Driving More Efficient Workload Management on Cray Systems with PBS Professional

Graham Russell
Technical Director Enterprise Computing
May 2016
PBS Professional
Architected for Cray Systems
Re-architected Cray Support

Flexible PBS Server/Scheduler

• PBS Server/Scheduler can exist outside the Cray environment
• Utilize PBS Professional High Availability
• Impose per-named user/group/project limits
• Manage Cray compute nodes & Repurposed Compute/MAMU Nodes
• Manage Cray XC & Cray CS systems
Re-architected Cray Support

Full “PBS vnode” features supported for Cray

- MPI task selection and placement – seamless transition between Cray and clusters with select & place language
- See which compute nodes the job is running on via qstat
- See each compute node in via pbsnodes
- Topology-aware scheduling
  Grouping & placement set framework
- More robust reservations
- Plus numerous speed improvements
Topology-aware Scheduling

Speeds Application Performance and Boosts Utilization

Topology represented by “placement sets”

- Both inter (clusters, switches, grids) and intra (NUMA) node topology
- Infiniband, Ethernet, custom networks -- tree, torus, hypercube, dragonfly, anything
- Easily updated without the need to restart PBS

PBS Professional

*illustrations from HPC-Opinion.blogspot.com
Tunable Scheduling Formula

Define any policy – including on-the-fly “exceptions”

Simple formulas are very simple (big jobs go first)

\[ ncpus \times \left( \frac{walltime}{3600.0} \right) \]

Complex formulas are pretty simple too… (adds priority accrual for smaller jobs, high-priority queue, deferred queue, “run this job next”)

\[ \left( ncpus \times \left( \frac{walltime}{3600.0} \right) \right) \times Wsize + \left( \frac{eligible\_time}{3600.0} \right) \times Wwait + \text{special\_p} \]
### Estimated Job Start Times

% qstat -T

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Username</th>
<th>Queue</th>
<th>Jobname</th>
<th>SessID</th>
<th>NDS</th>
<th>TSK</th>
<th>Memory</th>
<th>Time</th>
<th>Req'd</th>
<th>Req'd</th>
<th>Est Start</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.quark bill</td>
<td>workq</td>
<td>foo</td>
<td>12345</td>
<td>1</td>
<td>1</td>
<td>128mb</td>
<td>00:10</td>
<td>R</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.quark bill</td>
<td>workq</td>
<td>bar</td>
<td>--</td>
<td>1</td>
<td>1</td>
<td>128mb</td>
<td>00:10</td>
<td>Q</td>
<td>11:30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.quark bill</td>
<td>workq</td>
<td>gril</td>
<td>--</td>
<td>1</td>
<td>1</td>
<td>128mb</td>
<td>00:10</td>
<td>Q</td>
<td>11:40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.quark bill</td>
<td>workq</td>
<td>baz</td>
<td>--</td>
<td>1</td>
<td>1</td>
<td>128mb</td>
<td>00:10</td>
<td>Q</td>
<td>Tu 18</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

% qstat -f

```plaintext
...  
estimated.exec_vnode = (pepsi:ncpus=1)  
estimated.start_time = Fri Apr 26 11:52:44 2013
```
PBS Professional
Recent Enhancements for Cray
EXCLUSIVE vs non-EXCLUSIVE Reservation Mode

• EXCLUSIVE mode has been used
  • Only one ALPS reservation can be made on nodes; special privileges
  • Previously, PBS created all ALPS reservations as EXCLUSIVE

• non-EXCLUSIVE mode does not mean "shared"
  • Refer to Cray documentation for more information on EXCLUSIVE ALPS reservations.
  • Allows PBS Pro to suspend/resume jobs

• PBS allows EXCLUSIVE or non-EXCLUSIVE to be user selectable
  • non-EXCLUSIVE is default

• To request an EXCLUSIVE ALPS reservation, submit a job with "-l place=excl"
  • EXCLUSIVE on the Cray is not the same as the PBS place=excl

• Examples of when jobs may require EXCLUSIVE reservations
  • aprun -sl/-sn/-S
  • aprun -F exclusive
  • requesting network performance counters
Suspend & Resume w/ CLE 5.2 and later

- **Preemption - Suspend and Resume**
  - PBS releases only the CPUs when a job is suspended
  - Cray compute nodes do not have swap space, thus memory is not released

- **Requires jobs to request non-EXCLUSIVE reservations**
  - Non-EXCLUSIVE reservations are default
  - Cray does not allow ALPS "EXCLUSIVE” reservations to be suspended

- **Configuring PBS Scheduler (PBS_HOME/sched_priv/sched_config)**
  - preemptive_sched: True All
  - preempt_order: SR
    - Don’t want to requeue jobs? Remove the ‘R’
  - kill -HUP <pbs_sched_pid>
Xeon Phi Co-Processor Support

