DataWarp Administration Tutorial

David Paul
Benjamin Landsteiner
CUG2018 – May 2018
Introductions

● Dave Paul
  ● LBNL/NERSC
  ● Member of the Computational Systems Group
  ● Focus on filesystem stability on Cray systems, including DataWarp
  ● Involved with the NERSC DataWarp Early User program

● Ben Landsteiner
  ● Cray Inc. for 9 years
  ● DataWarp architect
  ● Prior projects include ALPS, WLMs, KNC, kernel

May 2018
Agenda

- Introductions and Format (5 minutes; done)
- DataWarp Introduction (25 minutes)
- System Configuration & Tuning (30 minutes)
- Log files & Analysis (30 minutes)
- Break (30 minutes)
- Slurm & DataWarp (30 minutes)
- Common Problems & Solutions (30 minutes)
- Tools for DataWarp System Administration (30 minutes)
Format

● Plenty of material in the tutorial
  ● Slide material augments the official documentation
  ● If something isn't clear, let us know and we will try to improve it

● Please ask questions throughout!

● Some examples come from NERSC
  ● Log files used in tutorial available on request
DataWarp Introduction
Overview – What is DataWarp?

- **DataWarp is an IO Accelerator**
  - An implementation of the Burst Buffer concept, plus more
- **Has both Hardware & Software components**
- **Hardware**
  - XC service node, directly connected to Aries network
  - PCIe SSD Cards installed on the node
- **Software**
  - DataWarp Service daemons
  - DataWarp Filesystems (using DVS, LVM, XFS)
  - Integration with WorkLoad Managers (Slurm, M/T, PBSpro)
Usage overview (scratch)

Without DataWarp

1: #!/bin/bash
2: #SBATCH --ntasks 3200
3:
4: export JOBDIR=/lus/global/my_jobdir
5: srun -n 3200 a.out

With DataWarp Scratch

1: #!/bin/bash
2: #SBATCH --ntasks 3200
3: #DW jobdw type=scratch access_mode=striped capacity=1TiB
4: #DW stage_in type=directory source=/lus/global/my_jobdir destination=$DW_JOB_STRIPED
5: #DW stage_out type=directory source=$DW_JOB_STRIPED destination=/lus/global/my_jobdir
6:
7: export JOBDIR=$DW_JOB_STRIPED
8: srun -n 3200 a.out
Usage overview (cache)

Without DataWarp

```bash
#!/bin/bash
#SBATCH --ntasks 3200
export JOBDIR=/lus/global/my_jobdir
srun -n 3200 a.out
```

With DataWarp Transparent Caching

```bash
#!/bin/bash
#SBATCH --ntasks 3200
#DW jobdw type=cache access_mode=striped pfs=/lus/global capacity=10TiB
export JOBDIR=$DW_JOB_STRIPED_CACHE/my_jobdir
srun -n 3200 a.out
```
Hardware Overview

- SN: Storage Node
- CN: Compute Node
- DW: DataWarp Node
- LN: LNET Router Node
- OSSs/OSTs: Object Storage Service/OST
- Aries: Software-defined storage
- Infiniband Fabric: High-speed network

Legend:
- SSD: Solid State Drive
- HCA: High-Performance Computing Adapter

May 2018

9
Software Overview (Orchestration & Data)
Transparent Cache Data Path

- **Compute nodes**
  - DVS client

- **DataWarp nodes**
  - DVS server
  - SSD space
  - DataWarp File System
  - Data Caching Filesystem
  - PFS client
Software Overview (Data path, scratch)
Software Overview (Data path, cache)
Transparent Cache Orchestration

- Sets up and manages the data path
- Workload Managers
- DataWarp Service
- Node Health services
  - Scalable fanout of commands
- MUNGE
  - Security
Software Overview (Orchestration)
API Clients

- API clients send requests through the DW API gateway
  - `dwrest`
- `dw_wlm_cli`: commandline script for interacting with API GW for WLMs
- `dwstat`: status command
- `dwcli`: perform actions
- Authentication through MUNGE
- API GW discovery - `dwgateway` and `libdws_thin0`
  - Not shown
API Gateway - dwrest

- RESTful API with JSON
- HTTPS
- MUNGE authentication
  - nginx
  - gunicorn
- API GWs on multiple nodes possible
  - Resiliency
dwsd

- DataWarp Scheduler Daemon
- Persists state in dwsd.db SQLite file
- Processes requests from API GW
- Dispatches tasks to dwmd, such as interacting with LVM, mounting filesystems, initiating end of job stage out
- Learns about dwmd from heartbeats
- All messages encrypted with MUNGE
- Uses RCA to verify node crashes
- Dispatched requests are asynchronous
- Responses to dwmd requests are received asynchronously
- DataWarp Manager Daemon
- Exists on every SSD-endowed node under DWS ownership
- Interacts with LVM volume group dwcache
- dwmd forks for every request
- Periodically heartbeats back to dwsd
- Responses to dwsd requests occur with new socket connection
- Existing Cray software component, part of Node Health
- Scalably executes commands, pushes files, etc via a Tree-based overlay network
dws*.py (ok, and lvm*.py too)

- **Python scripts for performing actual tasks**
  - Creating/destroying logical volume
  - Mounting/unmounting XFS, dwfs, dcf$s, DVS mounts
  - Managing swap files
  - Kicking off end-of-job stage-out
  - Checking on health of dwcache volume group
  - Requesting SSD health information from capmc
- **Control data sent via a JSON file pushed with xtnhd**
- **Uses cgroups and “out of order task” (ooot) cache to ensure tasks are carried out *in order***
  - It is possible though unlikely for a teardown task to get processed *before* a setup task, which can lead to admindown nodes
Security within DWS

- Relies heavily on MUNGE
  - Works well in environments where UID and GID namespace is identical across nodes
- DWS daemons only process messages that...
  - ...are encrypted with MUNGE
  - ...were sent by trusted user IDs
Client security

- Client-API gateway communication over HTTPS
- Client authentication with MUNGE in HTTP header
- Authorization
  - Admins, users, and none
- Admins specified in configuration file, default root and crayadm
- Admins can see everything, do almost anything, and do things on behalf of users
- Users can see things associated with or usable by their user id
System Configuration & Tuning
Points of Configuration

- cray_dws config set
- Over-provisioning
  - Intel P3608 only
- LVM setup
- Software Runtime
  - Pools
  - Putting server nodes in to pools
- WLM
  - Slurm example
Points of Configuration: cray_dws

