PBS PROFESSIONAL - OPTIMIZING THE "WHEN & WHERE" OF SCHEDULING CRAY DATAWARP JOBS

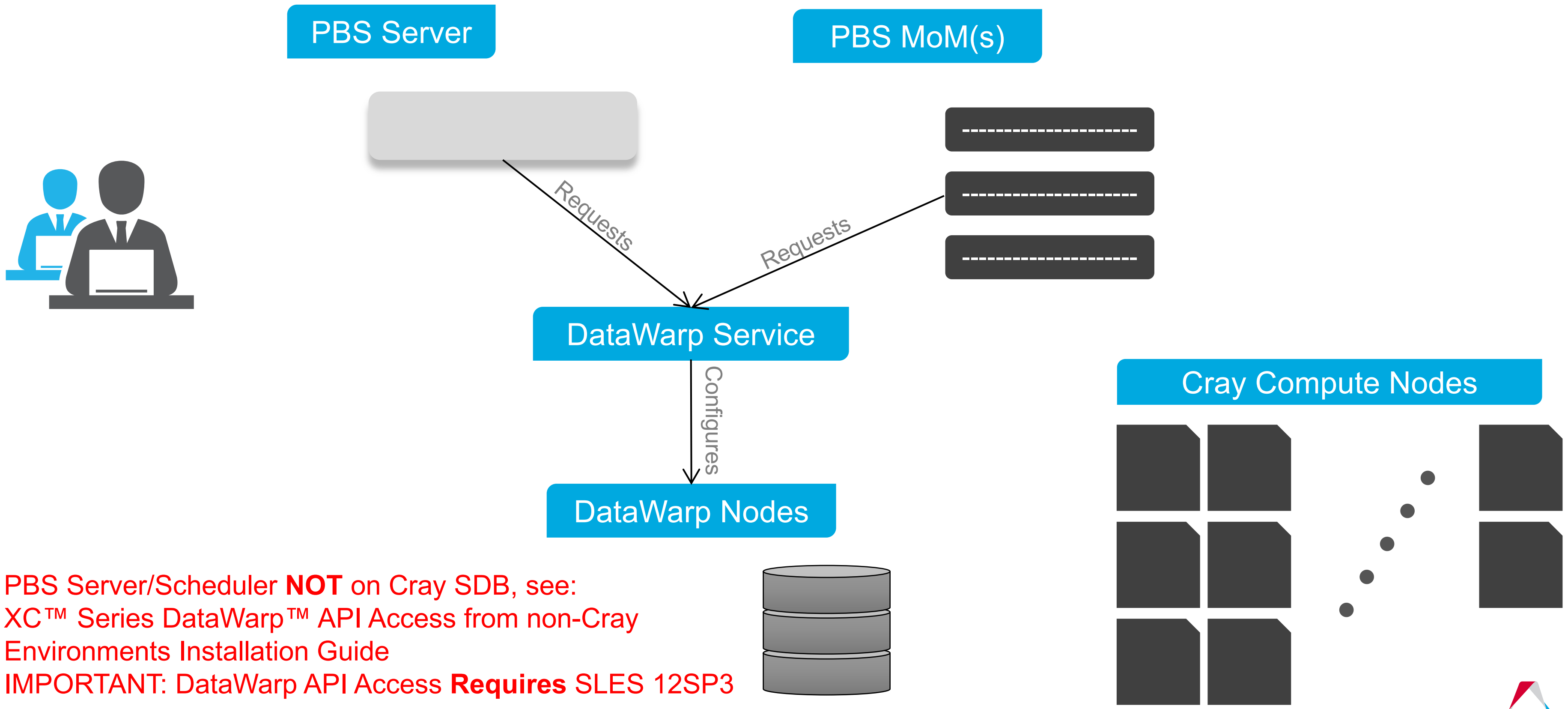
Scott Suchyta | Director, Solutions & Integrations | May 20-24, 2018

Altair

OBJECTIVES

- Schedule jobs based on the availability of DataWarp storage capacity.
- Setup the DataWarp *job instance* before the job begins execution
- Teardown the DataWarp *job instance* after the job terminates (i.e., normal, error, abort)
    - Release compute nodes – don't waste compute cycles waiting for data staging
- When a non-successful DataWarp exit code is detected,
    - Attempt to re-queue the job if the job has not started execution, or
    - Leave the data and job instance allocation intact if the job had executed allowing the user/admin to manually resolve any issues.

