Altair

PBS Works™

Cray Workload Management with PBS Professional 12.0

Making HPC Faster, Simpler, and Smarter

Sam Goosen & Scott J. Suchyta

May 2013

Who is Altair?

Altair's Divisions and Companies

Altair

HyperWorks®

Simulation and
Optimization Technology

Business Analytics

Business Intelligence and
Data Analytics Solutions

PBS Works™

Workload Management and
Cloud Computing Solutions

ProductDesign

Product Innovation
Consulting

solidThinking™

Industrial Design
Technology

Staffing

Onsite, Hybrid and
Recruiting Services

Innovation Intelligence®

Altair

27+

Years of Innovation

40+

Offices in 19 Countries

1800+

Employees Worldwide

Global Presence

Altair

Seattle, USA	Montreal, Canada	Lund, Sweden	Moscow, Russia	Beijing, China
Salt Lake City, USA	Toronto, Canada	Gothenburg, Sweden		Shanghai, China
Mountain View, USA		Coventry, UK	Delhi, India	
Los Angeles, USA		Bristol, UK	Pune, India	Tokyo, Japan
Austin, USA	Detroit, USA	Manchester, UK	Chennai, India	Osaka, Japan
Houston, USA	Boston, USA	Stuttgart, Germany	Hyderabad, India	Nagoya, Japan
	Milwaukee, USA	Cologne, Germany	Bangalore, India	
	Charlotte, USA	Hamburg, Germany		Seoul, Korea
Mexico City, Mexico	Huntsville, USA	Hanover, Germany	KL, Malaysia	
		Munich, Germany		
		Graz, Austria		
		Paris, France		
		Lyon, France		Melbourne, Australia
	Sao Paulo, Brazil	Sophia Antipolis, France		
		Toulouse, France		
		Torino, Italy		
		Madrid, Spain		
		Thessaloniki, Greece		

Over **40 offices** across **19 countries**

Altair Knows HPC

Altair

Altair is the only company that…

Makes HPC tools

PBS Works™

Develops HPC applications

HyperWorks®

and uses these to solve real challenges!

ProductDesign

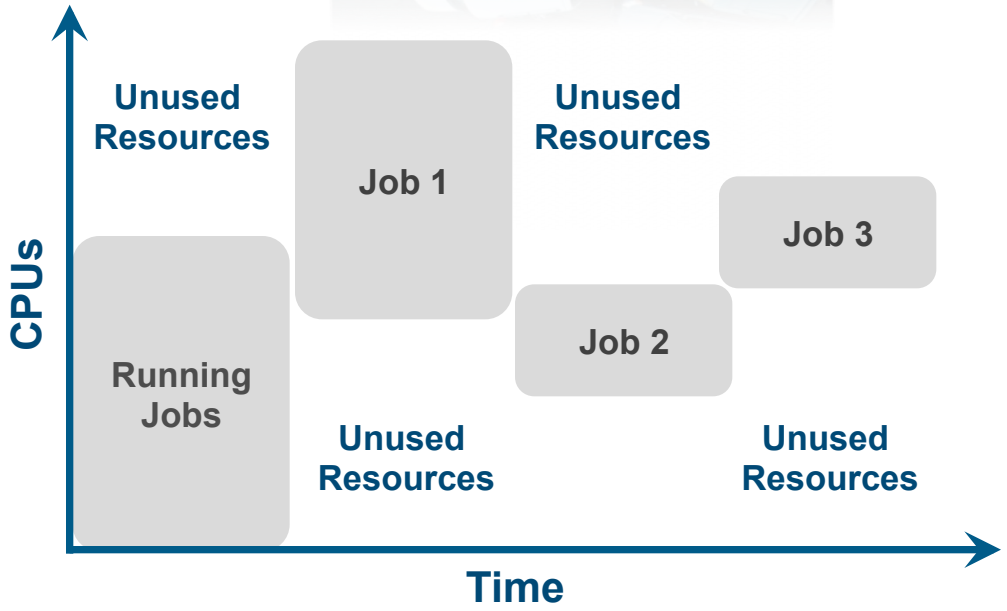
500 Altair engineers worldwide use HPC every day for real-world modeling & simulation

Brief Technical Update

High-performance Computing

Altair

- Complex problems
- World-wide teams
- Changing environment
- Finite resources
- Fragile infrastructure
- Hard deadlines
- Shrinking budgets



PBS Works: Enabling On-Demand Computing

Altair

- **Increase Productivity**
- **Meet HPC Goals**
- **Reduce Expenses**

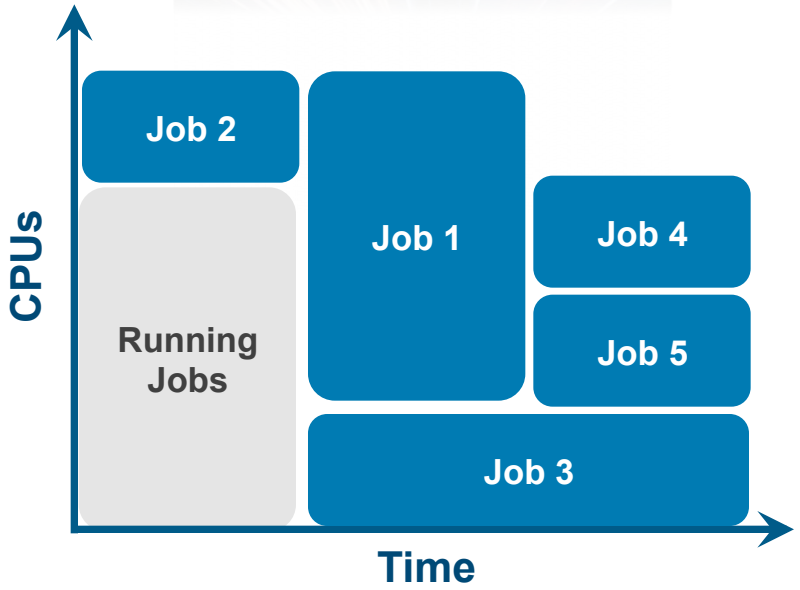
Easy to Use

Hard to Break

Do More (with less)

Keep Track and Plan

Open Architecture



PBS Works: Enabling On-Demand Computing

Altair

- **Increase Productivity**
- **Meet HPC Goals**
- **Reduce Expenses**

Easy to Use

Hard to Break

Do More (with less)