- **Specify DataWarp servers in datawarp_nodes node group**
- **Enable cray_ipforward service**
  - DWS uses capmc for SSD health information, which requires access to SMW
- **Enable cray_munge service**
  - DWS uses MUNGE for authentication
- **Enable cray_persistent_data service**
  - Persisting /var/opt/cray/dws ensures DW filesystems and pool data survive reboots
- **Configure cray_dws**
  - Enable the service
  - Set managed nodes to datawarp_nodes node group
  - Set api gateway nodes to login_nodes node group
  - Set external_api_gateway_hostnames to FQDNs of login nodes with external network access to allow eLogin nodes and other non-XC nodes native access to the DataWarp RESTful API
  - Set dwrest_cachemount_whitelist to list of PFS on system
  - Set allow_dws_cli_from COMPUTES if needed
- **Enable cray_dw_wlm service**
  - Configuration options that impact behavior of dw_wlm_cli during failures
Points of Configuration: Over-provisioning

- Intel P3608 SSDs only
- Increases drive lifetime by reducing byte quantity available for filesystems
  - …but probably not needed – your call!
- Replace /dev/nvme0 with /dev/nvme1, /dev/nvme2, /dev/nvme3 to get all devices on a node
- See Cray S-2564 for value for your SSD

```
dwnode# module load linux-nvme-ctl
dwnode# nvme set-feature /dev/nvme0 \ 
>  -n 1 -f 0xC1 -v 3125623327
```
Points of Configuration: LVM setup

- Only needed one time per set of hardware
- Create Volume Group dwcache from all available SSDs
- Restart dwmd daemon when finished

```
dwnode# pvcreate /dev/nvme0n1 /dev/nvme1n1 \ 
> /dev/nvme2n1 /dev/nvme3n1
<success output>
dwnode# vgcreate dwcache \ 
> /dev/nvme0n1 /dev/nvme1n1 /dev/nvme2n1 /dev/nvme3n1
<success output>
dwnode# systemctl start dwmd
```
LVM Volume Group dwcache
LVM Tools Bootcamp

● Logical Volume Manager
● Block devices converted to *Physical Volumes* with `pvcreate`
  ● View PVs with `pvs/pvdisplay`
● PVs grouped in to *Volume Groups* with `vgcreate`
  ● View VGs with `vgs/vgdisplay`
● *Logical Volumes* carved out of VGs with `lvcreate`
  ● View LVs with `lvs/lvdisplay`
● Remove with `lvremove`, `vgremove`, or `pvremove`
Underlying SSD file system

<table>
<thead>
<tr>
<th>PV</th>
<th>VG</th>
<th>Attr</th>
<th>PSize</th>
<th>PFree</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/nvme0n1</td>
<td>dwcache</td>
<td>lvm2</td>
<td>1.46t</td>
<td>1.46t</td>
</tr>
<tr>
<td>/dev/nvme1n1</td>
<td>dwcache</td>
<td>lvm2</td>
<td>1.46t</td>
<td>1.46t</td>
</tr>
<tr>
<td>/dev/nvme2n1</td>
<td>dwcache</td>
<td>lvm2</td>
<td>1.46t</td>
<td>1.46t</td>
</tr>
<tr>
<td>/dev/nvme3n1</td>
<td>dwcache</td>
<td>lvm2</td>
<td>1.46t</td>
<td>1.46t</td>
</tr>
</tbody>
</table>

```
# pvs

Total: 4 [5.82 TiB] / in use: 4 [5.82 TiB] / in no VG: 0 [0]
```
Points of Configuration: Create DWS pool

- Create a storage pool with dwcli
- Pools must have a granularity of at least 16MiB
- Nodes can only belong to pools if the node allocation granularity (dwstat nodes) is a factor of the pool granularity
- **Large granularity**
  - Less sharing & interference
  - Less bandwidth OR more capacity waste
- **Small granularity**
  - More bandwidth potential
  - More interference potential
  - Less capacity waste
  - Server crash will impact more servers
Pool Size Recommendations

- **Recommendations**
  - Turn `equalize.fragments` on (default as of 6.0.UP05)
  - Pool granularity should be as small as possible, usually 16MiB
  - Pools should consist of nodes that are all the same size, performance
  - If you must mix nodes in a pool with different node allocation granularities, calculate LCM(16MiB, node1 alloc gran, node2 alloc gran, …) and use that

- **Can’t turn `equalize.fragments` on?**
  - Performance will suffer
  - Use `dwpoolhelp` tool to assist
dwcli create pool

```
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

crayadm@login> dwstat pools
  pool units quantity    free  gran
wlm_pool bytes        0       0 16MiB
```
Points of Configuration: Put nodes in to pool

- Find server nodes with dwstat nodes
- Put server nodes into pool with dwcli

```bash
 crayadm@login> module load dws
 crayadm@login> dwcli update node --name dwnode --pool wlm_pool
 update request for nodes entity with name = dwnode accepted,
 "dwstat nodes" for status
 crayadm@login> dwstat pools
 pool units quantity free gran
 wlm_pool bytes 5.82TiB 5.82TiB 16MiB
```
Nodes in a Pool

- **1TiB allocation granularity**
  - This is very high, closer to 16MiB is recommended

- **Depending on your allocation granularity, you can waste space**
  - 0.4TiB per node wasted here
Updating DataWarp Configuration Files

- Persistent changes should be made through configurator
- Immediate, one-time changes can be made to .yaml files directly
  - Then send SIGHUP or `systemctl reload dwsd/dwmd/dwrest`
  - Syntax errors will NOT cause daemons to crash or abort, but they will complain in the log file
- api-gw:/etc/opt/cray/dws/dwrest.yaml
- sdb:/etc/opt/cray/dws/dwsd.yaml
- ssd-node:/etc/opt/cray/dws/dwmd.yaml
Interesting dwsd.yaml Options

- `scratch_limit_action`, `cache_limit_action`: controls what to do when SSD excessive writes detected
  - Do nothing, log only, error only, log and error
- Set the following to 0 to disable the SSD write protection by default
  - `scratch_namespace_max_files_default`
  - `scratch_namespace_max_file_size_default`
  - `cache_max_file_size_default`
  - `instance_write_window_length_default`
  - `instance_write_window_multiplier_default`
- Change DVS stripe size with `scratch_stripe_size`
  - Default of 8388608 bytes
- Change DWFS substripe size with `scratch_substripe_size`
  - Default of 8388608 bytes
- Change DWFS substripe width with `scratch_substripe_width`
  - Default of 12 for stripe
  - Default of 1 for private
Interesting dwmd.yaml options

