PBS Professional - Optimizing the "When & Where" of Scheduling Cray DataWarp Jobs

Scott Suchyta
Altair Engineering, Inc.
Troy, MI
scott@altair.com

Abstract - **Integrating Cray DataWarp with PBS Professional was not difficult. The challenge was identifying when and where it made sense to use the "applications I/O accelerator technology that delivers a balanced and cohesive system architecture from compute to storage." As users began to learn more about how their applications performed in this environment, it became clear that when and where jobs ran could greatly affect performance and efficiency. With DataWarp, job data is staged into a "special" storage object (the where), the job executes, and the data is staged out. The catch: minimize wasted compute cycles waiting for the data staging (the when).**

Keywords - Cray; DataWarp; Burst Buffer; Altair; PBS Pro; Job Scheduling

I. INTRODUCTION

Before diving into the details of the Cray DataWarp™ integration with PBS Professional® (PBS Pro), it is important to know the meaning of these technologies and how these technologies are used.

Cray DataWarp provides an intermediate layer of high bandwidth, file-based storage to applications running on compute nodes. It is comprised of commercial SSD hardware and software, Linux community software, and Cray system hardware and software. DataWarp storage is located on server nodes connected to the Cray system's high-speed network (HSN). I/O operations to this storage completes faster than I/O to the attached parallel file system (PFS), allowing the application to resume computation more quickly and resulting in improved application performance. DataWarp storage is transparently available to applications via standard POSIX I/O operations and can be configured in multiple ways for different purposes[1].

PBS Pro is a fast, powerful workload manager designed to improve productivity, optimize utilization & efficiency, and simplify administration for HPC clusters, clouds and supercomputers. PBS Pro automates job scheduling, management, monitoring and reporting, and is the trusted solution for complex Top500 systems as well as smaller cluster owners.

The integration utilizes the PBS Pro Plugin Framework (a.k.a., Hooks) to satisfy the expectations and requirements from several Cray PBS Pro sites.

1. Schedule jobs based on the availability of DataWarp storage capacity.
2. Setup the DataWarp job instance before the job begins execution, such that the following DataWarp functions are executed
    a. paths
    b. setup
    c. data_in
    d. pre_run
3. Teardown the DataWarp job instance after the job terminates (i.e., normal, error, abort), such that the following DataWarp functions are executed
    a. post_run (free the compute nodes!)
    b. data_out
    c. teardown
4. When a non-successful DataWarp exit code (1) is detected, the integration will attempt to re-queue the job if the job has not started execution or leave the data and job instance allocation intact is the job had executed allowing the user/admin to manually resolve any issues.

II. INTEGTATION APPROACH

The integration relies on existing PBS Pro features and utilizes the PBS Pro Plugin Framework (a.k.a., Hooks). The interface is made up of Python objects, members, and methods. You can operate on the objects and use the methods in your Python code. There are multiple hook events, as seen in Figure 1 Simplified view of Hook Events. A hook writer can import the pbs module, which provides an interface to PBS Pro and the hook environment[3].

Each hook can accept (allow) or reject (prevent) the action that triggers it. The hook can modify the input parameters given for the action. The hook can also make calls to functions external to PBS Pro. In addition, the pbs module can read and/or modify things such as job, server, vnode, and queue attributes, and the event that triggered the hook.

