Making the Jump to Light Speed with Cray's DataWarp

An Administrator's Perspective

Tina Declerck, Dave Paul
NERSC
Lawrence Berkeley National Laboratory
Berkeley, CA USA
tmdeclerck@lbl.gov, dpaul@lbl.gov

***Abstract*—Cori, the first phase of NERSC's next generation supercomputer, has 144 DataWarp nodes available to it's users. Cray's DataWarp technology provides an intermediate storage capability, sitting between on node memory and the parallel file system. It utilizes Cray's DVS to provide access from compute nodes on a request basis. In order for this to work, the workload manager interacts with Cray's DataWarp API's to create the requested file system and make it available on the nodes requested by the job (or jobs). Some of the tools needed by an administrator are therefore included in the workload manager, SLURM at our site, and other information requires use of tools and commands provided by Cray. It is important to know what information is available, where to find it, and how to get it.**

Keywords-DataWarp

I. INTRODUCTION

At the National Energy Research Scientific Computing (NERSC) center, storage is an area that continues to grow – as is true at most sites. The amount of data increases with the increase in ability to process more data. Although some codes do a lot of processing on the same data; even those require some data to process and will generally also produce data that needs to be stored. In order to keep a system balanced, therefore, users need a faster way to get data from disk storage to on-node memory for processing and then move processed data back out to disk. Without this, processors sit idle much of the time waiting for I/O. Tiered storage is one of the solutions that can provide better access to data for applications. To address this problem NERSC was looking for a burst buffer. Cray's solution is DataWarp. It provides a solid state disk (SSD) that is integrated with the Cray XC-40 systems using service nodes. However, this solution was still under development. NERSC entered into a non-recurring engineering (NRE) contract to work with Cray on developing key DataWarp features for our workload and also to ensure the administrative interface provided the information required for both debugging and to allow our work load manager to interact with it. Many sites are primarily looking for a way to provide better checkpoint / restart solutions which Cray's DataWarp addresses but our users have additional use cases we wanted to ensure were also addressed. In addition, some of the key features NERSC was looking for are administrative interfaces to allow better control and ability to diagnose and resolve problems. The goal of this paper is to provide some lessons learned from our experiences and guidelines for understanding and using Cray's DataWarp solution.

Administrators are very aware that many tools, although useful and improve our users' ability to accomplish their scientific goals, are yet another thing that will need to be monitored, verified, and repaired. Cray's DataWarp definitely fits in that category. NERSC has been working with Cray for about a year and a half developing the DataWarp features and have gained quite a lot of experience in that time finding problems, diagnosing them, and working with Cray on finding good solutions, and then testing those to ensure they work. During this process, Cray has been quickly providing solutions and patches for the issues we've found to allow us to continue to work with our users. This has been somewhat hampered by the major software upgrade with CLE 6.0 / SMW 8.0 since solutions for some issues were fixed in the update but not on CLE 5.2 UP04 / SMW 7.0 that NERSC is currently using. This has required Cray to work on two different solutions, which slowed things down a bit.

II. DATAWARP HARDWARE

The first step is to understand how Cray's DataWarp is structured at the hardware level. A service node has 2 PCIe slots, each of which hosts a SSD. We are using the Intel P3608 that provides 3.2 TB per SSD. At the system level, these are each seen as 2 devices so each node sees 4 nvme devices. Cray supports other options so your configuration may be different. These can be over-provisioned which increases the drives endurance but decreases the available space. The default is 3 drive writes per day (DWPD) over a period of 5 years. We have configured ours for 10 DWPD. Since ssd's are consumable devices and wear out over time NERSC wanted to ensure the devices would last with the heavy usage we expect from our users. These are then configured with LVM and XFS. In our case the four devices are combined into a single volume group which is then used to create logical volumes as needed when a request is generated. The size of the logical volume is a multiple of the granularity.