• Automatically detects a Xeon Phi in the ALPS inventory

• PBS vnode attributes
  • accelerator_model
  • accelerator_memory

• pbsnodes output
  • resources_available.accelerator_model=Xeon_Phi
  • resources_available.accelerator_memory=4000MB

• Example: You want 30 PEs and a Xeon Phi accelerator on each host, and the accelerator should have at least 4000MB, and you don't care how many hosts the job uses

-l select=30:ncpus=1:accelerator_model="Xeon_Phi":accelerator_memory=4000MB
Hyper-Threading Support

• **PBS fully supports Cray hyper-threading**
  • Requires some additional configuration by sys admin

• **Automatic detection w/ BASIL 1.3 and later**
  • nppus
    • Total number of Physical Processing Units per vnode
  • vps_per_ppu
    • Number of virtual processors per physical processing unit
    • vps_per_ppu='1' indicates 1 virtual processor per PPU; hyperthreading off
    • vps_per_ppu='2' indicates 2 virtual processors per PPU; hyperthreading on

• **Interaction w/ batchCPCU (alps.conf)**
  • Desire hyperthreading on by default, then set batchCPCU to 0
  • Desire hyperthreading off by default, then
    • Set batchCPCU to 1
    • Set the server’s default_chunk.nppus to the number of compute units per node (# of CPUS per node / # of Threads per CPU)
      • qmgr -c “set server default_chunk.nppus=N”
Hyper-Threading Support, cont.

- **nppcu**
  - Number of Processors Per Compute Unit - Cray BASIL 1.3 attribute in the RESERVE XML for specifying how many processors of a compute unit should be used.
  - Value of ncpus / nppus

- **If nppus is requested and nppcu is not an integer**
  - PBS does not set nppcu in the ALPS reservation request.
  - PBS logs a message in the MoM log at level DEBUG3:
    - “The ratio of ncpus/nppus is not an integer; therefore it is not known how many processors per compute unit to request, so none were requested.”
Power-aware Scheduling

Green Provisioning™ – Save Power, Save Money, Save the Earth

Power capping, e.g., power\_budget = 0.5 MW
- Implemented via Cray's capmc
- Fit more hardware into your datacenter

Per-job power profiles, e.g., power = 600 W
- Implemented via Cray's capmc
- Run at optimal power for your workload

Energy accounting, e.g., energy = 64.2kWh
- Integrated with Cray's capmc & RUR
- Viewable in qstat and PBS accounting log
- Visualize via PBS Analytics

Power idle nodes on/off
- Coming soon
Power-aware Scheduling Demo

Other notable enhancements

- Allow periodic mom hook to modify job resources on "this node"
- Support config file for hooks
- Support for cgroups on Login nodes & Repurposed Compute Nodes (MAMU nodes)
- New qsub -f option keeps qsub from daemonizing itself after executing
- Improve server/MoM inventory performance; vnode_pool, which can be used to reduce the communication traffic between server and MoMs
  - Typically create all MoM nodes in same vnode_pool so only 1 node in pool reports inventory
- Periodically re-query the ALPS inventory and update PBS
PBS 13.0.401 Release

- Merge of Cray capabilities with PBS 13.0 branch
- Includes all the Cray specific functionality from 12.2.404

- Highlights
  - Increased job submission rates
  - Significantly shorter scheduling cycles
  - Reduced time to results
### PBS Pro 13.0 – Architected for Exascale

<table>
<thead>
<tr>
<th>Category</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>• 10x faster throughput</td>
</tr>
<tr>
<td>Scale</td>
<td>• 5x larger system sizes</td>
</tr>
<tr>
<td>Resilience</td>
<td>• 100% health check infrastructure</td>
</tr>
<tr>
<td>Power</td>
<td>• Profiles, caps, accounting</td>
</tr>
<tr>
<td>Scheduling</td>
<td>• Expanded priority formula, fairshare formula, and preemption targets</td>
</tr>
<tr>
<td>Plugins</td>
<td>• New hook events and extensions</td>
</tr>
<tr>
<td>More</td>
<td>• Usability, platforms, Windows, ...</td>
</tr>
</tbody>
</table>

**Big benefits for all PBS users – big and small**
PBS Pro 13.0.400: Cray Specific Features

Vnode Per NUMA Node Option

- Create vnode per compute node – improves scheduler performance

- This feature is available only for systems without accelerators e.g. GPUs and co-processors
PBS Pro 13.0.401: Cray Specific Features

- **Adjust Reservations**
  - `pbs_ralter` – Adjust Start & End Time. Can change the end time for a running reservation with running jobs

- **ASAP Reservations**
  - Fix For Server Crash

- **ALPS Sync Hook**
  - Update Inventory Sync Hook
Roadmap: PBS Pro 13.0.402: Cray Specific Features