PBS PROFESSIONAL & DATAWARP

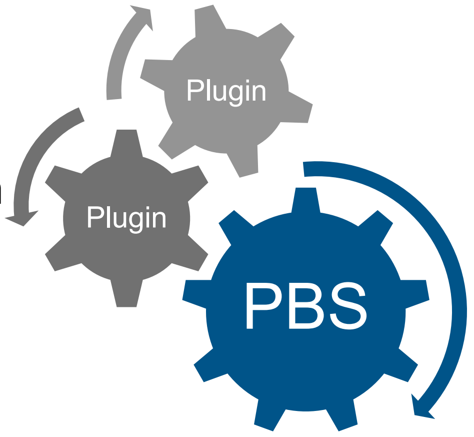


PBS Server/Scheduler **NOT** on Cray SDB, see:
XC™ Series DataWarp™ API Access from non-Cray Environments Installation Guide
IMPORTANT: DataWarp API Access **Requires** SLES 12SP3

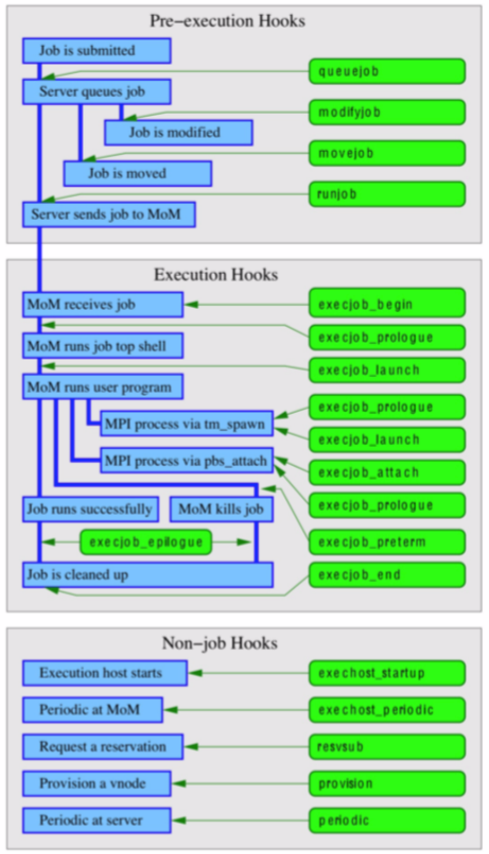
INTEGRATION – PBS PLUGINS ("HOOKS")