- 61 options in CLE 6.0UP04, all with brief descriptions
  - Majority are un-interesting path-related configuration options
- **dvs_mnt_opt**: custom options for DVS client mounts
  - dvs_scratch_mnt_opt: scratch only
  - dvs_cache_mnt_opt: cache only
- **dwfs_mnt_opt**: custom options for all DWFS mounts
  - dwfs_scratch_mnt_opt: scratch only
  - dwfs_cache_mnt_opt: cache only
- **dcfs_mnt_opt**: custom options for all DCFS mounts
- **log_mask**: enable extra dwmd logging
- **rscript_debug**: enable extra dws*.py debug logging
- **debug_flag**: developer knob
  - 0x1: Dump child task table
  - 0x2: SIGCHLD related messages
  - 0x4: Heartbeat related messages
self.dflags = {
    'dws_device_health': 0,
    'dws_n2rns': 1,
    'dws_n2slb': 2,
    'dws_namespace': 3,
    'dws_realm_member': 4,
    'dws_realm_member_reg': 5,
    'dws_swap': 6,
    'dws_sync_tasks': 7,
    'dws_util': 8,
    'lvm_fragment': 9,
    'lvm_info': 10,
    'dws_sync_tasks_dwfs2': 11,
    # insert new script here
    'test': 30,
    'all_script': 31,
    'all_debug': 32,
    'p_inputfile': 33,
    'p_map_table': 34,
    'p_input': 35,
    'p_data': 36,
    'p_path': 37,
    'p_info': 38,
    # insert new allfile flag here
    'p_tmppfile': 48,
    # insert new allfile flag here
    'p_tlock': 55,
    'p_mnt': 56,
    'level1': 57,
    'level2': 58,
    'level3': 59,
    # 60-62 are reserved for other actions
    'save_tmp': 63,
}

May 2018
rscript_debug details

- `'_input'` prints parameters
- `'_data'` prints processed input data such as json input data from request
- `'_path'` prints mount path related
- `'_info'` is some interesting data.
- `'p_map_table'` prints table data such as mount lookup table which is used for finding umount all for destroy.
- `'p_tmpfile'` prints created tmpfile data.
- `'p_tlock'` is task related debug output
- `'p_p_mnt'` is mount related data.
- `'save_tmp'` is set, dwmd will not remove any tmp input file.
- `'level3'` turns on `'p_tlock', 'p_input', 'p_data'`
- `'level2'` turns on `'p_path', 'p_mnt'`
- `'level1'` turns on `'p_info'`
Interesting dwrest.yaml options

- Some options are meant for use outside of WLM
  - user_mountroot_whitelist
- Or to protect from misuse
  - admin_mountroot_blacklist
- Grant “root like” privileges to DW functionality
  - admins
- Flexible but insecure filter for type=cache
  - cacheroot_whitelist
- Inflexible but secure filter for type=cache
  - cachemount_whitelist
Log Files & Analysis
Logging Overview

- dwsd, dwmd, dwrest log centrally to SMW with LLM
  - smw:/var/opt/cray/log/p#-<bootsession>/dws
- Log file per daemon type per day
- nginx log files stuck on internal API gateway nodes
  - Rarely needed anyway
- Data path tends to log to system console
logfile Navigation

- **nginx log file**
  - Useful for identifying if API clients can reach API gateway nodes
  - Also lists out underling API URIs

- **dwrest log file**
  - Useful in debugging staging issues

- **dwsd log file**
  - Useful to establish when objects were created, destroyed
  - Useful to track when nodes crashed, rebooted

- **dwmd log file**
  - Useful for finding out what exactly encountered difficulty
  - Tags most lines with DW object info and session token (i.e., WLM job)
Blown Fuses (a brief detour)

● The DWS will retry create/destroy operations. Persistent failures on an object, once the number of retries has exceeded, causes that object’s fuse to blow
  ● An operation will not be retried while the fuse is blown

● Blown fuses almost always means a stuck application process (for activations) or a bug (situations that lead to the inability to unmount something)

● Replace the fuse with dwcli
  ● dwcli update instance --id 12 --replace-fuse
  ● ...but unless the underlying problem is fixed, the fuse may blow again
## Blown Fuses (example)

<table>
<thead>
<tr>
<th>sess state</th>
<th>token creator owner</th>
<th>created expiration nodes</th>
<th>label public confs</th>
</tr>
</thead>
<tbody>
<tr>
<td>40820 CA---</td>
<td>11808584 SLURM 43874 2018-04-24T14:05:52</td>
<td>never 128</td>
<td></td>
</tr>
<tr>
<td>40823 D----</td>
<td>11823562 SLURM 62716 2018-04-24T15:10:39</td>
<td>never 0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>inst state</th>
<th>sess</th>
<th>bytes</th>
<th>nodes</th>
<th>created expiration intact</th>
<th>label public confs</th>
</tr>
</thead>
<tbody>
<tr>
<td>22224 CA---</td>
<td>40820</td>
<td>12.75TiB</td>
<td>162</td>
<td>2018-04-24T14:05:52 never intact</td>
<td>I40820-0 private 1</td>
</tr>
<tr>
<td>22226 D---M</td>
<td>40823</td>
<td>402.81GiB</td>
<td>5</td>
<td>2018-04-24T15:10:39 never partial</td>
<td>I40823-0 private 1</td>
</tr>
<tr>
<td>22227 CA---</td>
<td>40825</td>
<td>382.67GiB</td>
<td>19</td>
<td>2018-04-24T16:02:39 never intact</td>
<td>dw_dpaul4 public 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>conf state</th>
<th>inst</th>
<th>type</th>
<th>activs</th>
</tr>
</thead>
<tbody>
<tr>
<td>22603 CA---</td>
<td>22224</td>
<td>scratch</td>
<td>1</td>
</tr>
<tr>
<td>22605 D---M</td>
<td>22226</td>
<td>scratch</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>reg state</th>
<th>sess</th>
<th>conf</th>
<th>wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>39609 CA---</td>
<td>40820</td>
<td>22603</td>
<td>wait</td>
</tr>
<tr>
<td>39612 D---F</td>
<td>40823</td>
<td>22605</td>
<td>wait</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>activ state</th>
<th>sess</th>
<th>conf</th>
<th>nodes</th>
<th>ccache</th>
<th>mount</th>
</tr>
</thead>
<tbody>
<tr>
<td>38802 CA---</td>
<td>40820</td>
<td>22603</td>
<td>128</td>
<td>no</td>
<td>/var/opt/cray/dws/mounts/batch/11808584_striped_scratch</td>
</tr>
<tr>
<td>38806 CA---</td>
<td>40824</td>
<td>22571</td>
<td>4</td>
<td>no</td>
<td>/var/opt/cray/dws/mounts/batch/hhuhun_PR_11824107_striped_scratch</td>
</tr>
<tr>
<td>38807 CA---</td>
<td>40812</td>
<td>22571</td>
<td>64</td>
<td>no</td>
<td>/var/opt/cray/dws/mounts/batch/hhuhun_PR_11812915_striped_scratch</td>
</tr>
</tbody>
</table>
Why Did the Fuse Blow?