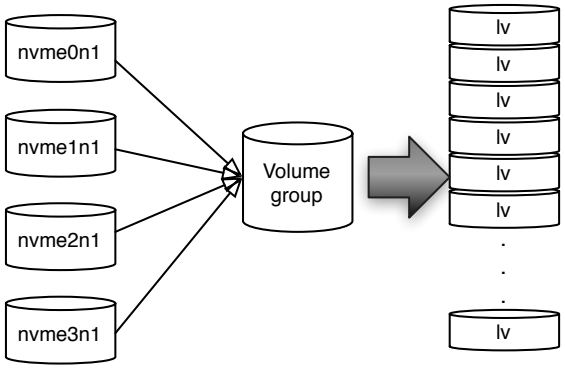

Figure 2. LVM Configuration

At NERSC we currently have a total of 144 DataWarp nodes, this will double when our phase 2 system arrives this summer. Since the XC40 has 2 service nodes per blade, NERSC's configuration places 2 DataWarp nodes on the same service blade. Hardware repairs on these nodes need to ensure they take that detail into account to ensure no data loss on the healthy ssd. When maintenance is done on a DataWarp node on an active system bot the healthy node and the one requiring work need to be drained prior to maintenance. This is a procedure most are familiar with on compute blades. Installing and configuring the DataWarp are not within the scope of this paper; Cray's DataWarp Installation and Configuration Guide provide this information. Some of the decisions regarding the configuration will be discussed since this does have an impact on how the DataWarp can be utilized.

III. Use Cases

Before going further into describing the terminology, it is good to understand how DataWarp can be allocated for use. A DataWarp allocation is described as an instance. A DataWarp instance can be allocated for a job, a job instance, or for a more extended timeframe which is called a persistent instance. A persistent instance can be used by any job requesting it that has the correct unix ownership and permissions. The instance can include multiple DataWarp nodes. A DataWarp job instance can be either shared or private. A persistent instance can't be private. Note that a persistent reservation uses a "#BB" prefix rather than the "#DW" because the command is interpreted by Slurm and not by the Cray Datawarp software. Access of the persistent instance is using the #DW with a persistentdw command. Shared means that all nodes in a job can see the same DataWarp namespace (i.e. mounted filesystem). In private mode each node has it's own local space that other nodes in the allocation can't see or access (like /tmp on a compute node). It can also be striped so the allocated DataWarp space is striped across multiple DW servers.

The user interface allows users to request either a jobdw or create_persistent using the #DW job directive. Options include capacity, access_mode and type; where capacity is the amount of space needed, access_mode is striped or private, and type is scratch (currently the only option available). The type of request is called the configuration. Since the DataWarp can be used to pre-stage data and then stage data out after job completion, there are stage_in and stage_out options for which the user specifies a source, destination, and type; where type is file or dir. Please note that these are the directives used by SLURM these may be different for other workload managers. Once the DataWarp instance is activated, which basically means a mount point has been created, the job can access the instance. In addition, the registration is a way to allow the DataWarp allocation to exist that is associated with a job but can continue to exist after the job completes. This allows the DataWarp instance to continue while data is being written out to the parallel file system. Once the data is written the instance can be torn down so the DataWarp can be used for the next job. Following are some example lines for accessing DataWarp. The first is a standard job instance using striped mode. The second is accessing a previously created persistent instance. The #BB are creating and destroying a persistent DataWarp instance. The last two are examples of staging data into and out of a DataWarp instance.

```
#DW jobdw capacity=10GB
access_mode=striped type=scratch

#DW persistentdw name=myBBname

#BB create_persistent name=myDWname
capacity=10GB access=striped
type=scratch

#BB destroy_persistent name=myBBname

#DW stage_in source=/path/to/filename
destination=$DW_JOB_STRIPED/filename
type=file

#DW stage_out
source=$DW_JOB_STRIPED/dirname
destination=/path/to/dirname
type=directory
```