Framework for Agility and Innovation

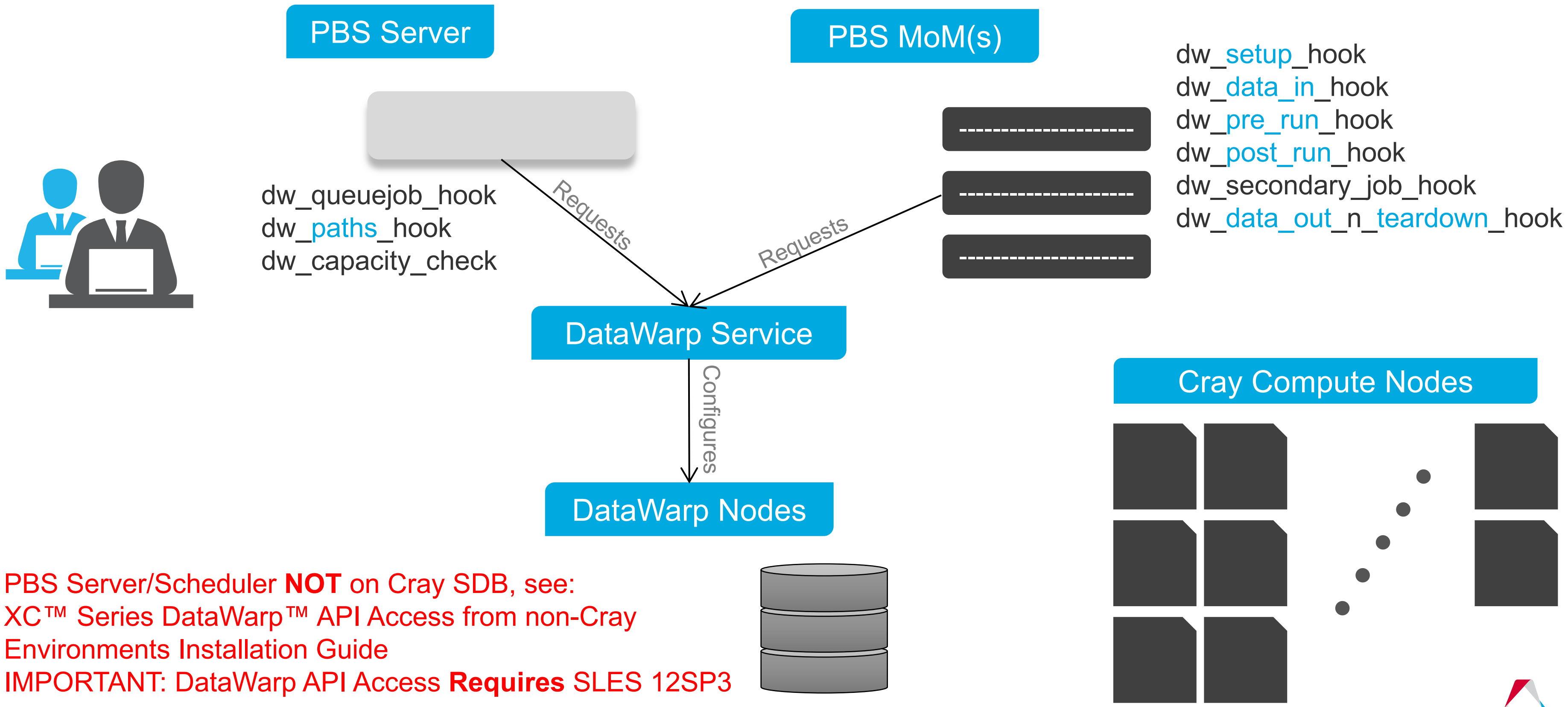
- Meet unique enterprise requirements
- Deliver site-specific and 3rd party integrations
- Support platform-specific features on day 1
- Build novel extensions
- Crowdsource solutions
- Share (even sell) via PBS ecosystem



Say “Yes!” to unique requests



INTEGRATION – PBS PROFESSIONAL & DATAWARP



PBS Server/Scheduler **NOT** on Cray SDB, see:
XC™ Series DataWarp™ API Access from non-Cray Environments Installation Guide
IMPORTANT: DataWarp API Access **Requires** SLES 12SP3

INTEGRATION – PBS PROFESSIONAL & DATAWARP

- dw_path_hook: Before job dispatch, the Server will invoke the DW path function to create the job's environment with the DataWarp-specific environment variables, which will be referenced in the user's job script.

- dw_setup_hook: Before the job begins execution, the PBS MOM will invoke the DW setup function to configure and allocate a new storage object for the job.

- dw_data_in_hook: After a successful setup, the PBS MOM will invoke the DW data_in function to initiate the DataWarp data_in function.

- dw_pre_run_hook: After a successful stage in, the PBS MOM will invoke the DW pre_run function to connect the compute nodes to the storage object.

INTEGRATION – PBS PROFESSIONAL & DATAWARP

- dw_post_run_hook: Once the job terminates (i.e., normal, error, or abort), the PBS MOM will invoke the DW post_run function to disconnect compute nodes from storage object. If the post_run is successful, then the dw_secondary_job_hook will execute to free the compute nodes.

- dw_data_out_n_teardown_hook: At this point, the compute nodes have been successfully disconnected from the storage object, the PBS MOM will invoke the data_out and teardown functions.