- Relatively straightforward in CLE 6.0.UP00 and higher
  - Tedious in prior releases (sorry)
  - dwmd log file tagging (next)
- Knowing why a fuse blew does not necessarily mean you can prevent it from happening again
  - Sorry, but you probably have to file a bug with Cray
- Especially on teardown, sometimes you just have to reboot nodes
  - But you don’t necessarily have to reboot right away!
  - Depending on what is stuck, you may just not be able to access all of DW space until the issue clears up
dwmd log file tagging

- dwmd LLM log file general format is
  - LLM prefix + <task id> + [hostname]: + (tags) + message
- LLM prefix: rfc5424 format
- <task id>: identifier logged in dwsd log
- [hostname]: on which node the message originates
- (tags): object id, session id, session token (i.e., batch job id)
- message: the actual error or success message
dwmd log file example

● **LLM prefix** + `<task id>` + `[hostname]`: + (tags) + message
● This message emitted for task id 681
● nid00350 generated the message
● Message concerns **configuration 28, session 27, with session token 32236** (i.e., batch job id)
● **Takeaway** - can search single dwmd log file for batch job id to more quickly identify certain DataWarp issues associated with the batch job
Why [hostname] is needed

- dws*.py may execute on nodes other than dwmd
Interactive Example

- As time permits
Break (back in 30!)
Slurm & DataWarp
Architecture of Slurm on Cori

Linux Slurm build
- queue1
  - slurmctld (HA)
  - slurmdbd
  - mysql
- elogin
- elogin
- elogin
- external

Native Cray Slurm build
- ctl1
  - slurmctld
- login/mom
  - salloc/srun
  - dw web services
- boot
  - aeil
- sdb
  - ncmd
  - appterm
- SMW
  - xtremoted

XC40 / HSN

User accessible, ssh, all network fs, sssd
Limited user access, require job, limited ssh, all network fs, sssd
No direct user access, restricted ssh, all network fs, sssd
No user access, root-only ssh, no network fs, restricted user database

Warning: this is a vastly simplified view of cori slurm deployment and map of interactions

Courtesy D. Jacobsen
Slurm configuration for DataWarp (very simple)

- `slurm.conf`: BurstBufferType=`burst_buffer/cray`
- `burst_buffer.conf`:
  - **DefaultPool**: name of the pool used by default for resource allocations
    - `wlm_pool`
  - **AltPoolName**: allows for different storage configurations (ex. Granularity size)
  - **DenyUsers**: list of user names and/or IDs prevented from using burst buffers
  - **Flags EnablePersistent**: allows users to create/destroy persistent burst buffers
  - **Flags TeardownFailure**: remove DW allocation on job failure

- **QoS/TRES** – control user access, user quotas, usage and report them
DWS’ dwcli vs. Slurm (one session)

# dwcli –j ls session
"created": 1473889069,
"creator": "CLI",
"expiration": 0,
"expired": false,
"id": 9711,
"links": {
   "client_nodes": []
"owner": 95448,
"state": {
   "actualized": true,
   "fuse_blown": false,
   "goal": "create",
   "mixed": false,
   "transitioning": false
"token": "tractorD"

# scontrol show burst | grep dpaul
Name=tractorD CreateTime=2016-09-14T14:37:49 Pool=wlm_pool Size=7200G State=allocated UserID= dpaul(95448)
# scontrol show burst

Name=cray DefaultPool=wlm_pool Granularity=80G TotalSpace=765600G UsedSpace=50400G
AltPoolName[0]=tr_cache Granularity=16M TotalSpace=61047200M UsedSpace=6842000M
Flags=EnablePersistent,TeardownFailure
StageInTimeout=86400 StageOutTimeout=86400 ValidateTimeout=5 OtherTimeout=300
GetSysState=/opt/cray/dw_wlm/default/bin/dw_wlm_cli

Allocated Buffers:
Name=udabb CreateTime=2018-04-28T13:33:26 Pool=wlm_pool Size=10400G State=allocated UserID=dgh(93131)
Name=rfmip_modat CreateTime=2018-04-30T21:18:23 Pool=wlm_pool Size=12400G State=allocated UserID=dfeld(96837)
Name=dpaul_tr CreateTime=2018-04-22T12:38:59 Pool=tr_cache Size=800G State=allocated UserID=dpaul(95448)
JobID=0_0(2793398) CreateTime=2018-04-31T00:28:50 Pool=(null) Size=0 State=allocated UserID=dfeld(96837)
JobID=2971140 CreateTime=2018-05-09T14:10:26 Pool=wlm_pool Size=1200G State=teardown UserID=kim(97002)

Per User Buffer Use:
UserID=dgh(93131) Used=10400G
UserID=dfeld(96837) Used=12400G
UserID=dpaul(95448) Used=800G
UserID=kim(91002) Used=1200G
DWS dwstat (administrator focused)