If a user requests a job using a dw instance, the reference for the job requesting DataWarp space is a session. It maps directly to a job or a persistent instance. A job can also have more than one instance if it is requesting DataWarp for both striped shared access and private access. Since an instance can span nodes, the part of the instance that resides on each of the DataWarp nodes is referred to as a fragment. With SLURM you can see the status of the DataWarp and the current active instances using "scontrol show burst". This provides general information about the pool and how much space is available, then it shows each instance that is allocated, and finally how much space each user is using.

```
nid00837:/var/tmp/slurm # scontrol show burst
Name=cray DefaultPool=wlm_pool Granularity=218016M TotalSpace=872936064M UsedSpace=281458656M
  StageInTimeout=86400 StageOutTimeout=86400 Flags=EnablePersistent,TeardownFailure
  GetSysState=/opt/cray/dw_wlm/default/bin/dw_wlm_cli
  Allocated Buffers:
    Name=userdw1 CreateTime=2016-03-02T15:01:01 Size=1090080M State=allocated UserID=user1(11111)
    JobID=2344665 CreateTime=2016-05-26T14:41:04 Size=21147552M State=staged-in UserID=user2(22222)
    JobID=2344663 CreateTime=2016-05-26T14:40:52 Size=21147552M State=staged-in UserID=user3(33333)
    Name=userdw2 CreateTime=2016-05-09T11:00:43 Size=1090080M State=allocated UserID=user3(33333)
    JobID=2360598 CreateTime=2016-05-27T11:55:40 Size=6813G State=staged-in UserID=user4(44444)
    JobID=2360597 CreateTime=2016-05-27T11:55:40 Size=6813G State=staged-in UserID=user4(44444)
    Name=myBBname CreateTime=2016-02-19T13:45:33 Size=218016M State=allocated UserID=user5(55555)
    Name=userdw3 CreateTime=2016-05-18T13:34:43 Size=31612320M State=allocated UserID=user6(66666)
    Name=userdw4 CreateTime=2016-05-05T16:58:02 Size=1090080M State=allocated UserID=user7(77777)
    Name=userdw5 CreateTime=2016-05-25T12:23:25 Size=218016M State=allocated UserID=user8(88888)
    JobID=2360962 CreateTime=2016-05-27T13:33:06 Size=654048M State=staged-in UserID=user9(99999)
    JobID=2361024 CreateTime=2016-05-27T13:56:57 Size=218016M State=staged-in UserID=user10(10101)
    JobID=2360971 CreateTime=2016-05-27T13:56:57 Size=218016M State=staged-in UserID=user11(11000)
    Name=userdw6 CreateTime=2016-05-05T18:42:48 Size=654048M State=allocated UserID=user12(12121)
    Name=userdw7 CreateTime=2016-05-05T18:31:36 Size=654048M State=allocated UserID=user12(12121)
    Name=userdw8 CreateTime=2016-05-05T16:01:02 Size=654048M State=allocated UserID=user12(61692)

  Per User Buffer Use:
    UserID=user1 (11111) Used=1090080M
    UserID=user3(33333) Used=42295104M
    UserID=user2(22222) Used=1090080M
    UserID=user4(44444) Used=13626G
    UserID=user5(55555) Used=218016M
    UserID=user6(66666) Used=31612320M
    UserID=user7(77777) Used=1090080M
    UserID=user8(88888) Used=218016M
    UserID=user9(99999) Used=654048M
    UserID=user10(10101) Used=218016M
    UserID=user11(11000) Used=218016M
    UserID=user12(12121) Used=1962144M
```

Figure 1. Example of SLURM scontrol show burst output

IV. TERMINOLOGY

Now that you understand the basic hardware configuration and how users can access the DataWarp, it is even more important to understand how these are configured. Because these devices can be used in many ways, it was necessary to provide a flexible interface. Initially, it's confusing but one goal of this paper is to walk through the terms used for DataWarp and their meanings so that their uses will be clear. May of the terms have been introduced in the previous examples of use cases. One of the more confusing aspects of the configuration is the granularity. There is a granularity setting for the node, which defines the smallest size an allocation of that node can be divvied into. There is also a granularity for the pool(s). Once you have defined your DataWarp nodes, they need to be configured into pools. A node can belong to only one pool – it can't be divided into multiple pools. A node does not have to be part of a pool, but the workload manager can't use it if it isn't part of a pool. The DataWarp software allows configuration of multiple pools, however, SLURM currently only supports a single pool. A pool is the largest allocation of space; all DataWarp storage on the system is combined into pools. When a pool is created it you need to specify a granularity. This is important because it is the smallest unit that can be allocated (ours is currently 212GB). One other note is that the node granularity must be a factor of the pools' granularity. Cray provided a program to help choose a granularity setting. There are several factors involved so this is the easiest way to choose a valid granularity setting. One thing to keep in mind, you notice in the use cases that users don't choose how many DataWarp nodes to use in a stripe, they must choose a size that is a multiple of the granularity to get more than a single DataWarp node. Depending on use a larger or smaller granularity may be better. This is one reason you may want to configure multiple pools. With our current configuration our users must sometimes request much larger DataWarp allocations than needed to provide the stripe required for performance.