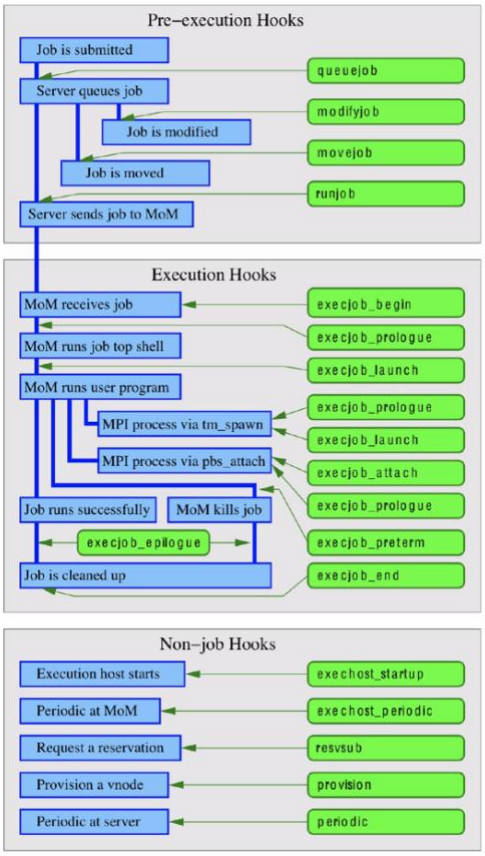

Figure 1 Simplified view of Hook Events

III. INGTEGRATION WORKFLOW

Referring to Figure 2 High-level diagram of PBS Pro daemons and DataWarp hooks, we will walk through the lifecycle of what the user is required to do before job submission through the final steps of DataWarp teardown.

For sites that do not install the PBS Server/Scheduler on the Cray Service Database (SDB) node or allow users to submit jobs from systems outside of the Cray environment, it is required to following the instructions provided in the XC™ Series DataWarp™ API Access from non-Cray Environments Installation Guide[3]. IMPORTANT: DataWarp API Access Requires SLES 12SP3.

A. Validating DataWarp Directives

It is strongly recommended that the user validate the correctness of the DataWarp directives (#DW) by executing a Cray supplied utility call `dw_wlm_cli` prior to job submission.

```
dw_wlm_cli {-f | --function} job_process {-j | --
job} jobscriptfile
```

The user is expected to correct any errors prior to job submission. Failure to correct the error(s) will result in the delaying of the job execution or worse the job may exit the batch queue system without executing the job.

There is a pending PBS Pro RFE for allowing a queuejob hook to read the job script. Once this PBS Pro feature is available in the software, then the user would not be recommended to perform this step.

B. Job Submission

Upon the submission of the job to the PBS Server, the dw_queuejob_hook will be executed. This hook will be responsible for validating the user's DataWarp capacity and pool request. The validation of the user's request should be nearly instantaneous.

From the user's perspective, they will submit the job with PBS custom resources that are specific to DataWarp.

```
required: -l dw_capacity=<value>
optional: -l dw_pool=<value>; assuming
resources_default.dw_pool=<value>
```

Here is an example job script:

```
#!/bin/bash
#PBS -l
select=1:vntype=cray_login+16:ncpus=4:vntype=cray_c
ompute
#PBS -l walltime=1:00:00
#PBS -l dw_capacity=2TiB
#DW jobdw type=scratch access_mode=striped
capacity=2TiB

aprun -n 64 IOR -a POSIX -g -b 8G -t 1M -e -o
$DW_JOB_STRIPED/ior_example1 -G 1234567890 -w -k

aprun -n 64 IOR -a POSIX -g -b 8G -t 1M -e -o
$DW_JOB_STRIPED/ior_example1 -G 1234567890 -W
```

If the job is submitted with -l dw_capacity, then the dw_queuejob_hook will construct the proper job submission request for PBS Professional and create the appropriate attributes for the DataWarp workflow. Otherwise, the dw_queuejob_hook (and other DataWarp hooks) will accept the job "as-is" and it will be assessed by any other site-defined hooks before being accepted by the PBS Server.

If the dw_queuejob_hook should timeout or fail, the user will receive an error message.

If the hook times out, the administrator can increase the duration of the hook's alarm.

C. Job Eligibility (Job Scheduling)

Relying on existing PBS Professional job scheduling capabilities (parameter server_dyn_res, which calls dw _capacity_check), the scheduler will compare the user's dw_capacity request with the availability of DataWarp capacity. If there is sufficient capacity and all other scheduler policies are satisfied, the scheduler will inform the Server of which nodes to dispatch the job and request the job to be executed. If there is insufficient capacity, then the job will remain queued, but will be eligible for scheduling at the next scheduling cycle.