# dwstat most

<table>
<thead>
<tr>
<th>pool</th>
<th>units</th>
<th>quantity</th>
<th>free</th>
<th>gran</th>
</tr>
</thead>
<tbody>
<tr>
<td>tr_cache bytes</td>
<td>5.82TiB</td>
<td>5.82TiB</td>
<td>16MiB</td>
<td></td>
</tr>
<tr>
<td>wlm_pool bytes</td>
<td>809.96TiB</td>
<td>627.34TiB</td>
<td>200GiB</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sess</th>
<th>state</th>
<th>token</th>
<th>creator</th>
<th>owner</th>
<th>created</th>
<th>expiration</th>
<th>nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>9708</td>
<td>CA---</td>
<td>2993022</td>
<td>SLURM</td>
<td>90891</td>
<td>2016-09-14T14:27:48</td>
<td>never</td>
<td>8</td>
</tr>
<tr>
<td>9710</td>
<td>CA---</td>
<td>tractorD</td>
<td>CLI</td>
<td>95448</td>
<td>2016-09-14T14:31:43</td>
<td>never</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>inst</th>
<th>state</th>
<th>sess</th>
<th>bytes</th>
<th>nodes</th>
<th>created</th>
<th>expiration</th>
<th>intact</th>
<th>label</th>
<th>public</th>
<th>confs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1943</td>
<td>CA---</td>
<td>9708</td>
<td>27.73TiB</td>
<td>142</td>
<td>2016-09-14T14:27:48</td>
<td>never</td>
<td>true</td>
<td>I9708-0</td>
<td>false</td>
<td>1</td>
</tr>
<tr>
<td>1945</td>
<td>CA---</td>
<td>9710</td>
<td>27.73TiB</td>
<td>142</td>
<td>2016-09-14T14:31:43</td>
<td>never</td>
<td>true</td>
<td>tractorD</td>
<td>true</td>
<td>1</td>
</tr>
</tbody>
</table>
Using Datawarp without Slurm

$ dwcli create session --expiration 4000000000 --creator $(id -un) --token example session --owner $(id -u) --hosts example node created session id 10

$ dwcli create instance --expiration 4000000000 --public --session 10 --pool example-poolname --capacity 1099511627776 --label example-instance --optimization bandwidth created instance id 8

$ dwcli create configuration --type scratch --access-type stripe --root permissions 0755 --instance 8 --group 513 created configuration id 7

$ create activation --mount /some/pfs/mount/directory --configuration 7 --session 10 created activation id 7

NERSC
#!/bin/bash

#SBATCH -n 32 -t 2

#DW jobdw type=scratch access_mode=striped capacity=1TiB

#DW stage_in type=directory source=/lustre/my_in_dir destination=$DW_JOB_STRIPED

#DW stage_out type=directory destination=/lustre/my_out_dir source=$DW_JOB_STRIPED

export JOBDIR=$DW_JOB_STRIPED

cd $DW_JOB_STRIPED

srun –n 32 a.out
User Library example - libdatawarp

// module load datawarp (to get access to the user library for building)  
#include <datawarp.h>

// Get Info on DataWarp Configuration:  
int r = dw_get_stripe_configuration(fd, &stripe_size, &stripe_width, &stripe_index);

// Use dw_stage_file_in function to move a file from PFS to DataWarp   
int r = dw_stage_file_in(dw_file, pfs_file);

// Use dw_stage_file_out function to move a file from DataWarp to PFS   
int r = dw_stage_file_out(dw_file, pfs_file, DW_STAGE_IMMEDIATE);

// Use dw_query_file_stage function to check stage in/out completion  
int r = dw_query_file_stage(dw_file, &complete, &pending, &deferred, &failed);
#!/bin/bash
#SBATCH -p debug
#SBATCH -N 1
#SBATCH -t 00:01:00

( Create a Persistent Reservation/Allocation (PR) )
#BB create_persistent name=tractorD capacity=7TB access=striped type=scratch
exit

( Specify PR for a subsequent job - #sbatch omitted)
#DW persistentdw name=tractorD

( Copy in data in for the job)
#DW stage_in source=/global/cscratch1/sd/dpaul/decam.tar destination=$DW_PERSISTENT_STRIPED_tractorD/job1/runit.sh
  type=file

#DW stage_in source=/global/cscratch1/sd/dpaul/src_dir destination=$DW_PERSISTENT_STRIPED_tractorD/job1/
  type=directory

(continued)
Persistent Reservation/Allocation (PR) cont.

( Run the job )

cd $DW_PERSISTENT_STRIPED_tractorD/job1/
srun runit.sh < src_dir > output_dir

( Save results at job completion – here for clarity, must be at top of script & contiguous )

#DW stage_out source=$DW_PERSISTENT_STRIPED_tractorD/job1/output_dir
destination=/global/cscratch1/sd/dpaul/job1/ type=directory
Transparent Cache features

• BurstBuffer will be used as filesystem cache for all I/O to/from the PFS:

```bash
#DW jobdw pfs=/global/cscratch1/sd/dpaul/job_output/ capacity=800GB type=cache access_mode=striped pool=wlm_pool
```
Log legend

- rid: registration ID
- sid: session ID
- stoken: session token ID (aka jobID)
- aid: activation ID
DataWarp creation process outputs

```bash
===> cat PR_test.sh
#!/bin/bash
#SBATCH --partition=debug
#SBATCH --time=5:00
#SBATCH -C haswell
#BB create_persistent name=DW_TEST capacity=90GB access_mode=striped type=scratch pool=wlm_pool
#DW persistentdw name=DW_TEST

===> sbatch PR_test.sh
```

```
JOBID PARTITION    NAME   USER  STATE  TIME  TIME_LIMI  NODES Nodelist(REASON)
941398 debug PR_test_  dpaul  RUNNING  0:22  5:00    1 nid000021
```
Creation process outputs (slurmctld)

slurmctld:


[2018-05-07T11:18:04.964] sched: Allocate JobID=941398 NodeList=nid000021 #CPUs=64 Partition=debug
[2018-05-07T11:18:06.564] _start_pre_run: dws_pre_run for JobID=941398 ran for usec=1600110

[2018-05-07T11:18:06.564] Activation 154 created for configuration 22 and session 215

[2018-05-07T11:18:06.564] prolog_running_decr: Configuration for JobID=941398 is complete
[2018-05-07T11:18:06.564] Extending job 941398 time limit by 2 secs for configuration
Creation process outputs (dwmd)

dwmd:
2018-05-07 11:18:01 (31530): <Z942> task namespace_create task_id 2942 DONE: ret=0
2018-05-07 11:18:05 (14839): <1> TCP connection from sdb:55776
2018-05-07 11:18:05 (14839): <1> Backgrounding as socket 10
2018-05-07 11:18:05 (14839): <1> method_async_task: task dwfs_realm_member_registration_create task_id 2943 host sdb-i
10
t: /var/opt/cray/dws/mounts/realm-member/33, /var/opt/cray/dws/mounts/registrations/166
t: /var/opt/cray/dws/mounts/realm-member/34, /var/opt/cray/dws/mounts/registrations/166
2018-05-07 11:18:05 (31544): <2943> Resumed get message for fd=11 #1

