following diagram shows the relationship between the majority of the DataWarp service terminology using Crow's foot notation. A session can have 0 or more instances, and an instance must belong to only one session. An instance can have 0 or more configurations, but a configuration must belong to 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 13. 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 14. 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 directory 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.

10.2 Prefixes for Binary and Decimal Multiples

The International System of Units (SI) prefixes and symbols (e.g., **k**ilo-, **M**ega-, **G**iga-) are often used interchangeably (and incorrectly) for decimal and binary values. This misuse not only causes confusion and errors, but the errors compound as the numbers increase. In terms of storage, this can cause significant problems. For example, consider that a kilobyte (10^3) of data is only 24 bytes less than 2^{10} bytes of data. Although this difference may be of little consequence, the table below demonstrates how the differences increase and become significant.

To alleviate the confusion, the International Electrotechnical Commission (IEC) adopted a standard of prefixes for binary multiples for use in information technology. The table below compares the SI and IEC prefixes, symbols, and values.

SI decimal vs IEC binary prefixes for multiples					
SI decimal standard			IEC binary standard		
Prefix (Symbol)	Power	Value	Value	Power	Prefix (Symbol)
kilo- (kB)	10^3	1000	1024	2^{10}	kibi- (KiB)
mega- (MB)	10^6	1000000	1048576	2^{20}	mebi- (MiB)
giga- (GB)	10^9	1000000000	1073741824	2^{30}	gibi- (GiB)
tera- (TB)	10^{12}	1000000000000	1099511627776	2^{40}	tebi- (TiB)
peta- (PB)	10^{15}	1000000000000000	1125899906842624	2^{50}	pebi- (PiB)
exa- (EB)	10^{18}	1000000000000000000	1152921504606846976	2^{60}	exbi- (EiB)
zetta- (ZB)	10^{21}	1000000000000000000000	1180591620717411303424	2^{70}	zebi- (ZiB)
yotta- (YB)	10^{24}	1000000000000000000000000	1208925819614629174706176	2^{80}	yobi- (YiB)

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