

Queuing and Scheduling on Compute Clusters

Andrew Caird

acaird@umich.edu

The reason for me being here

- Give some queuing background
- Introduce some queuing ideas
- Talk about what we do at the Center for Advanced Computing

CAC

The Center for Advanced Computing

- 3 full-time employees, 1 student

CAC

The Center for Advanced Computing

- 3 full-time employees, 1 student
- 4 clusters, ~580 dual-processor nodes
- largest cluster is called *nyx*
 - 256 nodes, 6 partitions: 5 private, 1 general
 - ~150 jobs scheduled per day for 150 people
 - ~50 jobs per day in the general partition for 100 people

CAC

The Center for Advanced Computing

- 3 full-time employees, 1 student
- 4 clusters, ~580 dual-processor nodes
- largest cluster is called *nyx*
 - 256 nodes, 6 partitions: 5 private, 1 general
 - ~150 jobs scheduled per day for 150 people
 - ~50 jobs per day in the general partition for 100 people
- Either we know what we're doing, or we're making an enormous disaster

The Problem

- efficient use of resources
 - On a 256 node cluster, a 10% increase in efficiency is 25 nodes, or about \$50,000/year in hardware and staff costs
- guaranteeing the owners of private nodes access and the same efficiency

The Resource Manager

- starts and stops jobs
- stage-in/-out of files
- used by end-users to submit jobs, check job status
- manages queues
- can manage resources
- usually comes with some sort of scheduler

The Scheduler

- tells resource manager which jobs to start and stop
- maintains job properties (class, QoS, etc.)
- schedules jobs based on information from the resource manager and its own info
- can manage resources
- sets and manages reservations

Queuing Products

- PBSPro from Altair (www.pbspro.com)
 - We use this on Linux and MacOS
 - Very stable, scales well
 - Opaque scheduler, poor interface with other schedulers

Queuing Products

- PBSPro from Altair (www.pbspro.com)
 - We use this on Linux and MacOS
 - Very stable, scales well
 - Opaque scheduler, poor interface with other schedulers
- Torque (was OpenPBS)
(www.clusterresources.com)
 - We used this on Linux
 - Much more stable than in the past; lots of recent development
 - Only a simple scheduler, great interface to Maui (same developers)

Queuing Products

- Maui / Moab (only schedulers)
(www.clusterresources.com)
 - Maui: free, Moab: commercial
 - very transparent, very configurable

Queuing Products

- Maui / Moab (only schedulers)
(www.clusterresources.com)
 - Maui: free, Moab: commercial
 - very transparent, very configurable
- Load Sharing Facility (LSF) from Platform Computing (www.platform.com)
- Sun Grid Engine (SGE) open-sourced from Sun
(gridengine.sunsource.net)
- LoadLeveler from IBM
(www.ibm.com/servers/eserver/clusters/software/loadleveler.html)

Our scheduling concepts

Scheduling Concepts:

- Resource Manager dependent or Scheduler dependent
- queue permissions, node-to-queue mapping, etc. is controlled by one or the other

Resource Manager Dependant

Resource Manager Dependant (RMD) Queuing

- RM has node-to-queue mapping
- RM has collection-of-nodes information (partitions)
- scheduler gets its info from resource manager
- Advantages: more firm, less likely to be out of control, easier to understand
- Disadvantages: not as flexible as scheduler-controlled model

Scheduler Dependant

Scheduler Dependant (SD) Queuing

- Scheduler has node-to-queue mapping
- Scheduler has collections-of-nodes information (standing res)
- Advantages: more flexible; configuration all in one place (scheduler config); giving the scheduler more control can lead to more efficient use of resources
- Disadvantages: more complicated; less transparent to end users, can be confusing to them

SD vs RMD

- for “simple” clusters, RMD is fine
- SD more appropriate for large, multi-user clusters

Scheduling Concepts

- FIFO
- priority calculations (queue time, fairshare, class)

Scheduling Concepts

- FIFO
- priority calculations (queue time, fairshare, class)
- reservations: standing, one-time, and job
- backfill

Scheduling Concepts

- FIFO
- priority calculations (queue time, fairshare, class)
- reservations: standing, one-time, and job
- backfill
- resource usage limits, resource types
- maximum number of running and idle jobs

Examples from the CAC: RMD

We use resource manager dependent scheduling on our largest cluster, `nyx` and our Mac cluster, `eliza`.

- with a mix of standing and one-time reservations
- this is clearer for the users (they are sure they are getting “their” nodes)
- `nyx` is the last Linux cluster to be upgraded; we’ll move to scheduler dependent scheduling when we upgrade
- `eliza` may become part of `aon` (and `aon`’s scheduling policies), or may get SD scheduling on its own, or may just live out its life as it stands today

Examples from the CAC: SD

We use scheduler dependent scheduling on our other two cluster, the Athlon+Myrinet cluster morpheus and the private faculty-owned cluster aon.

- the faculty asked for scheduler dependent scheduling in order to enforce group limits while also allowing for the most efficient use of their cluster. There are no partitions or reservations, just queue limits; this works best on a homogeneous cluster
- we use a different type of SD scheduling on morpheus; it is based on standing reservations where the “private” reservations can spill into the public pool of nodes, but not (yet) the other way.

Things we don't or won't do

- preemption
 - we will probably do this to allow use of the private nodes
 - it will be a user-requested feature, for those punks who are feeling lucky

Things we don't or won't do

- preemption
 - we will probably do this to allow use of the private nodes
 - it will be a user-requested feature, for those punks who are feeling lucky
- checkpoint/restart
 - not practical yet in Linux or OS X
 - that said, there was a surprising (to me) amount of talk about this at SuperComputing (see sc05.supercomp.org/schedule)

Things we don't or won't do

- preemption
 - we will probably do this to allow use of the private nodes
 - it will be a user-requested feature, for those punks who are feeling lucky
- checkpoint/restart
 - not practical yet in Linux or OS X
 - that said, there was a surprising (to me) amount of talk about this at SuperComputing (see sc05.supercomp.org/schedule)
- suspend/restart
 - memory/swap issues
 - local storage issues

The End

Questions?

<http://cac.engin.umich.edu>