Keep Track and Plan

Open Architecture

PBS Portals

PBS Professional

PBS Analytics

PBS Professional Today: Feature Rich

Altair

GPU Scheduling	Fail-over	Beyond Petaflops Scalability	Dynamic Resources	Topology-aware Scheduling
Scheduling Formula	Policy-based Scheduling	Green Provisioning	Interactive Jobs	Job History
Fairshare	Heterogeneous Clusters	Hybrid Jobs (MPI+OpenMP)	User / Group / Project Limits	Multi-core
OS Provisioning	Kerberos	Over-subscription	$restrict_user	Peer Scheduling
Web Services	Age-based Scheduling	Standing Reservations	MPI Integrations	Backfill TopN
Job Dependencies	License Scheduling	Eligible Time	EAL3+ Security	Xeon Phi Scheduling
24x7 On-line Community	Extensible Plugin (‘hooks’)	Meta-scheduling	Checkpoint / Restart	MOM Hooks
On-demand Licensing	Estimated Job Start Times	Preemption	Job Arrays	“Shrink-to-fit”

Road to Exascale: 100x Today's Biggest Systems

Altair

Exascale is not new – it is a milestone along our Yottascale roadmap…

	Today	Exascale
Cluster size	10k hosts	→ 1M
Number of cores	100k cores	→ 10M (+ GPUs)
Workload	10k jobs running	→ 1M
Throughput	100k jobs per day	→ 10M
Energy use	20 megawatts	→ ??? — 1 gigawatt (*)
Physical size	100s of racks	→ 1000s

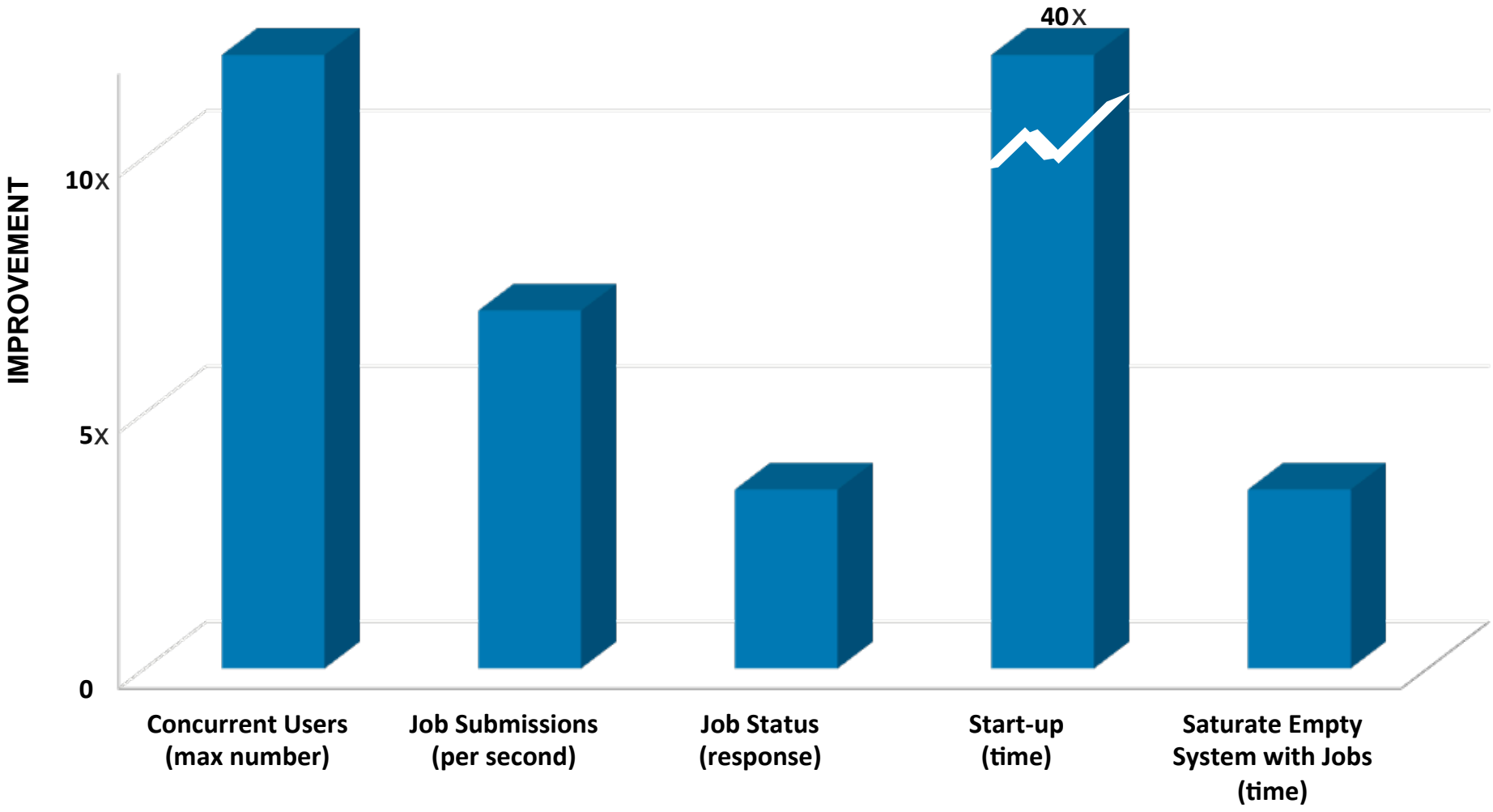
(*) Peak energy capacity of largest computing centers: ~20 megawatts

(Probably an upper bound, as 1 gigawatt is a whole nuclear power plant!)