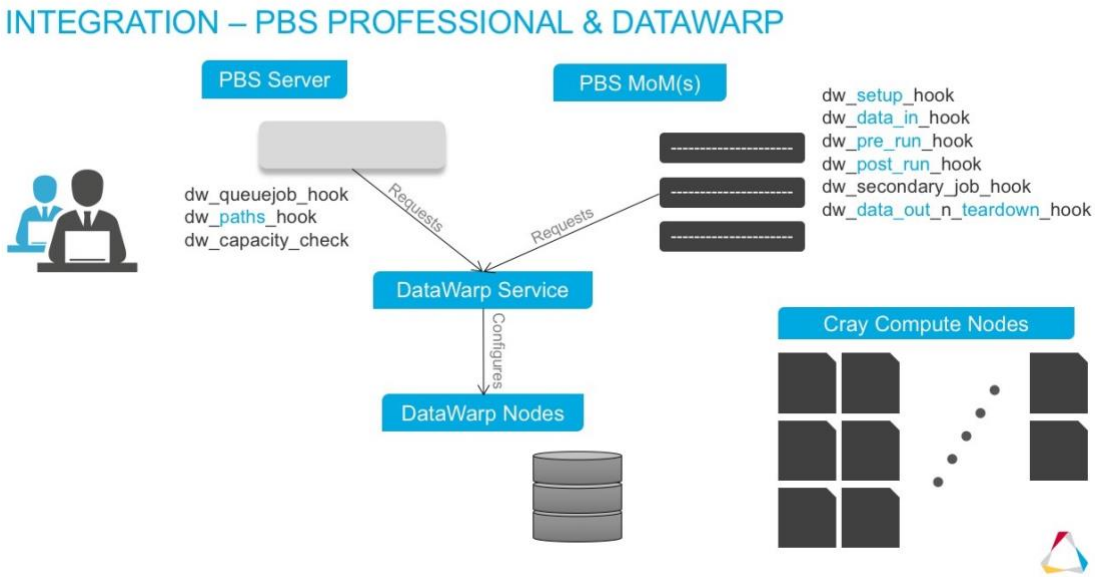

Figure 2 High-level diagram of PBS Pro daemons and DataWarp hooks

D. DataWarp Path

Prior to the job being dispatch to the nodes, the Server will execute a runjob hook, dw_paths_hook, that will setup the job's environment with the DataWarp-specific environment variables, which will be referenced in the user's job script. The updating of the job's environment with the DataWarp-specific environment variables should be nearly instantaneous. If the setup is successful, then the job will proceed to staging in the data.

If this hook should fail or timeout, then the job is re-queued and put in a 'H'old state. By placing the job into a 'H'old state, the administrator can investigate why the setup of the job's environment variables failed. Depending on the resolution of the issue, the administrator will be able to release the hold (`qrls -h s <pbs_jobid>`), and the job will be re-considered in the next scheduling cycle.

The dw_paths_hook will record information in the PBS Server logs (`$PBS_HOME/server_logs`). It is possible to increase verbosity of the log messages to troubleshoot the issue by enabling debug in the hook.

E. DataWarp Setup

Before the job begins execution, PBS Professional will need to setup the storage object and stage-in the user's data, if specified. The PBS MOM will execute an execjob_prologue hook, dw_setup_hook, that will allocate and configure a new storage object for the job. In practice, the setup of the job's storage object will take a few seconds. A successful setup will transition the job to staging the data in to the storage object.

If this hook should fail or timeout, then the job is re-queued and put in a 'H'old state. By placing the job into a 'H'old state, the administrator can investigate why the setup of the job's environment variables failed. Depending on the resolution of the issue, the administrator will be able to release the hold (`qrls -h s <pbs_jobid>`), and the job will be re-considered in the next scheduling cycle.

The dw_setup_hook will record information in the PBS MOM logs (`$PBS_HOME/mom_logs`). It is possible to increase verbosity of the log messages to troubleshoot the issue by enabling debug in the hook.

If the hook is timing out, the administrator can increase the duration of the hook's alarm.

F. DataWarp Stage-in

After a successful setup of the storage object, PBS Professional will initiate the DataWarp data_in function. It is expected that the data being staged in will be relatively quick; less than 5 minutes. The PBS MOM will execute an execjob_prologue hook, dw_data_in_hook, that will stage data into a storage object from an external source. A successful stage-in will connect the compute nodes to the DataWarp job instance.

If this hook should fail or timeout, then the job is re-queued and put in a 'H'old state. By placing the job into a 'H'old state, the user and/or administrator can investigate why the stage-in of the job's data failed. Remember, that the `dw_wlm_cli -f job_process`, which the user is expected to execute before job execution does not validate paths of files or directories. So, it is possible that the user has specified an invalid path.