Power Management

- Phase 2: Nodes On/Off

Release Expected: May 2016
Roadmap: PBS Pro 13.0.403: Cray Specific Features

- IMPS Installer
  - Including support for CLE 6.0
- Xeon Phi
  - LA support for KNL

Released Expected: June 2016
DataWarp Preview
New Storage Aware Scheduling Capability

- **PBS Storage Resource Manager**
- **Centralised data management**
- **Integrated with PBS Professional to allow storage and data to influence *when* and *where* PBS runs jobs**
  - Jobs are only dispatched when associated storage and data are ready
    - E.g. enough free capacity, storage device load within limits for CPU load, network load etc
    - Optimises job throughput and execution times
- **Designed to support heterogeneous storage devices**
  - Islands of storage
  - Burst buffers
  - Multiple storage devices per jobs
- **PBS no longer responsible for data staging**
  - Avoids idle compute nodes while copying data
  - Utilisation of compute nodes is increased
- **Autonomous management and monitoring of storage devices and data entities**
  - Decouples scheduling from data management
  - Monitoring of storage device resources and limits
Benefits of PBS Storage Resource Manager

• 2 data staging models
  • Just-in-time data staging to avoid filling up local storage
  • Immediate data staging, when jobs are submitted

• Job dependencies are data aware
  • Dependent jobs not dispatched until data is ready, instead of when previous job is completed
    • Support afterok, afternotok and afterany

• Enhanced user experience
  • User monitors job and data progress using enhanced version of qstat
**Architecture**

- **PBS Storage Resource Manager Daemon**
  - Forked Stage-in Processes
  - Forked Stage-out Processes

- **PBS Storage Resource Manager Interface**
- **Storage Devices** (e.g. Cray DataWarp)

- **Remote communication via socket**

- **Can be installed on cluster management node or a separate system**

- **Scripts**

- **PBS Cluster**
Example Cray DataWarp job script

```bash
#!/bin/bash

#PBS -l ncpus=128
#PBS -N test_dw_job

#DW jobw type=scratch capacity=10GB access_mode=striped,private pfs=/scratch
#DW stage_in type=directory source=/pfs/dir1 destination=$DW_JOB_STRIPED/dir1
#DW stage_in type=list source=/pfs/inlist
#DW stage_in type=file source=/pfs/file1 destination=$DW_JOB_STRIPED/file1
#DW stage_out type=directory destination=/pfs/dir1 source=$DW_JOB_STRIPED/dir1
#DW stage_out type=list source=/pfs/inlist
#DW stage_out type=file destination=/pfs/file1 source=$DW_JOB_STRIPED/file1

aprun -n 128 -N 32 $BB_JOB_PRIVATE/abc $BB_JOB_STRIPED/abc an_app
```
## Job/Data Monitoring with enhanced qstat

- `$ qstat -x`

<table>
<thead>
<tr>
<th>Job id</th>
<th>Name</th>
<th>User</th>
<th>Time Use S Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>19959.server{DataComplete}</td>
<td>test_dw_job</td>
<td>graham</td>
<td>00:00:00 F workq</td>
</tr>
<tr>
<td>19961.server{StagingOut}</td>
<td>test_dw_job</td>
<td>graham</td>
<td>00:00:00 F workq</td>
</tr>
<tr>
<td>19963.server{Ready}</td>
<td>test_dw_job</td>
<td>graham</td>
<td>00:00:00 R workq</td>
</tr>
<tr>
<td>19970.server{StagingIn}</td>
<td>test_dw_job</td>
<td>graham</td>
<td>0 Q workq</td>
</tr>
<tr>
<td>19977.server{Created}</td>
<td>test_dw_job</td>
<td>graham</td>
<td>0 Q workq</td>
</tr>
</tbody>
</table>
Closing
Altair Knows HPC

Altair is the only company that

*makes HPC tools…*

Altair | PBS Works™

AND *develops HPC applications.*

Altair | HyperWorks®

AND *uses these to solve real problems*

Altair | ProductDesign

700+ Altair engineers worldwide use HPC every day to design, simulate, and optimize highly-engineered products
For More Information

- **Stop by Altair’s table**
  - Discuss Altair HyperWorks, PBS Works and other offerings

- **Visit Altair online**
  - [www.altair.com](http://www.altair.com)
  - [wwwpbsworks.com](http://www.pbsworks.com)

- **Contact Altair**
  - Graham Russell
  - graham.Russell@uk.altair.com
Supporting Slides
CRAY PORT: ADMINISTRATOR’S POINT OF VIEW
New PBS Resources

• **vntype**
  - Differentiate between vnode types, two are defined by default
    - cray_login – login nodes where the MOM resides
    - cray_compute – Cray execution nodes where the computational work is done
  - Admins can define new types!!
  - Users can request the vnode type during submission

• **nchunk**
  - Number of chunks; replaces default_chunk.mppwidth
  - Only for specifying default number of chunks at server and queue level
    (default_chunk.nchunk)
  - Can NOT be requested by a job/reservation
New PBS Resources (cont.)