The terminology is also important to understand because it may help when troubleshooting issues to know which layer is potentially causing a problem. It is much easier to

diagnose problems if you understand the current state of the DataWarp and the jobs that are accessing them.

Here is a brief rundown of the terms and their meanings.

- Session – equates to a job or a persistent Data Warp Instance
- Instance – a DataWarp space that is allocated to a job or persistent over many jobs.
- Fragment – portion of a DataWarp instance on a DataWarp node
- Configuration – defines how a DataWarp instance is used
- Namespace – basically a directory or folder in a scratch configuration. A configuration can have 0 or more namespaces.
- Registration – binds a session with a configuration. This is what holds information on the job for stage-in and stage-out.
- Activation – defines an available instance configuration on a set of nodes.

Each of these terms describes a part of the DataWarp service and can help with diagnosing problems.

DW Allocations (Session/Instance/Fragment) that are stuck or stale can usually be cleared with a reboot of the Name Server (NS) /MetaData Server (MDS) node for the Fragment. This node is the first one listed for the Fragments belonging to the Instance (dwstat fragments | grep Instance#). The NS/MDS can also be identified from the client side, the compute node using it, if the DW allocation is still mounted. Output from the *mount* command contains 'mds=cX-0c0sXnX'.

Attempts to aid recovery by restarting DWS and/or DVS daemons on the DW-servers are NOT recommended and can lead to unpredictable behavior.

ACKNOWLEDGMENT

The authors would like to thank all Cray, SLURM and NERSC staff, who have worked together to make the Cori Phase 1 DataWarp a great resource for our users.

This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02- 05CH11231.

sess	state	token	creator	owner	created	expiration	nodes
2520	CA---	myBBname	CLI	33333	2016-02-19T13:45:33	never	0
3041	CA---	u1_bb1	CLI	11111	2016-03-02T15:01:01	never	0
6185	CA---	2128492	SLURM	55555	2016-05-09T07:13:58	never	96

inst	state	sess	bytes	nodes	created	expiration	intact	label	public	confs
2234	CA---	2520	212.91GiB	1	2016-02-19T13:45:33	never	true	myBBname	true	1
2550	CA---	3041	1.04TiB	5	2016-03-02T15:01:02	never	true	u1_bb1	true	1
5534	CA---	6185	1.87TiB	9	2016-05-09T07:13:58	never	true	I6185-0	false	1

conf	state	inst	type	access_type	activs
2505	CA---	2234	scratch	stripe	0
2821	CA---	2550	scratch	stripe	0
5811	CA---	5534	scratch	stripe	1

V. BEST PRACTICES AND LESSONS LEARNED

Give the DataWarp Service (DWS) the opportunity to complete recovery on its own. This may involve rebooting the DW-server(s) involved in the failure. The current version (5.2UP04) is fairly capable of recovery. The next release in 6.0UP01 is expected to have additional enhancements for recovery.

REFERENCES

[1] "Cray DataWarp Installation and Configuration Guide S-2547-5204," Cray Inc., Revision: b (03-02-16)

[2] "Cray DataWarp Administration Guide S-2557-5204b," Cray Inc., Revision: b (03-02-16)

[3] NERSC help web pages - batch job examples: http://www.nersc.gov/users/computational-systems/cori/burst-buffer/example-batch-scripts/