Linux Clusters Institute: Scheduling with PBSPro

Ben Matthews, Software Engineer, NCAR
What does a batch scheduler do?

- Large scale, high-performance Tetris!
- Divide up a shared resource “fairly”
  - “fair” may depend on politics/business logic
  - Jobs should never “starve”
- Prevent users from stepping on each other
- Ensure system is utilized as much as possible
- Record statistics/track usage
- Assist middleware [MPI] with multi-node job launch
PBSPro

• Developed at NASA Ames (Mountainview CA/Moffett Airforce Base)
• Acquired by Altair Engineering in the early 00’s
• Dual commercial/Open (Affero GPL v3)
• Support for various UNIX and Windows (mostly Linux + Windows)
• Various forks have been developed over the years
  • TORQUE (Adaptive Computing – Commercial)
  • OpenPBS (Open,ish)
• Reasonably Performant and Scalable
PBSPro: Features

• Rich “hook” infrastructure for customization
• Fairshare
• Backfill
• Compatibility with a wide variety of MPI implementations
• Well understood (if not well documented) accounting log
• Optional analysis components (commercial add-on only)
PBSPro: Basic Components

• Server
  • Runs on one or two, typically dedicated, servers
  • Mediates between other components and maintains the queues

• Scheduler
  • Takes a list of jobs from the server, tells the server which to run

• MoM
  • Runs on each compute node, starts and supervises user code. Optional job-setup on the compute node.

• Com
  • Facilitates communication between components (only needs configuration at scale)
Hardware Requirements

• For failover
  • Two servers
  • Shared filesystem (must support POSIX locking, but NFS ok)
• Server software is not well threaded yet: prefer higher clock frequency over many cores
• Database performance matters (SSD would be nice)
• Queues are stored in RAM, so memory usage scales with queue length. More RAM = better
System Software Setup

• One user account for PBS to run under (typically “pbsdata”)
• UID<>username mapping should be consistent across the cluster
• Optionally, MUNGE can be used to authenticate the cluster
• ssh (or rsh), allowing passwordless connections between cluster nodes (use ssh keys or host trust) strongly recommended
• scp or rcp (or similar) must work (passwordless) between submit hosts and cluster nodes (for file staging)
• shared filesystem on compute nodes is **not** required, but is strongly recommended
Installation

- RPMs provided to commercial customers for SLES and RHEL
- Can also build from source (and optionally produce RPMs)
- Four install types:
  - Server
  - Execution Host
  - “commands only” (head-node)
  - Everything (direct install from source)
- RPMs are relocatable (see the documentation for details)
Installation: pbs.conf

• Present on all nodes
• Specifies which components to start and where the servers are
• Used to generate the initial configuration
/etc/pbs.conf

PBS_EXEC=/opt/pbs
PBS_HOME=/gpfs/pbs
PBS_START_SERVER=0
PBS_START_MOM=0
PBS_START_SCHED=0
PBS_START_COMM=0
PBS_SERVER=laadmin1.ib0.laramie.ucar.edu
PBS_PRIMARY=laadmin1.ib0.laramie.ucar.edu
PBS_SECONDARY=laadmin2.ib0.laramie.ucar.edu
PBS_SCP=/usr/bin/scp
PBS_RSHCOMMAND=ssh
PBS_CORE_LIMIT=unlimited
PBS_MAIL_HOST_NAME="ucar.edu"
PBS_AUTH_METHOD=MUNGE
Other places to look for configuration

• $PBS_HOME/{sched,server}_priv/{sched,server}_config

• “qmgr”

• Per-component configuration typically generated during the first startup
Starting PBS

- `systemctl start pbs` \(\geq\) version 14, systemd systems
- `/etc/init.d/pbs start` \(\leq\) version 13, other init types
- Must be done on the server node(s) and all execution nodes
PBSPro: Commands

• qmqr
  • Configuration

• qsub
  • Submit Jobs

• qstat
  • View Status of Jobs/Queues/Servers

• qrls/qhold
  • Hold/Release Jobs

• pbs_rsub/pbs_rstat/pbs_rdel
  • Manipulate Reservations
PBS Objects

- Queues – collect jobs
- Nodes – run jobs
- Resources – generic properties
Queues