Exascale systems are predicted to arrive by ~2020

Road to Exascale: PBS Professional 11 is ~10x Faster

Altair



PBS Professional 12.0 is Even Faster…

Altair

10x+ faster Power-on start-up	• Less than 1 minute for huge clusters
10x+ faster Add / delete nodes	• Seconds (vs. minutes) per rack
“Instant” start for interactive / debug jobs	• No delays waiting for the scheduler to “finish” • (Policies & available resources permitting)
40% faster scheduling for complex workloads	• Node equivalence class optimization • String processing optimization • “Job can not run” optimization

Re-architected Cray Support

Altair

Full “PBS vnode” features supported for Cray

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

Topology-aware Scheduling

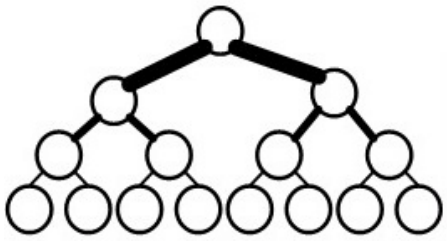
Altair

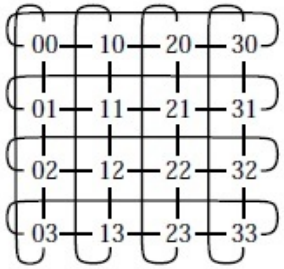
Speeds Application Performance and Boosts Utilization

Topology represented by “placement sets”

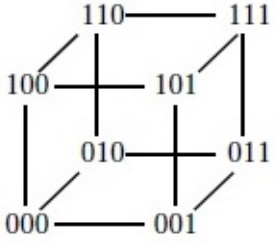
- Both inter (clusters, switches, grids) and intra (NUMA) node topology
- Infiniband, Ethernet, custom networks -- tree, torus, hypercube, dragonfly, anything
- Easily updated without the need to restart PBS

PBS Professional





$T(4,2)$



$M(2,3)$

*illustrations from HPC-Opinion.blogspot.com

Tunable Scheduling Formula

Altair

Define any policy – including on-the-fly "exceptions"

Simple formulas are very simple (big jobs go first)

```
ncpus * (walltime/3600.0)
```

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

```
(ncpus * (walltime/3600.0)) * Wsize +

(eligible_time/3600.0) * Wwait +

special_p
```

Optimized Backfill Scheduling

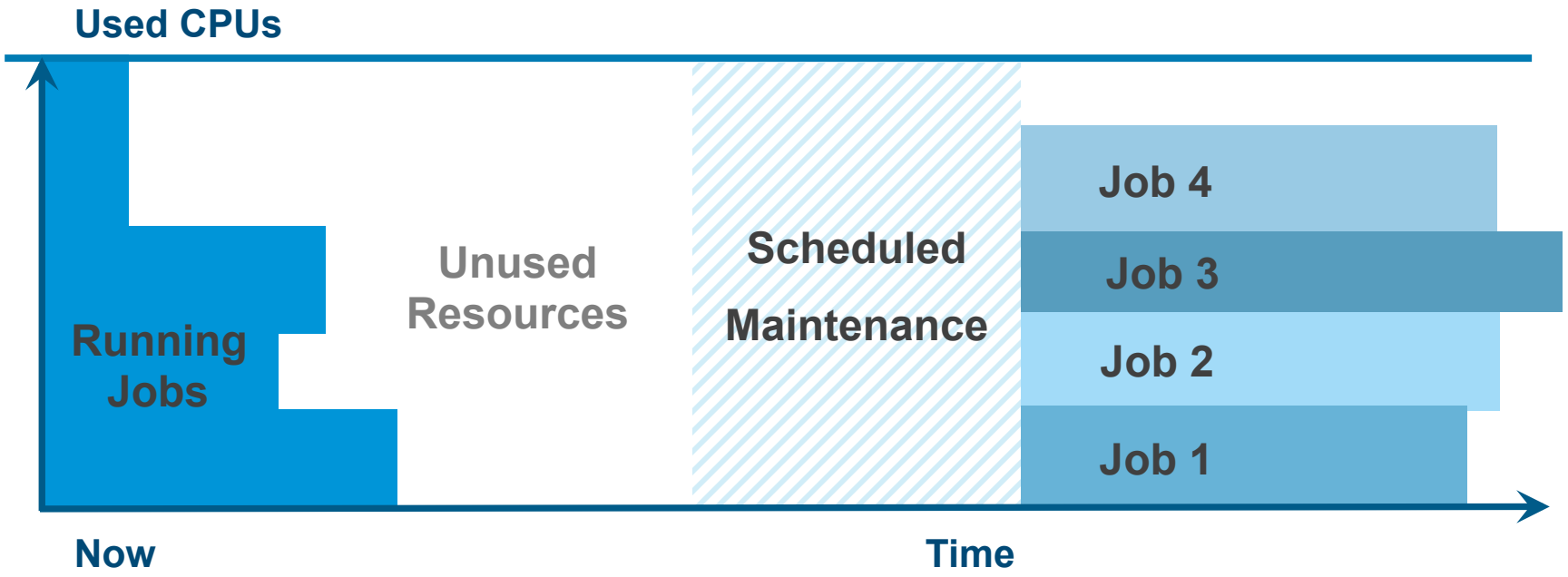
Altair

Eliminate Wasted Cycles without Delaying Any Work

Fills “gaps” without delaying any other jobs

Ensures very high utilization (esp. with advance reservations)

Run jobs right up to scheduled outages ("Shrink to Fit”)



Estimated Job Start Times

Altair

Plan Your Workflow & Meet Your Deadlines

```
% qstat -T

                                                                Req'd Req'd     Est
Job ID  Username Queue Jobname  SessID NDS TSK Memory Time  S Start
------- -------- ----- -------- ------ --- --- ------ ----- - -----
5.quark  bill    workq  foo      12345  1   1   128mb 00:10 R   -
9.quark  bill    workq  bar        --   1   1   128mb 00:10 Q 11:30
10.quark bill    workq  gril       --   1   1   128mb 00:10 Q 11:40
7.quark  bill    workq  baz        --   1   1   128mb 00:10 Q Tu 18
```

```
% qstat -f
    . . .
    estimated.exec_vnode = (pepsi:ncpus=1)
    estimated.start_time = Fri Apr 26 11:52:44 2013
```

Standing and Advance Reservations (with HA)

Altair

Guarantee resources for recurring needs

```
pbs_rsub -R 0500 -E 0800 \
    -r "FREQ=WEEKLY;BYDAY=MO,TU,WE,TH,FR;UNTIL=20131231" \
    -l select=200:ncpus=2 -l place=scatter:excl
```

- **Run the simulation from 5-8am every weekday morning**
- **Reserve the computing lab for classes on MWF 14:00-16:00**
- **Block out time for maintenance the first weekend of every month**

Users can create (without admin privileges), subject to access controls

Node failures are automatically detected and replaced, ensuring reservations are 100% fulfilled

Customizing PBS

PBS Plugins (“Hooks”)