Depending on the resolution of the issue, the administrator will be able to release the hold (`qrls -h s <pbs_jobid>`), and the job will be re-considered in the next scheduling cycle. NOTE: the storage object from the previous attempt will have been torn down.

The dw_data_in_hook will record information in the PBS MOM logs (`$PBS_HOME/mom_logs`). It is possible to increase verbosity of the log messages to troubleshoot the issue by enabling debug in the hook.

If the hook is timing out, the administrator can increase the duration of the hook's alarm.

G. DataWarp Pre Run

After a successful stage-in of the user's data, the PBS MOM will execute an execjob_launch hook, dw_pre_run_hook, that will connect the compute nodes to the storage object. In practice, the connection of the compute nodes is nearly instantaneous. A successful setup will transition the job to begin execution.

If this hook should fail or timeout, then the job is re-queued and put in a 'H'old state. By placing the job into a 'H'old state, the administrator can investigate why the setup of the job's environment variables failed. Depending on the resolution of the issue, the administrator will be able to release the hold (`qrls -h s <pbs_jobid>`), and the job will be re-considered in the next scheduling cycle.

The dw_pre_run_hook will record information in the PBS MOM logs (`$PBS_HOME/mom_logs`). It is possible to increase verbosity of the log messages to troubleshoot the issue by enabling debug in the hook.

If the hook is timing out, the administrator can increase the duration of the hook's alarm.

H. DataWarp Post Run

Once the job terminates (i.e., normal, error, or abort) PBS Professional will initiate the DataWarp post_run function. The PBS MOM will execute an execjob_epilogue hook, dw_post_run_hook, that will disconnect compute nodes from storage object. In practice, the disconnect of the compute nodes from the storage object will take a few seconds. A successful post_run will initiate the stage-out and teardown of the job.

If this hook should fail or timeout, then the job exits the system without initiating the proceeding hook that will be responsible for the stage-out and teardown of the storage object. This is an intention decision to avoid purging the user's data.

The dw_post_run_hook will record information in the PBS MOM logs (`$PBS_HOME/mom_logs`). It is possible to increase verbosity of the log messages to troubleshoot the issue by enabling debug in the hook. The administrator will need to investigate why the post_run failed and will need to manually rectify the issue.

If the hook is timing out, the administrator can increase the duration of the hook's alarm.

I. DataWarp Stage-out & Teardown Job

At this point, the compute nodes have been successfully disconnected from the storage object, and the data can be staged-out and the storage object can be torn down. Although, it is expected that the data produced by the job will be large (e.g., 100s of GB and more). In an effort to avoid having PBS Professional keep the compute nodes allocated to the job while staging out data, the PBS MOM will execute an execjob_epilogue hook, dw_secondary_job_hook, as the user by submitting a secondary job. The secondary job will be responsible for triggering the stage-out of the user's data and teardown of the storage object. In practice, the submission of the secondary job will be instantaneous. A successful submission of the secondary job will allow the job to exit, barring no other site-specific hooks are executing later.

If this hook should fail or timeout, then the job exits the system without initiating the stage-out or teardown of the storage object. This is an intentional decision to avoid purging the user's data.

The dw_post_run_hook will record information in the PBS MOM logs (`$PBS_HOME/mom_logs`). It is possible to increase verbosity of the log messages to troubleshoot the issue by enabling debug in the hook.

If the hook is timing out, the administrator can increase the duration of the hook's alarm.

J. DataWarp Stage-out & Teardown

The stage-out of the user's data and teardown of the storage object is initiated by the secondary job that was submitted on behalf of the user. The job itself will execute very quickly (i.e., `/bin/true`) on the PBS MOM node that had executed the original job. Then the PBS MOM will execute an execjob_epilogue hook, dw_data_out_n_teardown_hook, that will stage-out the user's data. If data is staged out successfully, the teardown of the storage object is executed.

If this hook should fail or timeout, then the job exits the system without initiating the stage-out and teardown of the storage object. This is an intentional decision to avoid purging the user's data. Remember, that the `dw_wlm_cli -f job_process`, which the user is expected to execute before job execution does not validate paths of files or directories. So, it is possible that the user has specified an invalid path.