- Two types:
  - Execution
  - Routing
- Routing queues accumulate jobs and pass them on to execution queues based on resource requests
- Execution queues store jobs and dispatch them to nodes
- May be fixed length or unrestricted (up to the amount of memory on your PBS server)
- May impose various restrictions/access controls
Manipulating Queues

- `qmgr -c 'create queue new_queue'` #create a queue
- `qmgr -c 'set queue new_queue your_resource = foo'` #set a resource
- `qmgr -c 'set queue new_queue started=true'` #run jobs
- `qmgr -c 'set queue new_queue enabled=true'` #accept new jobs
- `qmgr -c 'print queue @default'` #list queues
Resources

- Used to control the flow of jobs through PBS
  - Typically used much less extensively by non-PBS schedulers (SLURM)
- Can be requested by the user, a hook, or required by a queue
- Static (set by an admin) or Dynamic (collected by the server/MoM)
Resources

• Things your job needs
• Could be strings, numbers, etc
• Can be a thing that a specific node or queue provides
• Could be provided by something external (like a license server) but tracked by the scheduler
• Could be a simple property of a job
Adding Custom Resources

• Make an entry in $PBS_HOME/server_priv/resourcedef
• If you are going to schedule based on your resource, add it to the “resources” list in $PBS_HOME/sched_priv/sched_config
• Attach the resource to nodes/queues/etc in qmgr
Nodes

• Physical Hardware to run on (where MoM runs)
• Can be subdivided into vnodes which can be independently scheduled
• Can be assigned resources that can then be used to control which jobs will run
• Can be assigned dynamic resources, which are periodically measured by user provided or built-in scripts (load average, memory usage, etc).
Nodes

• May be in various states – see the output of “pbsnodes –a”
• Common States:
  • offline – node is broken or marked down by an administrator
  • job-exclusive – node is completely allocated to a job
  • free – node is available for use
  • down – Server and MoM aren’t communicating
  • resv-exclusive – node is completely reserved
• Nodes can be in multiple states, for example, reserved and running a job
Built-in Node resources

• Memory
• CPUs
• Free Disk
• Load Average
• Hostname
• OS
• Vnode
• System specific details (Cray, etc)
Manipulating Nodes

- `qmgr -c 'create node node001'` #create a new node
- `qmgr -c 'print node @default'` #list nodes
- `pbsnodes -o -c "this node is broken" node001` #offline node001
- `pbsnodes -r node001` #online node001
- `pbsnodes -a -F dsv` #list nodes, parseable
ACLs and Security

• Currently apply to user-id or a user’s **primary** group
• Can be used to restrict who may run how many jobs in which queue
• Can be supplemented with user-supplied python-scripts for additional flexibility (“hooks”)
• Configured as queue properties via qmgr
• Kernel provided user information is trusted, however MUNGE can be used to provide some level of authentication
• Users can be granted limited administrative privileges (killing other user’s jobs, running qmgr, etc)
Hooks

• Short python scripts that are run at various points in the scheduling process
• Can be used to implement additional business logic or functionality
• Need to be fast and robust
  • Run by the Server or MoM and can therefore break the Server or MoM
• Newish feature with some rough edges but very flexible
  • Avoid them if you can
• Configured via qmqr
Components of a Job

- Resources you need? `-l select=1:ncpus=1:mpiprocs=1`
- For how long? `-l walltime=1:00:00`
- Where? `-l place=scatter`
- Which Queue? `-q workq`
- Where to put the output? `-o stdout.log -e stderr.log`
- See the `qsub` manpage for other options
Placement

• Placement statement is three colon delimited (optional) clauses:
  • -l place=arrangement:sharing:grouping

• Arrangement determines where the resources are allocated
  • free: use any free vnodes
  • pack: try to use vnodes from one (or as few as possible) hosts
  • scatter: one chunk per host
  • vscatter: one chunk per vnode

• Sharing
  • excl: vnodes aren’t shared (but hosts might be)
  • exclhost: hosts aren’t shared
  • group= group by some resource
Select

• Request a number of “chunks” containing some resources
• Chunks are plus delimited
• Resources are colon separated
• First part of a chunk is the number of copies
• `qsub -l select=2:ncpus=4:mem=1gb+1:ncpus=2:mem=5gb`
  • Give me 2 vnodes, with 4 cpus and 1gb of ram
  • Also 1 vnode with 2 cpus and 5gb of ram
• Can be quite complex – see the documentation for details
Sample Job