Altair

Change / augment capabilities in the field, on-the-fly, without source

Unified data model based on industry-standard Python

Admission control events

- Validation, allocations, on-the-fly tuning, novel limits, logging, patches, …

Job execution events

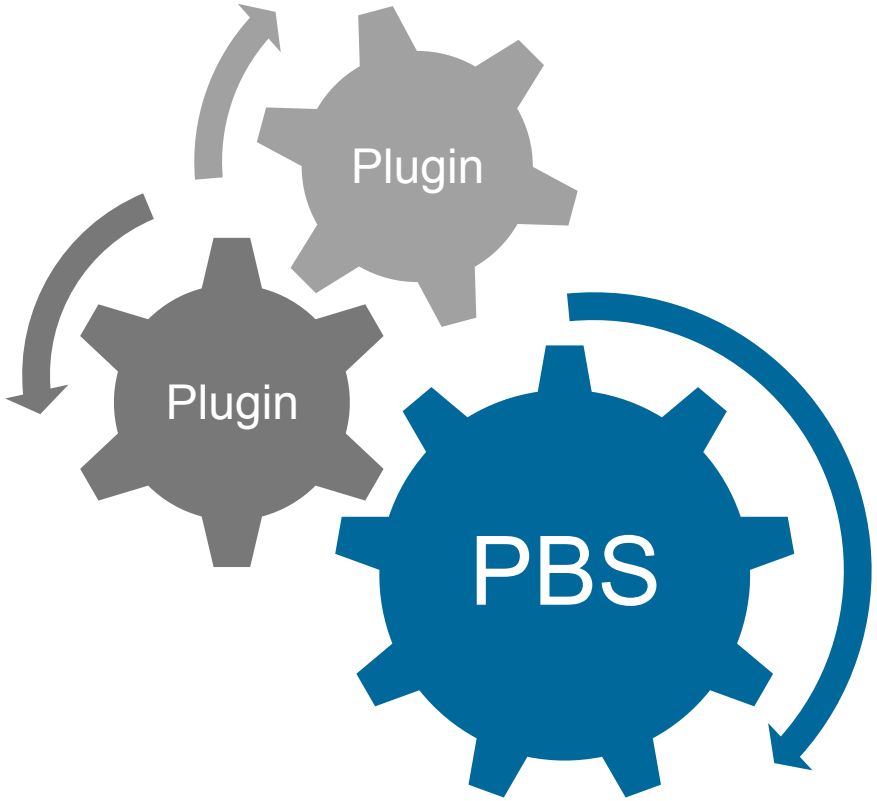
- Parallel node setup / cleanup
- Periodic monitoring
- Job termination

Plugin Examples

Altair

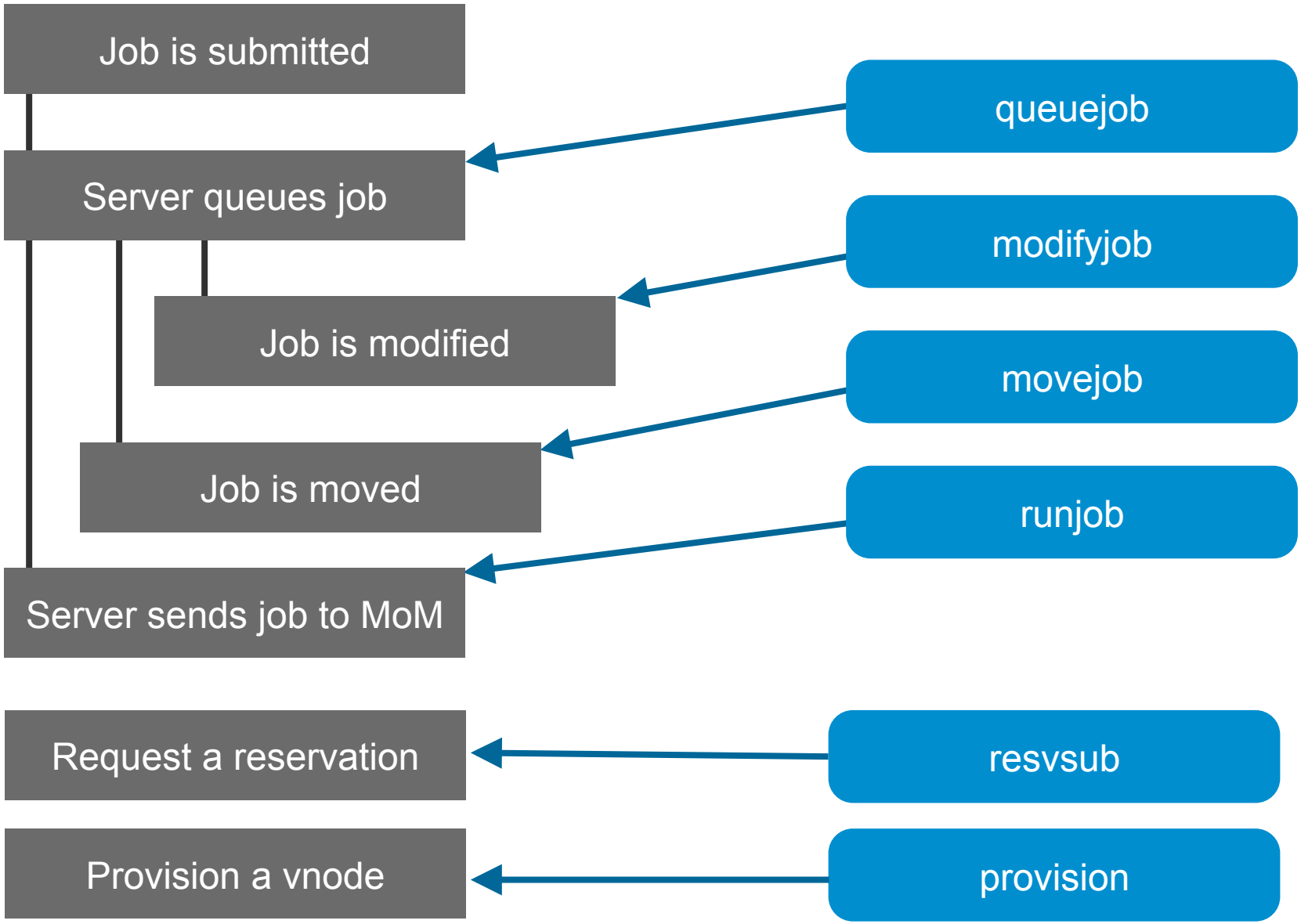
Plugins Deliver Real Capabilities

- **Fault detection & mitigation**
  - Mitigate “black hole” syndrome
- **Customized access control**
- **Customized runtime environments**
- **Allocation management**
- **Backward compatibility**
  - Automatic conversion of Cray “mpp*” syntax to newer select/place syntax



Plugins: Admission Control and Management

Altair



Submission Hooks

Altair

Change / augment / filter jobs as they arrive

Admission control – validate access

Verify complete job submission

Optimize job resource requests

On-the-fly job tuning at submission time!