2018-05-07 11:18:05 (31544): <2943> task dwfs_realm_member_registration_create task_id 2943 DONE: ret=0
2018-05-07 11:18:05 (14839): <1> Backgrounding as socket 10
2018-05-07 11:18:05 (14839): <1> method_async_task: task node_to_registered_namespace_create task_id 2944 host sdb-ipo
gif0-1 port 2015
opt/cray/dws/mounts/registrations/166 /var/opt/cray/dws/mounts/n2rns/273 -o path=/var/opt/cray/dws/mounts/n2rns/273,no
defile=/tmp/tmpetfk_D.maxnodes=2,blksize=8388608,dvfs,deferopens,mds=0-0c0s6n1,attrcache_timeout=3,nouserenv,multifsy
nc,parallelwrite,nocache
cray/dws/mounts/n2rns/273/22, /var/opt/cray/dws/mounts/batch/DW_Test_941398_striped_scratch
2018-05-07 11:18:06 (31557): <2944> Resumed get message for fd=11 #1
Creation process outputs (dwmd cont.)

```
2018-05-07 11:18:06 (31557): <2944> task node_to_registered_namespace_create task_id 2944 DONE: ret=0
2018-05-07 11:20:07 (14839): <1> Backgrounding as socket 10
2018-05-07 11:20:07 (14839): <1> method_async_task: task node_to_registered_namespace_destroy task_id 2945 host sdb-ipogf0-1 port 2015
2018-05-07 11:20:08 (31625): <2945> Resumed get message for fd=11 #1
2018-05-07 11:20:08 (31625): <2945> task node_to_registered_namespace_destroy task_id 2945 DONE: ret=0
2018-05-07 11:20:08 (14839): <1> TCP connection from sdb:55794
2018-05-07 11:20:08 (14839): <1> Backgrounding as socket 10
2018-05-07 11:20:09 (31630): <2946> Resumed get message for fd=11 #1
```
Creation process outputs (dwmd cont.)

2018-05-07 11:20:09 (31630): <2946> task namespace_wait task_id 2946 DONE: ret=0
2018-05-07 11:20:09 (14839): <1> Back grounding as socket 10
2018-05-07 11:20:09 (14839): <1> method_async_task: task dwfs_real member registration_destroy task_id 2947 host sdb-ipogif0-1 port 2015
2018-05-07 11:20:09 (31643): <2947> Resumed get message for fd=11 #1
Creation process outputs (dwstat)

```
dwstat:
nid00025: # dwstat all

    pool units quantity free   gran
    wlm_pool bytes 11.64TiB 8.34TiB 80.56GiB

    sess state   token creator owner created expiration nodes
    214 CA---    DW_TEST  CLI 15448 2018-05-07T11:17:59  never  0

    inst state sess bytes nodes    created expiration intact     label public confs
    37 CA---    214 161.12GiB 2  2018-05-07T11:17:59  never  intact    DW_TEST public  1

    conf state inst  type activs
    22 CA---    37 scratch    0

    frag state inst capacity   node
    65 CA--    37 80.56GiB nid00025
    66 CA--    37 80.56GiB nid00026

    ns state conf frag span
    22 CA--    22 65 2

    node pool online drain gran capacity insts activs
    nid00025 wlm_pool online  fill 16MiB 5.82TiB  1 0
    nid00026 wlm_pool online  fill 16MiB 5.82TiB  4 0

did not find any cache configurations, swap configurations, registrations, activations
```
dw_wlm_cli – command line use

```bash
# /opt/cray/dw_wlm/default/bin/dw_wlm_cli -f show_sessions

{"sessions": [""created": 1525846539, "creator": "CLI", "expiration": 0, "expired": false, "id": 2, "links": {"client_nodes": []}, "owner": 15448, "state": {"actualized": true, "fuse_blown": false, "goal": "create", "mixed": false, "transitioning": false}, "token": "dw_dpaul4"}, {"created": 1525847068, "creator": "CLI", "expiration": 0, "expired": false, "id": 4, "links": {"client_nodes": []}, "owner": 15448, "state": {"actualized": true, "fuse_blown": false, "goal": "create", "mixed": false, "transitioning": false}, "token": "dev_scratch"}, {"created": 1525882943, "creator": "CLI", "expiration": 0, "expired": false, "id": 6, "links": {"client_nodes": []}, "owner": 15448, "state": {"actualized": true, "fuse_blown": false, "goal": "create", "mixed": false, "transitioning": false}, "token": "dw_scratch"}, {"created": 1525896220, "creator": "CLI", "expiration": 0, "expired": false, "id": 12, "links": {"client_nodes": []}, "owner": 73143, "state": {"actualized": true, "fuse_blown": false, "goal": "create", "mixed": false, "transitioning": false}, "token": "no_WT_1985"}, {"created": 1525896221, "creator": "CLI", "expiration": 0, "expired": false, "id": 13, "links": {"client_nodes": []}, "owner": 73143, "state": {"actualized": true, "fuse_blown": false, "goal": "create", "mixed": false, "transitioning": false}, "token": "WT_1985"}, {"created": 1525905621, "creator": "CLI", "expiration": 0, "expired": false, "id": 28, "links": {"client_nodes": []}, "owner": 30821, "state": {"actualized": true, "fuse_blown": false, "goal": "create", "mixed": false, "transitioning": false}, "token": "NCBI_DB2"}, {"created": 1525924632, "creator": "CLI", "expiration": 0, "expired": false, "id": 33, "links": {"client_nodes": []}}
```
Common Problems & Solutions
SSD fails with DWS state on it

- Sometimes SSDs fail
- Since the DWS tries and retries to initiate and wait for stageout activity, it needs to be told when this is futile
- Find the relevant registrations and set them to --haste with dwcli
  - dwcli update registration --id 74 --haste
SSD Failure Detection

- In rare cases SSDs have failed in a way that has locked up XFS and DVS
  - This results in node health marking compute nodes admindown
- DataWarp Service now attempts to *detect* failing SSDs
- Upon detection, dwmd will *intentionally* panic a DW server node
  - This allows processes to do some cleanup so compute nodes do not go admindown
- False positives are possible. Be suspicious of hardware!