#!/bin/bash
#PBS -l walltime=1:00
#PBS -lselect=2:ncpus=1:mpiprocs=2
#PBS -A SSSG0001
#PBS -N test_job
#PBS -q share

cd PBS_O_WORKDIR
mpirun ./a.out
Sample Job - Running

chmod +x job_script.sh
qsub ./job_script.sh
Scheduling Features: backfill

- Scan lower priority jobs for tasks that can be fit between larger/longer jobs
- Expensive, but can significantly improve utilization
- Most effective when jobs make accurate walltime requests
- The number of jobs to schedule around is configurable (may have a significant impact on your scheduling time)
  - backfill_depth
Scheduling Features: Fairshare

• Jobs are prioritized by a configurable per-user (or per-project) importance score (potentially hierarchical)
• Jobs are de-prioritized by historical usage (subject to some half-life)
• Users who are “important” and/or haven’t used much CPU*time recently run first
• Prevent any one user from monopolizing the system
• Give those who are more important a little more priority while still preventing starvation
Scheduling Features: Fairshare

- Priority = usage/allocated_percentage_of_system
- Usage is decayed by a configurable factor every configurable unit of time
- Default decay is a 24 hour half-life
- Usage is a configurable expression, often cpu*seconds
- Percentage is based on “shares”
  - Configure 10 users with 100 shares each, each user gets 10%
  - Shares are in arbitrary units
  - Defined in $PBS_HOME/sched_priv/resource_group
- Smaller priority = run sooner
Scheduling Features: Placement Sets

• Special resources attached to nodes
• Scheduler will try to keep each job in the smallest possible placement set
• Ensure locality – try to keep a job’s assigned nodes within a switch/rack/datacenter/etc
Array Jobs

• An optimized way to run N of the same job
• Each job is passed its index (useful to specialize)
• Potentially faster to schedule than individual jobs
• `qsub -J "1-100"`
Job Sort

• Which job should we run next?

• Configurable. Can be based on a user-defined python expression, fairshare, or static queue based priority

• Many schedulers allow you to use multiple scaled factors – PBS does not (yet)
  • Secondary factors used only to break ties – if two jobs have exactly equal fairshare priority, only then will queue priority take effect
  • Can turn off fairshare completely

• Can use a site-specific python function, but can’t integrate fairshare score
Additional Configuration: Health Check

• Verify that a node is healthy right before launch
  • Filesystems mounted
  • All the RAM present
  • CPUs present and at a reasonable frequency/temperature
  • Network up
  • OS correct
  • Daemons running

• If not, requeue the job and mark the node offline
Additional Configuration: Health Check

• Example hook provided with PBS: $PBS_EXEC/unsupported/NodeHealthCheck.py
• NHC was presented yesterday – can be integrated with PBS via hooks
• Good idea to run something right before and/or right after each job
• Details tend to be site specific – track what your users break and check for that
Additional Configuration: cgroups

• Contain each job to the resources that it requested
• Prevent out-of-memory events or other buggy code from breaking your compute nodes
• Newer OS may require integration with systemd instead of cgroups directly
• No direct support in PBS, but can be accomplished with hooks. Altair can provide a sample upon request
Troubleshooting: Why won’t my job run??

• Because you asked for something silly!
  • qstat -f jobid
  • Check the comments field
  • Check that the job needs <= sizeof(cluster)

• Because too many nodes are broken
  • pbsnodes -l

• Because the system is otherwise broken
  • Filesystem mounted?
  • Network up?

• Because the system is busy
  • qstat -a
Troubleshooting: Why did X’s job run before mine?