Run Time Hook

Altair

Ensure allocation management limits are strictly enforced

Generic run job hook complements submission hook and enables pre-dispatch checks

Jobs can be requeued, held, released, and delayed

Enables almost any type of user / group / project /… limits, including limits set by allocation management systems

- E.g., Fred cannot start OptiStruct jobs on Sunday

Modify Job, Move Job, Reservation Hooks

Altair

Ensure site management limits are strictly enforced

Prevent users from modifying or moving job after submission

Controls what can be included in a reservation request and by whom

Enforce policy decisions that were set on jobs at submission time

New Hooks: Job Execution Events (MOM Hooks)

Altair

Job Lifecycle

1. Set up
2. Stage-in file(s)
3. Prologue
4. Launch job
5. Epilogue
6. Stage-out file(s)
7. Clean up

PBS Professional v12

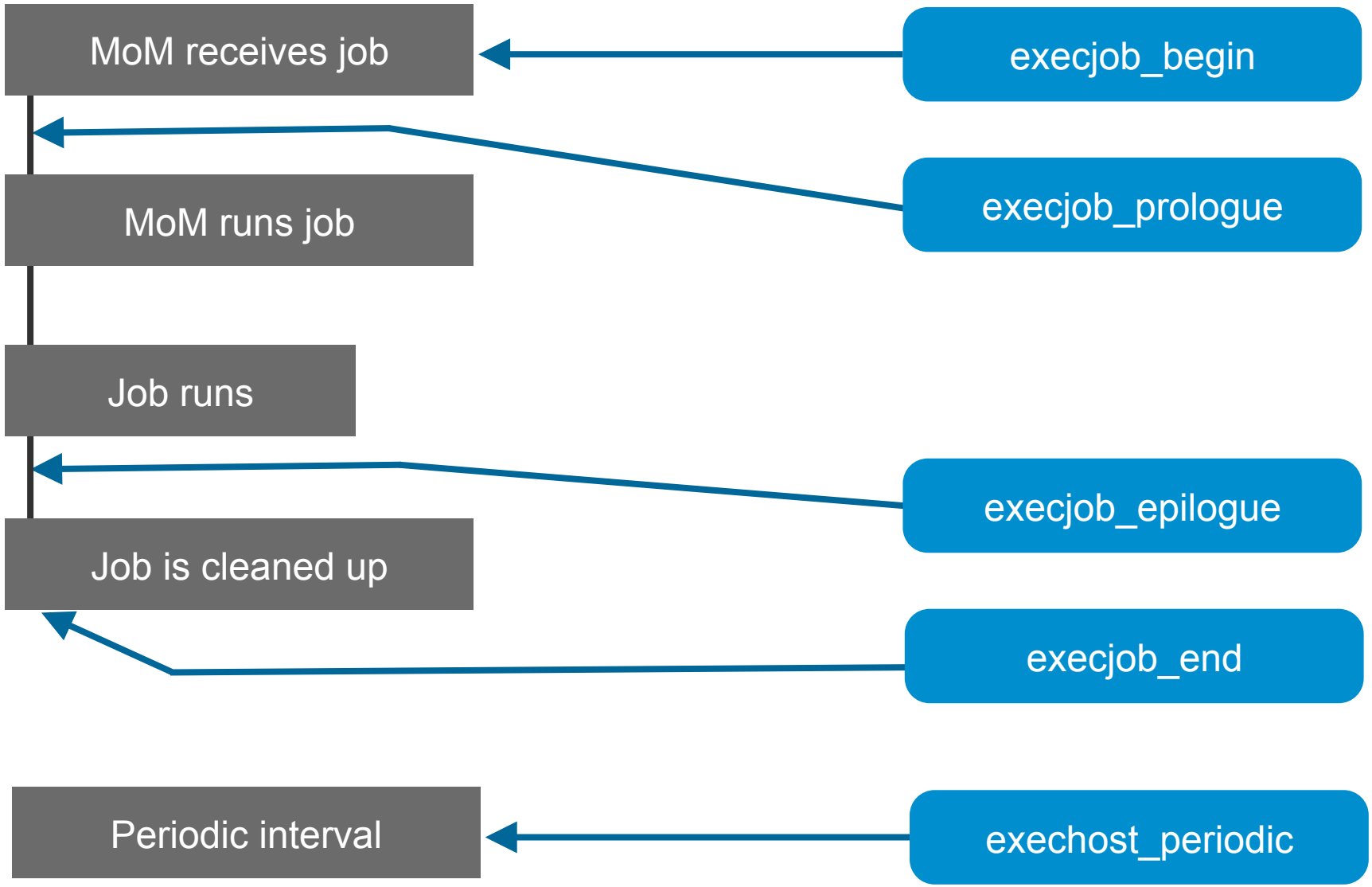
1. Adds Setup Hook (before environment…)
2. …
3. Adds Prologue Hook (replaces Prologue)
4. …
5. Adds Epilogue Hook (replaces Epilogue)
6. …
7. Adds Cleanup Hook (after obit…)

Plus…

- Periodic hooks
- Can execute as the user
- Configurable via qmgr
- Debugging facilities

Plugins: Execution (New in 12.0)

Altair



Cray Use Cases for MoM Hooks

Altair

- **Pre-job Health Checks**
  - Is ALPS really running?
  - Call node health checker before the job goes further
  - Set resource value(s) on PBS node
- **Amend User Environment**
  - Set environment variables to control job/application functions
  - Run script in same user environment as job will run in
  - Start user based accounting
- **Post-job Metrics & Health Checks**
  - Requeue checkpointed jobs
  - Reset the resources_used on a job that will be requeued
  - Set a flag for power usage & update a custom resource with values to be captured in accting logs
  - Stop user based accounting