• **naccelerators**
  • Specifies the number of accelerators on a host
    • On a Cray: number of “UP” accelerators
    • Visible in pbsnodes output

• **accelerator**
  • Specifies whether an accelerator is associated with the vnode
  • Boolean Resource
    ```
    set node <nodename> resources_available.accelerator=True
    ```
  • Users can request accelerator w/ compute nodes
    ```
    qsub -l select=1:ncpus=2:accelerator=true
    ```
    myscript
New PBS Resources (cont.)

- **accelerator_model**
  - Automatically detected in ALPS inventory
  - Specifies the model of the accelerator
    - Example: Tesla_x2090
  - Users can request specific accelerator models
    ```
    qsub -l select=3:ncpus=2:acclerators=True:accelerator_model="Tesla_x2090":accelerator_memory=4000MB myscript
    ```

- **accelerator_memory**
  - Automatically detected in ALPS inventory
  - Specifies the amount of memory associated with the accelerator
  - Users can request accelerator memory
    ```
    qsub -l select=3:ncpus=2:acclerators=True:accelerator_model="Tesla_x2090":accelerator_memory=4000MB myscript
    ```
New Cray Custom Resources

- **PBScrayhost**
  - Used to differentiate the Cray systems
  
  - Enabling one instance of PBS Professional to manage multiple Cray systems
    - Requires CLE >2.2
    - If managing multiple Crays, qmgr -c “set sched do_not_span_psets=True”

- CLE 2.2 Cray systems; PBScrayhost=default
  - **WARNING:** Do not use one instance of PBS to manage more than one Cray CLE 2.2 system because it cannot differentiate between systems

- Users can request the PBScrayhost during submission
  qsub -l select=3:ncpus=2:PBScrayhost=examplehost
New Cray Custom Resources (cont.)

- **PBScraynid**
  - Set to the ALPS node ID
  - Visual cue for associating the PBS vnode to the Cray compute node

- **PBScrayseg**
  - Set to the ALPS NUMA node
  - Useful for the user if aprun -S, -sl, or -sn are used

- **PBScrayorder**
  - Order in which the nodes are returned in ALPS inventory (AKA NID Ordering)
  - Use node_sort_key: “PBScrayorder LOW” to match the order of the ALPS inventory
New Cray Custom Resources (cont.)

- PBS\textsc{craylabel\_<label name>}
  - ALPS inventory w/ labels get automatically created!
  - Boolean Resource
    \[
    \text{resources\_available.PBS\textsc{craylabel\_interlagos}=True}
    \]
  - Users can then request or avoid this resource using True or False
    \[
    \text{qsub -l select=3:ncpus=2:PBS\textsc{craylabel\_interlagos}=true myscript}
    \]

\textbf{NOTE: It is NOT required to have labels defined via xtprocadmin. Admins can create a custom string resource and associate a ‘label’ ... All within PBS!}
Hosts and vnodes

george_80_0
Mom = login1
state = free
pcpus = 4
resources_available.arch = XT
resources_available.host = george_80
resources_available.mem = 8192000kb
resources_available.ncpus = 4
resources_available.PBScrayhost = george
resources_available.PBScraynid = 80
resources_available.PBScrayorder = 1
resources_available.PBScrayseg = 0
resources_available.vnode = george_80_0
resources_available.vntype = cray_compute
sharing = force_exclhost

george_80_1
Mom = login1
state = free
pcpus = 4
resources_available.arch = XT
resources_available.host = george_80
resources_available.mem = 8192000kb
resources_available.ncpus = 4
resources_available.PBScrayhost = george
resources_available.PBScraynid = 80
resources_available.PBScrayorder = 1
resources_available.PBScrayseg = 1
resources_available.vnode = george_80_1
resources_available.vntype = cray_compute
sharing = force_exclhost

pbsnodes -av output

1 Compute Node with 2 NUMA Nodes
pbsnodes -av: vnode name

george_80_0
Mom = login1
state = free
pcpus = 4
resources_available.arch = XT
resources_available.host = george_80
resources_available.mem = 8192000kb
resources_available.ncpus = 4
resources_available.PBScrayhost = george
resources_available.PBScraynid = 80
resources_available.PBScrayorder = 1
resources_available.PBScrayseg = 0
resources_available.vnode = george_80_0
resources_available.vntype = cray_compute
sharing = force_exclhost
Hosts and vnodes: state

gorge_80_0

Mom = login1
state = free
pcpus = 4
resources_available.arch = XT
resources_available.host = george_80
resources_available.mem = 8192000kb
resources_available.ncpus = 4
resources_available.PBScrayhost = george
resources_available.PBScraynid = 80
resources_available.PBScrayorder = 1
resources_available.PBScrayseg = 0
resources_available.vnode = george_80_0
resources_available.vntype = cray_compute
sharing = force_exclhost