The dw_data_out_n_teardown_hook will record information in the PBS MOM logs (`$PBS_HOME/mom_logs`). It is possible to increase verbosity of the log messages to troubleshoot the issue by enabling debug in the hook. The administrator will need to investigate why the post_run failed, and will need to manually rectify the issue.

It is expected that the data produced by the original job will be large (e.g., 100s of GB and more), which could take upwards of 30 minutes or more to stage out. If the hook is timing out, the administrator can increase the duration of the hook's alarm.

IV. INSTALLATION

The site administrator will need to create the DataWarp pool(s) prior to configuring PBS Professional. Review the DataWarp manuals for details on how to setup and configure DataWarp pool before proceeding.

After successfully creating the DataWarp pool(s), the administrator, as root, will need to complete a few manual steps to install and configure the integration of DataWarp with PBS Professional. These steps consist of importing the DataWarp hooks, updating the PBS Scheduler, and finalizing the default values.

V. CONFIGURATIONS

The integration has several configuration options, which the site will need to consider if the defaults are reasonable for their site.

A. DataWarp Hook Alarms

In practice, the `dw_wlm_cli` command can take several seconds to complete execution for several of the functions. The data_in and data_out could take multiple minutes, depending on the user's data. Therefore, it advisable for the site to become familiar with the execution time of the `dw_wlm_cli` command and adjust the respective hook alarms. If the `dw_wlm_cli` command exceeds the hook alarm, then the hook will fail.

If one of the DataWarp hooks should timeout, the proceeding hooks in the workflow will execute. This is by design of the PBS Pro Plugin framework. However, each hook has logic to handle a failure scenario.

The DataWarp hooks have the following default alarm attributes.

```
set hook dw_queuejob_hook alarm = 10
set hook dw_paths_hook alarm = 600
set hook dw_setup_hook alarm = 600
set hook dw_data_in_hook alarm = 600
set hook dw_pre_run_hook alarm = 600
set hook dw_post_run_hook alarm = 600
set hook dw_secondary_job_hook alarm = 600
set hook dw_data_out_n_teardown_hook alarm = 1800
```

To change a hook's alarm attribute, the administrator will execute

```
qmgr -c "set hook <hook_name> alarm = <value>"
```

where the value is a positive integer, and the units are in seconds.

B. DataWarp Command Retry

Each DataWarp hook can be configured to retry the DataWarp command, `dw_wlm_cli`. By default, each DataWarp hook will attempt to retry the command three (3) times.

```
number_of_tries = 3
```

NOTE: the dw_queuejob_hook does not have this option.

If you change the debug variable to True, then you will need to re-import the hook into the PBS Server via qmgr command.

```
qmgr -c "import hook <hook_name> application/x-
python default <hook_name>.py"
```

C. DataWarp Hook Verbosity

By default, each DataWarp hook is configured to log minimal job-specific information in the daemon logs. However, it is possible to increase the verbosity of the DataWarp hook to help troubleshoot an issue. Each DataWarp hook has a variable call debug and is defined after the import declaration of the hook.

```
debug = False
```

IMPORTANT: there is a separate debug variable for the run_command function, which will only enable the debug message for when the hook is executing a command, e.g., `dw_wlm_cli` and PBS commands.

If you change the debug variable to True, then you will need to re-import the hook into the PBS Server via qmgr command.

```
qmgr -c "import hook <hook_name> application/x-
python default <hook_name>.py"
```

VI. TROUBLESHOOTING

The integration attempts to log as much useful and user-friendly information as possible. From the user's perspective, there are a few conditions where the job submission may fail, and the error message is returned to the user instantaneously. From the administrator's perspective, troubleshooting the DataWarp workflow can be tricky. When a non-successful DataWarp exit code is detected, the integration attempts to re-queue the job if the job has not started execution or leaves the data and job instance allocation intact if the job had executed allowing the user/admin to manually resolve any issues.