Admission/Run Time Control

Problems/Solutions #1

Altair

- **Problem Statement:**
  - Users request incorrect queue for the project they belong to
- **Requirements:**
  - User's job request is corrected when possible
- **Solution: queuejob hook**
  - Hook sets correct queue as per project user belongs to

```
u = job.Job_Owner
p = job.project
if job.queue != "" :
        q = job.queue.name
proj.q = pickle.load(f)
if q != proj.q :
        q = proj.q
        pbs_logmsg(pbs.LOG_DEBUG, "queue change to %s" % (proj.q,))
```

Problems/Solutions #2

Altair

- **Problem Statement:**
  - Resource requests are made using select/place syntax that are impossible to convert into ALPS reservations
- **Requirements:**
  - User's job submission is rejected when it is not possible to correct
- **Solution: queuejob hook**
  - If problem is not "correctable" (not sure what user was really going for) job is rejected at submission time so that they don't have to wait for it to run before they are informed they made a mistake

```
# Need integer value for nppcu when creating ALPS reservation
cu = job.Resource_List['compute_unit']
cpus = job.Resources_List['ncpus']
if (cpus % cu) != 0 :
        e.reject ("ncpus/compute_unit does not resolve to integer value")
```

Problems/Solutions #3

- **Problem Statement:**
  - Application licenses were available when sched cycle started but by the time the job is run the license have been used by a previously scheduled job
- **Requirements:**
  - User's job is re-queued at runtime
- **Solution: runjob hook**
  - A final check just before runtime shows that the licenses are gone
  - The job will be rejected and requeued until the licenses become available

Problems/Solutions #3 (cont'd)

Altair

```
try:
    e = pbs.event()
    j = e.job
    jid = j.id
    f = j.Resource_List['app_feature']
    pfd = os.popen("lmstat -f app_feature")
        line = pfd.readline()
        l = line.split()
        total = l[6]
        in_use = l[11]
    pfd.close()

    if (total - in_use) < f :
        e.reject ("Job %s: Not enough of feature %s available at run time" % (jid,f,))

except SystemExit:
    pass
```

On-the-fly Tuning

Problems/Solutions #4

Altair

- **Problem Statement:**
  - Application requires that a certain variable be set in its environment
- **Requirements:**
  - The variable is not available in user's environment
  - The user requests application in PBS Pro "software" resource
- **Solution: MoM begin hook**
  - Detect application set into software resource
  - Set environment variable - it is passed through to all following stages of jobs life cycle

```
application = job.Resource_List['software']
if application != "" :
        e.job.Variable_List['PBS_APP'] = application
```

Problems/Solutions #5

Altair

- **Problem Statement:**
  - Job fails immediately after it starts to run due to system/configuration issue
- **Requirements:**
  - PBS should detect failure
  - Node should be offlined so no further jobs run there
  - Job should be re-queued
- **Solution: MoM epilogue hook**
  - Detect non-zero exit code and that walltime of job is <1% of job's requested walltime
  - Generate problem report and email to user/admin
  - Offline node where job was running
  - Requeue job
  - Restart sched cycle so no more jobs get sent to the problem node

Problems/Solutions #5 (cont'd)

Altair

```
EMAIL=None # can set to say EMAIL="john_doe@foo.com"
e = pbs.event()
jid = e.job.id
# If job had non-zero exit status…
if e.job.Exit_status != 0 :
    w = float(e.job.Resource_List['walltime'])
    w_used = float(e.job.resources_used['walltime'])
   # …and walltime was less than 1% of requested
    if (w_used/w)*100 < 1 :
        nid = pbs.get_local_nodename()
        if EMAIL:
           fd=open(EMAILMSG, "w")
           msg="Node %s will be set offline due to failure of job %s" % (nid,jid,)
           fd.write(msg)
           fd.close()
           os.system("mail -s '" + SUBJECT  + "' "  + EMAIL + " < " + EMAILMSG)
        e.vnode_list[nid].state = pbs.ND_OFFLINE
        e.job.rerun()
        pbs.server().scheduler_restart_cycle()
        e.reject ("Job %s: Rejecting job and setting node %s offline" % (id,nid,))
```

Problems/Solutions #6

Altair

- **Problem Statement:**
  - Site wants to record aprun submission used in PBS accounting logs
- **Requirements:**
  - aprun cmd_line tracked in Cray aprun syslog
- **Solution: MoM epilogue**
  - Read value of resource from Cray aprun logs
  - Set Resources_Used value so that it is captured in PBS accounting log

```
e = pbs.event()
j = e.job
cmd_line = <parse cmd_line out of aprun syslog>
j.resources_used['aprun_cmd_line'] = cmd_line
```

Allocation & Accounting

INPE/Cray

INPE: Problem Statement & Requirements

Altair

Problem Statement:

- The MPP resources on the Cray must be divided between the 3 institutions:
  - 40% of the machine for CPTEC
  - 30% for FAPESP
  - 30% for CCST

Requirements:

- New institutions can be added in the future
- Users from each institution will run jobs that must be assigned to a project
- Each project has an associated institution and project type -- operations, production, development, or research
- Must be in Portuguese

INPE: Implementation

Altair

- **MySQL database**
  - Tables representing the allocation allotment for each institution
- **queuejob hook**
  - User must specify a project (-A) & walltime (-l walltime)
  - User must have an entry for a project in the projects table
  - Project type must match the queue being requested
- **runjob hook**
  - Query the database to check if the user's institution has enough allocations to run
  - If not, the job will be rejected and remained queue until the next month
  - Deduct request allocation from database
- **epilogue hook**
  - Credit the job's unused resources used from the institutions' total

User's Experience

- **Custom messages in Portuguese!**


```
bass-p4:/tmp # qsub -lmppwidth=1 qsub.script
qsub: TUPA: Especifique walltime


bass-p4:/tmp # qstat -as
bass-p4:
Req'd Req'd Elap
Job ID Username Queue Jobname SessID NDS TSK Memory Time  S Time
------ -------- ----- ------- ------ --- --- ------ ----- - ------
273.bass-p4 root workq qsub.scrip --  1   1    --   24:00 Q  --
Not Running - PBS Error: Não existem recursos suficientes para o instituto
```

Intel Hyperthreading

Problem Statement & Requirements

Altair

Problem Statement:

- Prior to CLE 4.1 the Cray ALPS inventory did not offer a way to make a distinction between physical and virtual (Hyperthreaded) CPUs (aka PEs)
- Running jobs on Hyperthreaded CPUs can result in less than optimal performance and may want to be avoided

Requirements:

- Approach must be flexible and allow users to indicate whether or not to use Hyperthreaded CPUs

Implementation

- **Admin sets value for custom resources on each PBS vnode to represent the number of Compute Units available**
  - Compute Unit is physical chip containing one or more CPUs that share execution resources
- **User requests number of Compute Units and number of ncpus they want on each vnode**
  - If they request ratio of 1 ncpus:1 CU they will not be use any Hyperthreaded CPUs
  - If they request ratio of 2 ncpus:1 CU they will use Hyperthreaded CPUs
  - Default (do not request CUs) is to use whatever is available
- **Queuejob hook is used to make sure ratio is valid**
  - Depending on site job could be rejected or ratio could be be “corrected”

Now Available: PBS on XE6m-200

Altair

Affordable Supercomputing with Optimized Workload Management

- Limited time promotional pricing of PBS Professional for the XE6m-200 product family
- Deep benefits for users
- For more information: www.pbsworks.com/pdfs/Cray-Altair-XE6m200-Offer.pdf

PBS Works™

PBS Works User Group 2013

Altair

Taking Place at the Americas Altair Technology Conference

October 1-3, 2013 | Garden Grove, California | Orange County

Register Today!

altair.com/pbsworksug13

Listen

Learn

Network

Thanks for your time!!

Altair

- **Stop by the Altair table**
  - Cray-PBS Professional Developer Lisa Endrjukaitis is here!!
  - Discuss PBS Works and other offerings

- **Contact Altair**
  - Sam Goosen
  - Product Specialist, PBS Professional
  - Email: smgoosen@altair.com
  - Scott Suchyta
  - Director, Partner Solutions & Integration
  - Email: scott@altair.com

- **Visit Altair online**
  - www.altair.com
  - www.pbsworks.com
  - www.altairhyperworks.com

Supporting Slides

CRAY PORT: ADMINISTRATOR'S POINT OF VIEW

New PBS Resources

Altair

- **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.)

Altair

- **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.)

PBS Works™

- **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

Altair

- **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.)

Altair

- **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.)

Altair

- **PBScraylabel_<label name>**
  - ALPS inventory w/ labels get automatically created!
  - Boolean Resource

    resources_available.PBScraylabel_interlagos=True

  - Users can then request or avoid this resource using True or False

    qsub -l select=3:ncpus=2:PBScraylabel_interlagos=true myscript

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

Altair

```
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

Altair

```
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
```

vnode name
- Cray Host
- ALPS Node ID
- ALPS NUMA Node #

Hosts and vnodes: state

Altair

```
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
```

State of the vnode
apstat -n UP & B[ATCH]

Hosts and vnodes: Cray Custom Resources

Altair

```
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
```

Cray Host

ALPS Node ID

ALPS NID Ordering

ALPS NUMA Node

Hosts and vnodes: vntype

Altair

```
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
```

vntype

Distinguish between login, compute, etc

Hosts and vnodes: vntype

Altair