State of the vnode
apstat -n UP & B[ATCH]
Hosts and vnodes: Cray Custom Resources

george_80_0
Mom = login1
state = free
pcpus = 4
resources_available.arch = XT
resources_available.host = george_80
resources_available.mem = 8192000kb
resources_available.ncpus = 4
resources_available.PBScrayhost = george
resources_available.PBScraynid = 80
resources_available.PBScrayorder = 1
resources_available.PBScrayseg = 0
resources_available.vnode = george_80_0
resources_available.vntype = cray_compute
sharing = force_exclhost
Hosts and vnodes: vntype

george_80_0

Mom = login1
state = free
pcpus = 4
resources_available.arch = XT
resources_available.host = george_80
resources_available.mem = 8192000kb
resources_available.ncpus = 4
resources_available.PBScrayhost = george
resources_available.PBScraynid = 80
resources_available.PBScrayorder = 1
resources_available.PBScrayseg = 0
resources_available.vnode = george_80_0
resources_available.vntype = cray_compute
sharing = force_exclhost

Distinguish between login, compute, etc
Hosts and vnodes: vntype

downloads

resources

files

libraries

george_80_0

Mom = login1
state = free
cpus = 4
resources_available.arch = XT
resources_available.host = george_80
resources_available.mem = 8192000kb
resources_available.ncpus = 4
resources_available.PBScrayhost = george
resources_available.PBScraynid = 80
resources_available.PBScrayorder = 1
resources_available.PBScrayseg = 0
resources_available.vnode = george_80_0
resources_available.vntype = cray_compute
sharing = force_exclhost
Example: Serial Workload (non-MPP resources)

**Objective:** You do not want users reserving Cray MPP nodes for pre- or post-computational work, but would rather them use dedicated service nodes for this task. Which could be a mixture of external and internal login (service) nodes.

**Admin Tasks:**
1. Use qmgr to set the value for vntype on the vnodes representing external login nodes:
   ```
   qmgr -c "set node eslogin1 resources_available.vntype+="cray_serial"
   ```

2. Use qmgr to add cray_serial to the vnodes representing internal login nodes:
   ```
   qmgr -c "set node login1 resources_available.vntype+="cray_serial"
   ```

**User Task:**
- Submit the job
  ```
  qsub -l select=2:ncpus=2:vntype=cray_serial myscrip
  ```
Example: Gating Queues Based on PEs (min/max limits)

**Objective:** You are trying to gate your queues based on the total number of processing elements requested by a job.

**Solution:**
- Instead of: resources_min.mppwidth=8
- Use: resources_min.mpiprocs=8
- Make sure that mpiprocs can be counted for each job chunk, thus remember to set:
  
  qmgr –c “set server default_chunk.mpiprocs=1”

“Old” queue:
create queue workq
set queue workq queue_type=Execution
set queue workq resources_min.mppwidth=1
set queue workq resources_max.mppwidth=24

“New” queue:
create queue workq
set queue workq queue_type=Execution
set queue workq resources_min.mpiprocs=1
set queue workq resources_max.mpiprocs=24
CRAY PORT:
USER’S POINT OF VIEW
Requesting Job Resources – Chunks & Select

• A chunk is the ‘smallest’ unit of a job which can be placed on the host(s)/vnode(s)

  Syntax: qsub -I select=[ N: ]chunk

• Job requesting 3 chunks, each with 2 CPUs
  qsub -I select=3:ncpus=2

• Job requesting 3 chunks, each with 2 CPUs, PLUS 12 chunks, each with 1 CPU and have an accelerator
  qsub -I select=3:ncpus=2+12:ncpus=1:accelerator=true
Requesting Job Resources – mpiprocs

- **mpiprocs**
  - Defines the number of MPI processes for a job
  - Controls the content of the PBS_NODEFILE

- User requesting 3 chunks, each with 2 CPUs and running 2 MPI process

  ```
  qsub -l select=3:ncpus=2:mpiprocs=2
  ```

- **PBS_NODEFILE:**
  
  VnodeA
  VnodeA
  VnodeA
  VnodeB
  VnodeB
  VnodeC
  VnodeC
Requesting Job Resources – ompthreads

- **ompthreads**
  - pseudo-resource defining OMP_NUM_THREADS, per chunk
  - If ompthreads is not used, then OMP_NUM_THREADS is set to the value of the ncpus resource of that chunk

```bash
qsub -l select=3:ncpus=2:mpiprocs=2:ompthreads=1
```