• Because you’re unimportant or you’ve been hogging the system (fairshare)
• Because their job was smaller/shorter and was backfilled
• Because your job asked for something that was unavailable
  • Consumable resources
  • Special nodes
  • Placement sets
Troubleshooting: Fairshare

# pbs_fs

Fairshare usage units are in: cput
TREEROOT : Grp:  -1  cgrp:    0  Shares:  -1  Usage:14141166288 Perc:  100.000%
facilities: Grp:    0  cgrp:    2  Shares: 100000 Usage: 14141165485 Perc:  99.990%
        ASD : Grp:     2  cgrp:  271  Shares: 5000000 Usage:  5301136879 Perc:  49.995%
ACGD0005  : Grp:  278  cgrp:  284  Shares:   1 Usage:  791019631 Perc:  4.166%
ACGD0004  : Grp:  278  cgrp:  283  Shares:   1 Usage:    0 Perc:  4.166%
AACD0002  : Grp:  278  cgrp:  282  Shares:   1 Usage:   1854 Perc:  4.166%
ARAL0001  : Grp:  278  cgrp:  281  Shares:   1 Usage:    785 Perc:  4.166%
AHAO0001  : Grp:  278  cgrp:  280  Shares:   1 Usage:     421 Perc:  4.166%
Troubleshooting: No Output

• usecp
  • Was submission host alive?

• Check disks - is there space/quota available for output files?

• Is the job done? qstat -f [jobid]
Troubleshooting: Diagnostic collection

- `$PBS_EXEC/unsupported/pbs_diag`
  - Produces a tarball for PBS support
- `tracejob [jobid]`
  - Finds information about a job from the server log
  - Crippled if you’re using syslog (only)
- Increase the log-level (different for each component – see docs)
- Look for core files in `$PBS_HOME`
- If the server is hung, ”gstack [pid]” a few times to figure out what it’s doing
Analytics

• Very limited support in PBS
• PBS Analytics
  • Web based reporting using the accounting data produced by PBS
  • Commercial product
• XDMoD
  • NSF sponsored web-based reporting tool
  • Some Limited PBS (and SLURM) support
  • Integration with other data sources
• Gold
  • Open (PNNL) or Commercial (Adaptive) but no recent releases
Reservations

- Two kinds: Standing and advanced
- Standing reservations reserve some resources on a regular basis
  - ical syntax
- Advanced reservations reserve some resources at a specific future time
- Reservations are queues
- Reservations will disappear if they can not be fulfilled
- Reservations for offline nodes may be reconfirmed on other nodes up to a configurable time before the start
Reservations

- Nodes that are offline in a running reservation are not replaced.
- Reservation queues are numerical and have a configurable prefix which indicates their type:
  - R123456
  - S123457
- Names can be attached to reservations, but they aren’t really used.
- Generally, reservation queues have an ACL that restricts who can submit to them.
Reservations

• Common States:
  • Running – reservation can run jobs
  • Confirmed – resources are allocated, but the reservation hasn’t started yet
  • Degraded – reservation is running, but some reserved resource is unavailable
  • Unconfirmed – scheduler is still looking for resources
Reservations

# pbs_rsub -l select=1:ncpus=72 -l place=free -R 1300 -E 1400 R311175.laadmin1.ib0.laramie.ucar.edu UNCONFIRMED
# pbs_rstat -f R311175.laadmin1.ib0.laramie.ucar.edu
Resv_ID: R311175.laadmin1.ib0.laramie.ucar.edu
Reserve_Name = NULL
Reserve_Owner = pbsdata@laadmin1.ib0.laramie.ucar.edu
reserve_state = RESV_CONFIRMED
reserve_substate = 2
reserve_start = Thu Apr 27 13:00:00 2017
reserve_end = Thu Apr 27 14:00:00 2017
reserve_duration = 3600
queue = R311175
Resource_List.ncpus = 72
Resource_List.walltime = 01:00:00
Resource_List.nodect = 1
Resource_List.select = 1:ncpus=72
resv_nodes = (r1i0n1:ncpus=72)
Authorized_Users = pbsdata@laadmin1.ib0.laramie.ucar.edu
server = laadmin1.ib0.laramie.ucar.edu
mtime = Thu Apr 27 11:48:43 2017
time = Thu Apr 27 11:48:43 2017
Variable_List = PBS_O_LOGNAME=pbsdata,PBS_O_HOST=laadmin1.ib0.laramie.ucar.edu,PBS_O_MAIL=/var/spool/mail/pbsdata
Preemption

• If a more important job comes along, signal an existing job to make room
• Up to each user application to checkpoint (quickly) and terminate (or be kill - 9’ed)
• Not used too much at academic HPC sites
  • Not much support from common applications
Support

• Community support forum on http://pbspro.org/
• PBS User’s Group (This week in Las Vegas ;-((
• Jira bug tracker: https://pbspro.atlassian.net/secure/Dashboard.jspa
Questions?