A. Troubleshooting - Job Submission

```
qsub: Missing required option dw_capacity. Submit
with -l dw_capacity=<value>.
```

The user has submitted a job with -l dw_pool, but neglected to submit the job with -l dw_capacity. A DataWarp job submission requires -l dw_capacity.

```
qsub: Job requested -l dw_capacity - Missing
required -l dw_pool. Either resubmit with -l
dw_pool=<value>, or request administrator to set
the resources_default.dw_pool=<value> attribute.
```

The user submitted with the required dw_capacity attribute, however, a default dw_pool was not configured within qmgr. The administrator can configure the default dw_pool, or the user can request -l dw_pool at submission time.

```
qsub: Job requested a DataWarp pool (my_pool) that
is not available. Resubmit with eligible DataWarp
pool (wlm_pool,dev,test), or admin needs to
configure DataWarp pool.
```

The -l dw_pool request does not match the eligible DataWarp pools configured within qmgr. The user can resubmit the job requesting an eligible DataWarp pool, as provided in the error message. Or, the administrator will need to configure the missing DataWarp pool within qmgr.

B. Troubleshooting - DataWarp Workflow

DataWarp hooks were written to associate the record with the PBS jobid in PBS daemon logs:

```
PBS_HOME/server_logs: dw_queuejob_hook,
dw_paths_hook

PBS_HOME/mom_logs: dw_setup_hook, dw_data_in_hook,
dw_pre_run_hook, dw_post_run_hook,
dw_secondary_job_hook, dw_data_out_n_teardown_hook
```

The user/administrator will be able to parse the logs by jobid or use the PBS Pro tracejob command.

```
DataWarp: jobscript did not exist. Contact your
Administrator.
```

The `dw_wlm_cli` command requires the job script through the life cycle of the job's execution. If the hook cannot locate the job script, then this is a critical issue that will prohibit the integration from executing correctly.

The job script will be found in `$PBS_HOME/server_priv/jobs` or `$PBS_HOME/mom_priv/jobs` (when executing).

The job script filename will be based on the PBS jobid with a 'SC' file extension (e.g., 3855.dw01.SC). The log entry will have the full path to the expected job script.

The administrator should verify the

1. filesystem where `$PBS_HOME` is located is not full or have read-only permissions.
2. PBS Professional commands are installed.
3. `/etc/pbs.conf` file's `$PBS_EXEC` contains the correct path to the PBS Professional commands.

```
DataWarp <function> failed. Contact your
Administrator.
```

The hook attempted to execute the

and had received a non-zero exit code from the command.

The administrator should consider enabling debug within the hook; specifically, in the run_command function. By enabling debug, the stdout, stderr, and rc will be logged in the daemon log file.

```
DataWarp <data_in | data_out> failed. Verify #DW
directives (file and directory paths) or contact
your Administrator.
```

The hook attempted to execute the `dw_wlm_cli -f <data_in | data_out>` function and had received a non-zero exit code from the command.

It is possible that the user has specified an invalid path to a file or directory.

The administrator should consider enabling debug within the hook; specifically, in the run_command function. By enabling debug, the stdout, stderr, and rc will be logged in the daemon log file.

VII. Lesson Learned

As seen at all of the Cray PBS Professional sites, it is crucial that the site administrator verifies the `dw_wlm_cli` functions are working properly from the service and login nodes. Although the integration makes every attempt to requeue or hold jobs, so nothing is lost, the outcome can be confusing for the admins or users.

The use of cray_eswrap around the `dw_wlm_cli` command produced errors/exceptions, which will break the integration. Fortunately, the integration records the `dw_wlm_cli` command and arguments for admins to test themselves.

For Cray PBS Professional sites using RHEL/CentOS for PBS Professional Server/Scheduler cannot use integration because DataWarp API Access Requires SLES 12SP3.

References

[1] (2018) XC™ Series DataWarp™ User Guide (CLE 6.0 UP06)], Cray Inc. [Online] Available: https://pubs.cray.com/pdf-attachments/attachment?pubId=00529370-DB&attachmentId=pub_00529370-DB.pdf

[2] (2018) XC™ Series DataWarp API Access from non-Cray Environments Installation Guide (CLE 6.0 UP06), Cray Inc. [Online] Available: https://pubs.cray.com/pdf-attachments/attachment?pubId=00532494-DA&attachmentId=pub_00532494-DA.pdf

[3] (2018) PBS Professional Big Book (14.2, contains Install, Administrator, Reference, User, and Program Guides), Altair Engineering Inc. [Online] Available: https://pbsworks.com/pdfs/PBS14.2.1_BigBook.pdf