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

- Sometimes a blade needs servicing
- Server nodes set to drain with dwcli will not be used in new instance creations
  - `dwcli update node --name nid00350 --drain`
- Be sure to set both nodes on a blade to drain
- Wait for instance count on nodes to hit 0 to minimize disrupting existing usages of DataWarp (`dwstat nodes`)
  - May need to remove persistent instances
- When maintenance completes, unset drain state
  - `dwcli update node --name nid00350 --fill`
The Tale of Two dwcaches

- If node A has SSDs 1 and 2 and node B has SSDs 3 and 4, after maintenance node A may end up with 1 and 3 and node B with 2 and 4
- LVM Physical Volume headers for SSD 1 has information on VG dwcache, but SSD 3 does too (but it’s a different dwcache!)
- LVM is smart enough to know that they are really different VGs, but you’ll still see two dwcaches on each node
- Swap the hardware to fix or just re-initialize the SSDs with LVM
#DW stage_in/out failures with lots of files

- #DW stage_in/out where target has thousands of files can fail due to hitting timeouts
- **Short term**: increase the timeouts (which were always way too short)
  - DW admin guide includes instructions for using site-local ansible play to bump timeouts
  - Timeout bumps are for nginx and dwrest
- **Long term**: moving to new API that removes need for increasing timeouts, improves error messages
User reports of unexplained stage failures

- Batch job #DW stage_in/out can fail
  - System issue
    - Your PFS must be mounted on DW servers!
  - Typo in job script
- DataWarp Service does not give good error messages today
  - For Slurm, 'squeue -l -u username' will show an 'offline namespaces' error
- You must look in dwmd.log for clues!
  - Search for batch job id or udwfs_stage
- Long term: moving to new underlying staging API that improves error messages
Example outputs:

```
[2018-03-19T11:42:20.840] DataWarp REST API error: offline namespaces: [16] - \ ask a system administrator to consult the dwmd log for more information

# grep udwfs dwmd-20180510
<150>1 2018-05-10T01:09:34.098988-07:00 c6-4c0s4n2 dwmd 31377 p0-20180508t175601
[dws@34] <0> [nid10386]: dws_sync_tasks ERROR: __udwfs_stage_dir_out failed (Host is down)
ns_id=27 dwfs_id=4911 pfs_dir_path=/global/cscratch1/sd/tshep/bboutputs/GCM/1999/WT/.
dw_dir_path=/var/opt/cray/dws-mounts/realm-member/4911/27/runs/
stage_type=DW_STAGE_IMMEDIATE /proc/fs/udwfs-mounts/4911/label=
<150>1 2018-05-10T02:13:04.866419-07:00 c3-0c0s3n2 dwmd 1266 p0-20180508t175601
[dws@34] <1339> [nid00590]: (cid:30,sid:38,token:12264331) dws_namespace
WARN:udwfs destroy nsid=30 got EHOSTDOWN
```
User reports of unexplained IO errors

- SSD write protection will return one of three errno once activated
  - EROFS (write window exceeded)
  - EMFILE (maximum files created exceeded)
  - EFBIG (maximum file size exceeded)
- Log messages are emitted to the console log
- SEC rule looks for these and can take site-configured action
- If many users hit these, consider
  - Educating user base on SSD write limits
  - Raising the defaults to decrease false positives
  - Turn the functionality off
Example: ‘stuck umount’
Stuck Session

● Sometimes a session just won’t go away
● This USUALLY means any of:
  ● a registration cannot make forward progress; set --haste if that’s acceptable
  ● a fuse has blown
  ● a process is stuck
● There is no “force remove” option in DWS because while it would clear up status displays, it wouldn’t actually fix the problem
● How to fix? Case-by-case basis
  ● Restart daemons (especially dwmd)
  ● Reboot nodes
  ● Hunt down and kill stuck processes
  ● Replace fuse
Checking on DataWarp Health

- Especially after a system reboot, any of the previously mentioned hardware issues may arise
- New software updates may also introduce issues
- `datawarp_check.py` - basic DataWarp health check script
Back up DataWarp State

- Some DataWarp state (pools, drain state, node-pool association) can be backed up and restored
- State restoration necessary when…
  - Updating to a new release of CLE with backwards-incompatible changes
  - Rarely, dwsd database corruption
- `dwcli config backup` 
  - Saves data via RESTful API
- `dwbackup` 
  - Extracts data directly from dwsd database
  - Necessary if "backing up" after backwards-incompatible change
Backup examples

```
# dwcli method
crayadm@login> module load dws
crayadm@login> dwcli config backup >/home/crayadm/dw.json

# dwbackup method
sdb# module load dws
sdb# dwbackup >/home/crayadm/dw.json
```
Restore DataWarp state

- Saved DataWarp state can be restored at any time
- `dwcli config restore </home/crayadm/dw.json`
  - Can be run multiple times
  - If nodes are missing, you'll get a warning but can run the command later when the node boots

```
crayadm@login> module load dws
crayadm@login> dwcli config restore </home/crayadm/dw.json
pool check progress [==============] 1/1 100% done
node update progress [==============] 2/2 100% done
```
Tools for DataWarp System Administration
Tips and Tricks

- Use `pdsh` + `dshbak` to perform DW tasks in parallel

```bash
boot# N=$(ssh smw cfgset get \n> cray_node_groups.settings.groups.data.datawarp_nodes.members p0 | tr \n'\n' ','
) 
boot# pdsh -w $N 'lsblk -d' | dshbak -c
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>MAJ:MIN</th>
<th>RM</th>
<th>SIZE</th>
<th>RO</th>
<th>MOUNTPOINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>nvme0n1</td>
<td>254:0</td>
<td>0</td>
<td>1.5T</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>nvme1n1</td>
<td>254:64</td>
<td>0</td>
<td>1.5T</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>nvme2n1</td>
<td>254:128</td>
<td>0</td>
<td>1.5T</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>nvme3n1</td>
<td>254:192</td>
<td>0</td>
<td>1.5T</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>MAJ:MIN</th>
<th>RM</th>
<th>SIZE</th>
<th>RO</th>
<th>MOUNTPOINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>nvme0n1</td>
<td>254:0</td>
<td>0</td>
<td>2.9T</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>nvme1n1</td>
<td>254:64</td>
<td>0</td>
<td>2.9T</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
jq

- In CLE as of 6.0.UP05
- “sed for json”
- DW RESTful API is stable, better for scripting
- Find instances with blown fuse
  - `dwcli -j ls instances | jq -rS '.instances[] | select(.state.fuse_blown == true) | .id'`
- Select server nodes
  - `dwcli -j ls nodes | jq -rS '.nodes[] | select(.online == false and .capacity > 0) | .id'`
- Can use with dwcli actions to operate on multiple objects
datawarp_check.py

● Simple script for checking on DataWarp health
  ● Excludes WLM layer
● Useful to run after system boots
● Contact Cray support for a copy