<table>
<thead>
<tr>
<th>PBS_NODEFILE:</th>
<th>The OpenMP environment variables:</th>
</tr>
</thead>
<tbody>
<tr>
<td>VnodeA</td>
<td>For PBS task #1 on VnodeA: OMP_NUM_THREADS=1 NCPUS=1</td>
</tr>
<tr>
<td>VnodeA</td>
<td>For PBS task #2 on VnodeA: OMP_NUM_THREADS=1 NCPUS=1</td>
</tr>
<tr>
<td>VnodeB</td>
<td>For PBS task #3 on VnodeB: OMP_NUM_THREADS=1 NCPUS=1</td>
</tr>
<tr>
<td>VnodeB</td>
<td>For PBS task #4 on VnodeB: OMP_NUM_THREADS=1 NCPUS=1</td>
</tr>
<tr>
<td>VnodeC</td>
<td>For PBS task #5 on VnodeC: OMP_NUM_THREADS=1 NCPUS=1</td>
</tr>
<tr>
<td>VnodeC</td>
<td>For PBS task #6 on VnodeC: OMP_NUM_THREADS=1 NCPUS=1</td>
</tr>
</tbody>
</table>
Requesting Job Resources – Job Wide Resources

- **Job Wide Resources**
  - Resources that are requested outside a select statement
    - Examples: walltime, cput, …
  - Resources that are not associated to host(s)/vnode(s)

- **Job requesting 1 hour of walltime:**

  ```bash
  qsub -l select=3:ncpus=2 -l walltime=01:00:00 myscript
  ```
**Requesting Job Resources – Job Placement**

- **Users can specify how chunks are placed on vnodes using the “place” statement**

  Syntax:  `qsub -l select=<…> -l place= <type>| <sharing> | group=<res>`

  ```
  qsub -l select=3:ncpus=2 -l place=pack myscript
  ```

<table>
<thead>
<tr>
<th>Arrangement</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>type</code></td>
<td>free</td>
<td>place job on any vnode(s)</td>
</tr>
<tr>
<td></td>
<td>pack</td>
<td>all chunks will be taken from one host</td>
</tr>
<tr>
<td></td>
<td>scatter</td>
<td>only one chunk is taken from any host</td>
</tr>
<tr>
<td></td>
<td>vscatter</td>
<td>only one chunk is take from any vnode</td>
</tr>
</tbody>
</table>

| `sharing`   | excl  | only this job uses the vnodes chosen |
|             | exclhost | the entire host is allocated to the job |
|             | shared  | this job can share the vnodes chosen |

| `group`     | `<resource>` | chunks will be grouped according to a resource |
Requesting Job Resources – Job Placement “Free”

- **Request**: 3 chunks, each with 2 CPUs and running 2 MPI process, and place it ‘freely’. Each host has 8 CPUs and 2 GB memory

```bash
qsub -l select=3:ncpus=2:mpiprocs=2 -l place=free myscript
```

- **Variable** $PBS_NODEFILE contains list of vnodes

  vnodeA
  vnodeA
  vnodeA
  vnodeA
  vnodeA
  vnodeA
  vnodeA
  vnodeA
Requesting Job Resources – Job Placement “Scatter”

- **Request**: 3 chunks, each with 2 CPUs and running 2 MPI processes, but evenly distribute the chunks across the vnodes (scatter). Each host has 8 CPUs and 2 GB memory

```
qsub -l select=3:ncpus=2:mpiprocs=2 -l place=scatter myscript
```

- **Variable $PBS_NODEFILE contains list of vnodes**
  
  vnodeA
  vnodeA
  vnodeB
  vnodeB
  vnodeC
  vnodeC
Job Submission & Placement: aprun -sn

- qsub -l select=2:ncpus=4:mpiprocs=4 –l place=scatter

- Corresponding aprun statement:
  aprun -sn 1 -n 8 a.out
Job Submission & Placement: aprun -S

- qsub -l select=4:ncpus=2:mpiprocs=2 –l place=vscatter

- Corresponding aprun statement:
  aprun -S 2 -n 8 a.out
Job Submission & Placement: mpp* translation

Job Id: 43.login1

Job_Name = job

[...]
Resource_List.mppwidth = 8
Resource_List.ncpus = 8

[...]
Resource_List.place = free
Resource_List.select = 8:vntype=cray_compute
schedselect = 8:vntype=cray_compute:ncpus=1

[...]
Submit_arguments = -1mppwidth=8 job

Old syntax still works; qsub -l mppwidth=8

We also show what the command line arguments were when submitting (Submit_arguments).
Job Submission & Placement: mpp* translation

Job Id: 43.login1

    Job_Name = job
    [...] 
    Resource_List.mppwidth = 8
    Resource_List.ncpus = 8
    [...] 
    Resource_List.place = free
    Resource_List.select = 8:vntype=cray_compute
    schedselect = 8:vntype=cray_compute:ncpus=1
    [...] 
    Submit_arguments = -lmppwidth=8 job

For managers, to see what the scheduler is trying to solve. The schedselect parameter entails the union of the select specification of the job, and the queue and server defaults for resources in a chunk.
Where is my job running?