INSTALLATION & CUSTOMIZATION

INSTALLATION

1. Update /etc/pbs.conf with PBS_DATAWARP variable
2. Unpack PBS Professional DataWarp integration package
3. Import Hooks via qmgr
4. Create & Define custom resources for DataWarp pool(s) via qmgr
5. Update Scheduler Configuration file (PBS_HOME/sched_priv/sched_config)

No Restarting PBS daemons ☺

CUSTOMIZATION - RETRIES

- Cray interface can timeout or fail unexpectedly – doesn't hurt to retry ☺
- By default, each DataWarp hook will attempt to retry the command three (3) times

```
number_of_tries = 3
```

- Tune the DataWarp Command Retry based on site's experience via hook and re-import

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

CUSTOMIZATION - ALARMS

- Cray interface could take several seconds to minutes to respond
- Data staging duration varies from site to site (Not Applicable for Transparent Cache Mode)
- Tune the Hook Alarms based on site's experience (defaults below)

```
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
```

- Change Hook Alarm via qmgr (units in seconds)

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

CUSTOMIZATION - VERBOSITY

- By default, minimal logging in hook (e.g., start, end events)

```
debug = False
```

- Increase verbosity (including logging of DataWarp interface responses) by enabling debug via the hook and re-import

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

- DataWarp hooks record information into the 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

USER EXPERIENCE

USER EXPERIENCE – JOB SUBMISSION

- Integration introduces two custom resources

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

- The `dw_queuejob_hook` will construct the proper job submission request and create the appropriate attributes for the DataWarp workflow.
- If the `dw_queuejob_hook` should timeout or fail, the user will receive an error message.
- If the hook is timing out, the administrator can increase the duration of the hook's alarm.

IMPORTANT: Administrator can set hook execution order. Consider running dw_queue_job hook after other site specific hooks.

USER EXPERIENCE – JOB SUBMISSION

- Example job script

```
#!/bin/bash
#PBS -l select=1:vntype=cray_login+16:ncpus=4:vntype=cray_compute
#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
```

JOB ELIGIBILITY / SCHEDULING THE "WHEN"

- Relying on **existing** PBS Professional job scheduling capabilities...
- The scheduler will compare the user's `dw_capacity` (and `dw_pool`) request with the availability of DataWarp capacity (`dw_capacity_check`).
- 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.
- Else (insufficient capacity), the job will remain queued, but will be eligible for scheduling at the next scheduling cycle.

TROUBLESHOOTING

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.

TROUBLESHOOTING – DATAWARP WORKFLOW

- When a non-successful DataWarp exit code is detected,
  - Attempt to re-queue the job if the job has not started execution, or
  - Leave the data and job instance allocation intact if the job had executed allowing the user/admin to manually resolve any issues.
  - One more thing.. Log it, too!! ☺

- 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 Professional tracejob command.

TROUBLESHOOTING – DATAWARP WORKFLOW

```
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.

TROUBLESHOOTING – DATAWARP WORKFLOW

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

The hook attempted to execute the `dw_wlm_cli -f <function>` 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.

TROUBLESHOOTING – DATAWARP WORKFLOW

```
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.

LESSONS LEARNED

LESSONS LEARNED

- Crucial → Verify the `dw_wlm_cli` functions are working properly from the service and login nodes
  - Yes - The integration makes every attempt to requeue or hold jobs so nothing is lost, but it can be confusing for the admins or users
  - The use of `cray_eswrap` around the `dw_wlm_cli` command produced errors/exceptions
  - Fortunately, the integration records the `dw_wlm_cli` command and arguments for admins to test themselves.

- PBS Professional customers use a white-box Server/Scheduler (RHEL, SLES)
  - Advantage for sites managing multiple clusters, and using external high availability add-ons (e.g., RHEL HA)
  - Sites using RHEL/CentOS for PBS Professional Server/Scheduler cannot use integration because DataWarp API Access **Requires** SLES 12SP3.

FUTURE WORK

FUTURE INTEGRATION WORK

- Incorporate the `dw_wlm_cli -f process` into a queuejob hook for ease of use:
  - Detecting errors with user's #DW directives (no wrapper required)
  - Transparently requesting user's capacity and pool (no longer require user to supply on qsub line)
  - Pending - PBS-18715 Admin would like to read/modify a user's jobscript at job submission.

- If DataWarp `pre_run` function should fail, then execute `teardown -H`
  - Rare condition.. But it did happen at customer site.

- PBS Professional v18 introduces a new feature to release nodes
  - Potential to eliminate the `dw_secondary_job_hook` – simplify the workflow

THANK YOU!