```
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
```

Today, Compute Nodes are unable to execute more than one application, but tomorrow..

Example: Serial Workload (non-MPP resources)

Altair

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 myscript

Example: Gating Queues Based on PEs (min/max limits)

Altair

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

Altair

- **A chunk is the 'smallest' unit of a job which can be placed on the host(s)/vnode(s)**

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

- **Job requesting 3 chunks, each with 2 CPUs**

  qsub -l select=3:ncpus=2

- **Job requesting 3 chunks, each with 2 CPUs, PLUS 12 chunks, each with 1 CPU and have an accelerator**

  qsub -l select=3:ncpus=2+12:ncpus=1:accelerator=true

Requesting Job Resources – mpiprocs

Altair

- **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
  VnodeB
  VnodeB
  VnodeC
  VnodeC

Requesting Job Resources – ompthreads

Altair

- **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

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

PBS_NODEFILE:	**The OpenMP environment variables:**
VnodeA	For PBS task #1 on VnodeA: OMP_NUM_THREADS=1 NCPUS=1
VnodeA	For PBS task #2 on VnodeA: OMP_NUM_THREADS=1 NCPUS=1
VnodeB	For PBS task #3 on VnodeB: OMP_NUM_THREADS=1 NCPUS=1
VnodeB	For PBS task #4 on VnodeB: OMP_NUM_THREADS=1 NCPUS=1
VnodeC	For PBS task #5 on VnodeC: OMP_NUM_THREADS=1 NCPUS=1
VnodeC	For PBS task #6 on VnodeC: OMP_NUM_THREADS=1 NCPUS=1

Requesting Job Resources – Job Wide Resources

Altair

- **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:**

  qsub -l select=3:ncpus=2 -l walltime=01:00:00 myscript

Requesting Job Resources – Job Placement

Altair

- **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

Arrangement	Value	Description
type	free	place job on any vnode(s)
	pack	all chunks will be taken from one host
	scatter	only one chunk is taken from any host
	vscatter	only one chunk is take from any vnode
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”

Altair

- **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

  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

Requesting Job Resources – Job Placement "Scatter"

Altair

- **Request:** 3 chunks, each with 2 CPUs and running 2 MPI process, 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

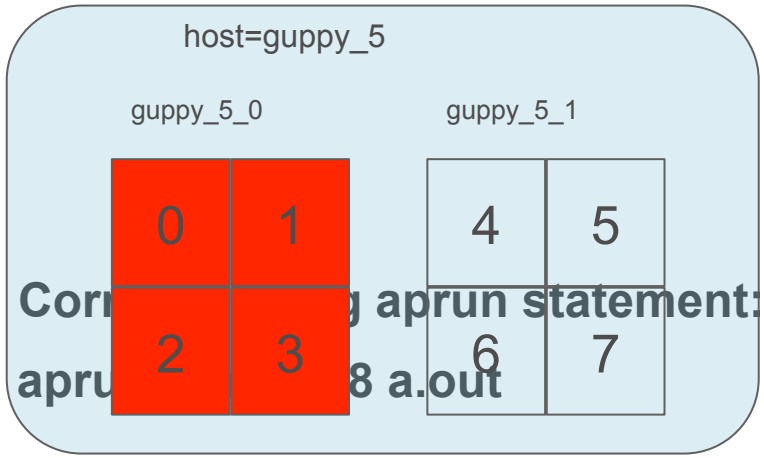
Job Submission & Placement: aprun -sn

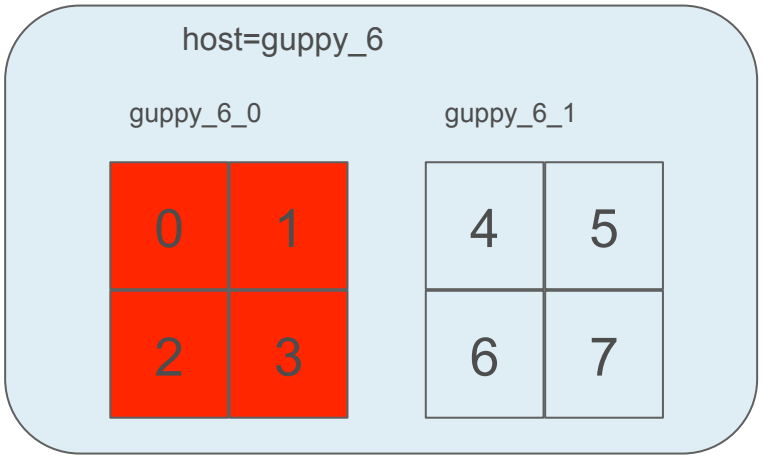
- **qsub**

Use a given number of segments

- **Corr g aprun statement:**

apru 8 a.out



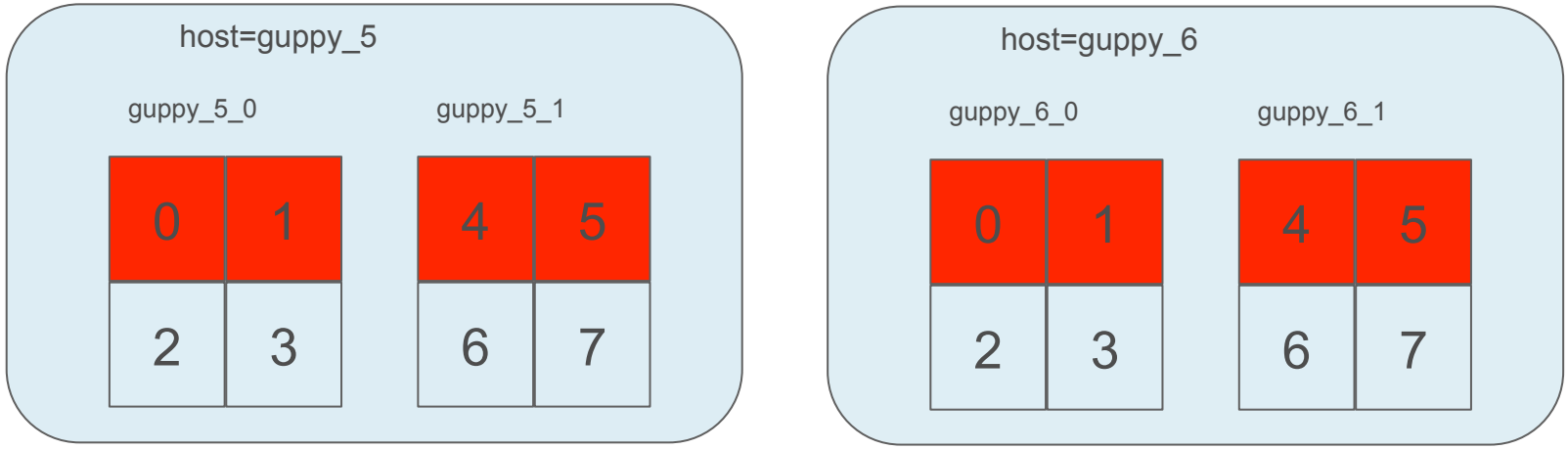


Job Submission & Placement: aprun -S

Altair

Specify the number of PEs per segment

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



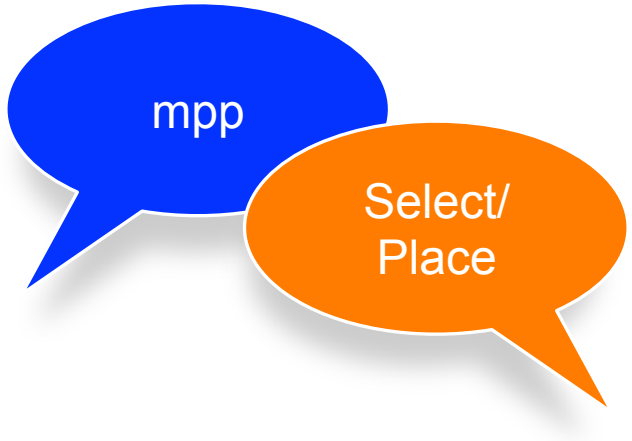
- **Corresponding aprun statement:**

  aprun -S 2 -n 8 a.out

Job Submission & Placement: mpp* translation

Altair

```
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
```



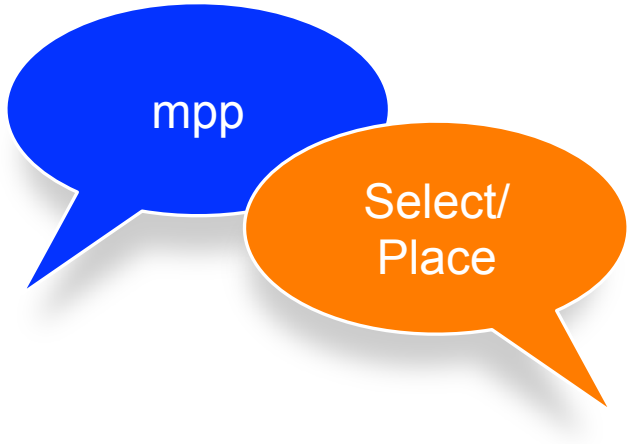
Old syntax still works;
qsub -l mppwidth=8

We also show what the command line arguments were when submitting (Submit_arguments).

Job Submission & Placement: mpp* translation

Altair

```
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
```



Automatically translated to select language

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?

Altair

```
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_host = Login Node, where jobscript is executed

Where is my job running? (cont.)

Altair

```
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!

Altair

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 11.2 Table 11-1 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

Altair

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

Altair

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.)

Altair

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

Altair

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.)

Altair

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

Altair

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