Job Id: 43.login1
  Job_Name = job
  Job_Owner = nishiya@login1
  job_state = R
  queue = workq
  server = sdb
  Ctime = Thu Jan 20 15:27:21 2011
  Error_Path = login1:/home/nishiya/test/job.e43
  exec_host = login1/0+login1/1+login1/2+login1/3+login1/4+login1/5+login1/6+login1/7+
              login1/8+login1/9+login1/10*0
  exec_vnode =
               (george_80_0:ncpus=1)+(george_80_0:ncpus=1)+(george_80_0:ncpus=1)+(george_80_0:ncpus=1)+
               (george_80_1:ncpus=1)+(george_80_1:ncpus=1)+(george_80_1:ncpus=1)+(george_80_1:ncpus=1)+
               (george_81_0:ncpus=1)+(george_81_0:ncpus=1)+(george_81_0:ncpus=1)+(george_81_1)
  Resource_List.place = free
  Resource_List.select = 10:vntype=cray_compute
  schedselect = 10:vntype=cray_compute:ncpus=1
  Submit_arguments = -lmppwidth=10 job
Where is my job running? (cont.)

Job Id: 43.login1

Job_Name = job
Job_Owner = nishiya@login1

...]
job_state = R
queue = workq
server = sdb
Checkpoint = u
ctime = Thu Jan 20 15:27:21 2011
Error_Path = login1:/home/nishiya/test/job.e43
exec_host = login1/0+login1/1+login1/2+login1/3+login1/4+login1/5+login1/6+login1/7+
login1/8+login1/9+login1/10*0
exec_vnode =

(george_80_0:ncpus=1)+(george_80_0:ncpus=1)+(george_80_0:ncpus=1)+(george_80_0:ncpus=1)+(george_80_1:ncpus=1)+(george_80_1:ncpus=1)+(george_80_1:ncpus=1)+(george_80_1:ncpus=1)+(george_81_0:ncpus=1)+(george_81_0:ncpus=1)+(george_81_1)

[...]
Resource_List.place = free
Resource_List.select = 10:vntype=cray_compute
schedselect = 10:vntype=cray_compute:ncpus=1

[...]
Submit_arguments = -lmppwidth=10 job

exec_vnode = NUMA Nodes, where application runs

You can see the NIDS!!
CRAY PORT: TROUBLESHOOTING
Don’t Do That!

Do NOT use vnode configuration files to configure nodes
• This overrides the ALPS inventory node information
• Use qmgr instead

Do NOT use resources_default.mpp*
• Not translated to select/place
• See PBS Admin Guide for new defaults

Users can NOT use –lselect/place with mpp* resources
• Example: -l mppwidth=6 -l select=mem=1GB
Useful to Know for Admins

No longer have to set ncpus=128 on the login nodes (read the new install/config instructions in the Install Guide/Admin Guide)

If you use min/max, set both mpp* min/max and select/place resource min/max on systems where users submit both types of jobs

resources_max.mppwidth=8
resources_max.mpiprocs=8*
*assumes default_chunk.mpiprocs=1 is set at the server level

Remember to add any new resources to the sched_config file “resources:” line if you want the scheduler to schedule based on it
• PBScrayhost, PBScrayseg, etc.

ALPS inventory is only read at:
• MOM startup
• MOM HUP
• When an ALPS reservation is rejected by ALPS
Useful to Know for Users

Compute Node ONLY Jobs

• Using the select/place language users no longer have to request a resource on the login node

Login & Compute Node Jobs

• If you actually want a resource on a login node, please list the login node first in your resource request in order to reduce inter-MOM communication
  • Example:

    Qsub -l select=1:ncpus=1:vntype=cray_login+2:ncpus=2:vntype=cray_compute
Useful to Know for Users (cont.)

Select Statement Order Matters!

qsub -l select=1:ncpus=1:vntype=cray_login+2:ncpus=2:vntype=cray_compute

qsub -l select=2:ncpus=2:vntype=cray_compute+1:ncpus=1:vntype=cray_login

**PBS assigns resources left to right**

- Useful for applications (e.g., CFD) which require decomposition node; make it first
- On Cray, the login node may or may NOT be the same login node where aprun is launched!
- In other words, introduce more inter-communication between PBS daemons
Admin Troubleshooting on a Cray

Transient ALPS reservation error preparing request:

- **Look in the mom_logs:**
  - “vnode <vnode name> does not exist”
  - “vnode <vnode name> has no arch value”
  - verify the reservation PBS makes in ALPS

- **HUP the MOM to re-read ALPS inventory**

Custom resource not showing up?