```
crayadm@login> ./datawarp_check.py
v3 2017-06-26
...  
PASS created session 1  
PASS created instance 1  
PASS created configuration 1  
PASS created activation 1  
...  
Session 1 is now deleted
```
Libhio Test Suite

- LANL-developed parallel IO package
- [https://github.com/hpc/libhio](https://github.com/hpc/libhio)
- Includes tests that run on DataWarp through WLMs
  - Fantastic sanity check on DataWarp and WLM integration
  - Varies DW allocation size, compute node count, IO pattern, etc
  - Each test outputs performance information
- Supports Slurm and Moab/TORQUE as WLM
KAUST DataWarp Regression Suite

- Written by Georgios Markomanolis at KAUST
- [https://github.com/gmarkomanolis/datawarp_regression](https://github.com/gmarkomanolis/datawarp_regression)

**Test coverage**
- IOR runs
- Stage in, stage out
  - Files and folders
- Persistent instances
- libdatawarp API

**Supports Slurm as WLM**
NERSC *bbcheck* utility

- dwstat wrapper script (python)
- Created by the Operations Technology Group
  - Basil Lalli, Tony Quan, John Gann
- Much easier to identify the pieces involved in a failure
- 30 minute *snapshots* with cron (for debugging)
NERSC *bbcheck* utility

```
corismw:~ # ~crayadm/bin/bbcheck -h
usage: bbcheck [-acmp]
Options and arguments:
  -a   : prints all sessions, not just those in error. Ignored if used with -c
  -c   : prints BB information in the same format as DWSTAT, rather than as a hierarchy
  -m   : prints data in monochrome, rather than color. Useful to keep terminal escape sequences out of data.
  -p   : attempts to report information about bad processes on fragments that are reporting problems

corismw:~ #
```
```
corismw:~ # ~crayadm/bin/bbcheck
The following nodes are drained:

<table>
<thead>
<tr>
<th>node</th>
<th>pool</th>
<th>online</th>
<th>drain</th>
<th>gran</th>
<th>capacity</th>
<th>insts</th>
<th>activs</th>
</tr>
</thead>
<tbody>
<tr>
<td>nid00206</td>
<td>wlm_pool</td>
<td>online</td>
<td>drain</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nid05581</td>
<td>wlm_pool</td>
<td>online</td>
<td>drain</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>nid05582</td>
<td>wlm_pool</td>
<td>online</td>
<td>drain</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>nid05709</td>
<td>wlm_pool</td>
<td>online</td>
<td>drain</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>nid05710</td>
<td>wlm_pool</td>
<td>online</td>
<td>drain</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>nid06477</td>
<td>wlm_pool</td>
<td>online</td>
<td>drain</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nid06478</td>
<td>wlm_pool</td>
<td>online</td>
<td>drain</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nid07437</td>
<td>wlm_pool</td>
<td>online</td>
<td>drain</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nid09166</td>
<td>wlm_pool</td>
<td>online</td>
<td>drain</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

There are no offline nodes

The following nodes are not in appropriate pools:

<table>
<thead>
<tr>
<th>node</th>
<th>pool</th>
<th>online</th>
<th>drain</th>
<th>gran</th>
<th>capacity</th>
<th>insts</th>
<th>activs</th>
</tr>
</thead>
<tbody>
<tr>
<td>nid11341</td>
<td>-</td>
<td>online</td>
<td>fill</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nid11342</td>
<td>-</td>
<td>online</td>
<td>fill</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nid11409</td>
<td>-</td>
<td>online</td>
<td>fill</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nid11410</td>
<td>-</td>
<td>online</td>
<td>fill</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nid11469</td>
<td>-</td>
<td>online</td>
<td>fill</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nid11470</td>
<td>-</td>
<td>online</td>
<td>fill</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```
# bbcheck -a

<table>
<thead>
<tr>
<th>Session ID</th>
<th>State</th>
<th>UID</th>
<th>Nodes</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>CA</td>
<td>15448</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This session contains the following instance:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Instance ID: 2  State: CA  MDS Node: nid11598/c0-5c1s3n2/bb272  Size: 0.38266 TiB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>State of this instance's fragments</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Fragments 46-64 are OK!</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This instance contains the following configuration:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Configuration ID: 2  State: CA</td>
</tr>
<tr>
<td>4</td>
<td>CA</td>
<td>15448</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This session contains the following instance:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Instance ID: 3  State: CA  MDS Node: nid12110/c3-5c0s3n2/bb284  Size: 37.38096 TiB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>State of this instance's fragments</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Fragments 65-72 are OK!</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This instance contains the following configuration:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Configuration ID: 3  State: CA</td>
</tr>
<tr>
<td>6</td>
<td>CA</td>
<td>15448</td>
<td>261</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This session contains the following instance:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Instance ID: 5  State: CA  MDS Node: nid03214/c4-1c2s3n2/bb60  Size: 42.05232 TiB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>State of this instance's fragments</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Fragments 74-334 are OK!</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This instance contains the following configuration:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Configuration ID: 5  State: CA</td>
</tr>
<tr>
<td>12</td>
<td>CA</td>
<td>73143</td>
<td>116</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This session contains the following instance:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Instance ID: 8  State: CA  MDS Node: nid02830/c2-1c2s3n2/bb50  Size: 2.33624 TiB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>State of this instance's fragments</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Fragments 857-972 are OK!</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This instance contains the following configuration:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Configuration ID: 8  State: CA</td>
</tr>
</tbody>
</table>
There are no offline nodes.

The following nodes are not in appropriate Pools:

<table>
<thead>
<tr>
<th>node</th>
<th>pool</th>
<th>online</th>
<th>drain</th>
<th>gran</th>
<th>capacity</th>
<th>insts</th>
<th>activs</th>
</tr>
</thead>
<tbody>
<tr>
<td>nid11341</td>
<td>online fill</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nid11342</td>
<td>online fill</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nid11409</td>
<td>online fill</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nid11410</td>
<td>online fill</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nid11469</td>
<td>online fill</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nid11470</td>
<td>online fill</td>
<td>0.02GiB</td>
<td>5961.64GiB</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There are no namespaces in a bad state.

There are no instances in a bad state.

There are no activations in a bad state.

There are no sessions in a bad state.

There are no fragments in a bad state.

There are no nodes in a bad state.

There are no configurations in a bad state.

There are no registrations in a bad state.
Q&A

David Paul – dpaul@lbl.gov
Benjamin Landsteiner – ben@cray.com
you folks understand this now right???