- **Look in the server_logs:**
  - “error: resource <name> for vnode <name> cannot be defined”
Admin Troubleshooting on a Cray (cont.)

Pbsnodes -av doesn’t show my compute nodes:

• Did you add the MOM vnode (i.e. login nodes) using
  qmgr -c “c n <login node>”

• Are the compute nodes in “batch” mode?
  • Use xtprocadmin to verify mode

The login node shows up as “stale”

• Does PBS list more than one vnode for the same login node?
  • PBS will use the hostname returned by the DNS
  • Although PBS will create a vnode for aliases, those vnodes will be marked stale because the PBS server does not talk to that vnode name
User Troubleshooting on a Cray

The job is not running on a compute node
• Did the job request vntype=cray_compute?
• Admin may want to set a default_chunk.vntype=cray_compute to help users out

The job is not running on the login node I want it to
• Did you list that login node first in the select/place request?
• Or did you use -l host with mpp* to tell it which login node to use?

The login node has some available resources but the job won’t run on it
• If the job running on the login node requested -l place=excl or -l place=exclhost then the job has the login node exclusively
Supplemental Slides
Thanks to all HPCwire readers for this honor!
We Work with Some of the Best

<table>
<thead>
<tr>
<th>Automotive</th>
<th>Aerospace</th>
<th>Heavy Equipment</th>
<th>Government</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ford</td>
<td>BAE Systems</td>
<td>ALSTOM</td>
<td>AIST</td>
</tr>
<tr>
<td>TOYOTA</td>
<td>Boeing</td>
<td>John Deere</td>
<td>NASA</td>
</tr>
<tr>
<td>CHRYSLER</td>
<td>Airbus Group</td>
<td>JFE</td>
<td>U.S. Air Force</td>
</tr>
<tr>
<td>GM</td>
<td>Bombardier</td>
<td>Caterpillar</td>
<td>Los Alamos National Laboratory</td>
</tr>
<tr>
<td>RENULT</td>
<td>Embraer</td>
<td>Oshkosh</td>
<td>National Research Council Canada</td>
</tr>
<tr>
<td>Honda</td>
<td>Honeywell</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Life/Earth Sciences</th>
<th>Electronics/Consumer Goods</th>
<th>Energy</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASF</td>
<td>Acer</td>
<td>ABB</td>
<td>AECOM</td>
</tr>
<tr>
<td>Bayer</td>
<td>Intel</td>
<td>Chevron</td>
<td>LIxil</td>
</tr>
<tr>
<td>Dow</td>
<td>IBM</td>
<td>ConocoPhillips</td>
<td>SOM</td>
</tr>
<tr>
<td>ALCOA</td>
<td>LG</td>
<td>ExxonMobil</td>
<td>SOM</td>
</tr>
<tr>
<td>MERCK</td>
<td>Bosch</td>
<td>WorleyParsons</td>
<td>WEIDLINGER</td>
</tr>
</tbody>
</table>

5,000 customers worldwide
All Across the Globe

45+ offices in 22 countries on 6 continents with over 2,200 engineers, scientists, developers, designers and creative thinkers
Software – At the Core of our Business

Simulation, optimization and analytics technologies that leverage high performance computing architectures to guide faster, more robust decisions to improve business and product performance.
Altair HyperWorks is a modern, open architecture CAE simulation platform, offering the best technologies to design and optimize high performance, weight efficient and innovative products.
Advanced Simulation Solutions

Integrated solvers platform with best-in-class scalability, quality, robustness. Solutions include structural (OptiStruct and RADIOSS), CFD (AcuSolve), multi-body dynamics (MotionSolve), and electromagnetic (FEKO) analysis. Altair’s solvers are optimization-ready and can be coupled for Smart Multiphysics analysis.

Simulation-driven Innovation
Optimization-ready
Structural, CFD, MBD and EM
Multiphysics
Systems Modeling / Development
Results Visualization & Validation

Analyze, understand and publish your simulation results with **HyperView** and **HyperGraph**, HyperWorks’ best-in-class post-processing and visualization environment for CAE and test data.

Results Visualization
Powerful Reporting
Math Functions
Engineering Analytics
PBS Works – Altair Knows HPC

PBS Works is the trusted leader in high-performance computing (HPC) workload management, with integrated solutions for intelligent job scheduling, web-based submission and analytics, and remote visualization.

Workload Management / Scheduling
Green Provisioning™
EAL3+ security
GPU/co-processor scheduling
Integrated management portals
Cloud Solutions for Compute-intensive Simulation

Altair’s cloud solutions simplify access to scalable on-premise and remote simulation and infrastructure resources to model, explore, analyze, organize, manage and collaborate across the enterprise.

Simulation Cloud Suite
HyperWorks Unlimited
Hosted HyperWorks Units