GridPP

UK Computing for Particle Physics

Advanced batch schedular configuration

Stuart Purdie

University of Glasgow

Experiment logo here

Hats gag?

Hats gag?

Hats gag?

Your university or
experiment logo here

Hats gag?

Your university or
experiment logo here

Hats gag?

Hats gag?

Hats gag?

Your university or experiment logo here

Overview

- Role of the schedular
- General principles of operation
  - Reservation
  - Priority
  - Throttling
- Principles of schedular configuration
- Special features
  - Preemption
  - Job start throttling

Some opening remarks

- Primarily Maui; but same principles, and therefore tuning ideas applicable to all batch schedulers I've worked with.
  - Terminology varies however, and I've gone with the Maui ones.
  - SGE users feel free to point out SGE terms where appropriate

Some opening remarks

- Primarily Maui; but same principles, and therefore tuning ideas applicable to all batch schedulers I've worked with.
  - Terminology varies however, and I've gone with the Maui ones.
  - SGE users feel free to point out SGE terms where appropriate
- We looked at switching to SGE at Glasgow, but with the uncertainty for long term support, we felt the pain of change would probably outweigh the benefits
  - Also, I fixed the key aspects pushing us away from Torque/Maui!

Some opening remarks

- Primarily Maui; but same principles, and therefore tuning ideas applicable to all batch schedulers I've worked with.
  - Terminology varies however, and I've gone with the Maui ones.
  - SGE users feel free to point out SGE terms where appropriate
- We looked at switching to SGE at Glasgow, but with the uncertainty for long term support, we felt the pain of change would probably outweigh the benefits
  - Also, I fixed the key aspects pushing us away from Torque/Maui!
- Aim to give people the set of knowledge and tools so they can work out how to optimise the schedular for the needs of their cluster, rather than copy one from a cookbook.

The role of a schedular

- Take information in
- Do some sums
- Start some jobs
- Pause
- Repeat

Risk analysis

- Mucking about risk: LOW

Risk analysis

- Mucking about risk: LOW
- Totally stuffing it up risk: LOW

Risk analysis

- Mucking about risk: LOW
- Totally stuffing it up risk: LOW
- That I'll be held to these claims risk: LOW

Those sums?

- Three main steps
  - Job throttling
  - Job priority calculation
  - Job eligibility (resource requirements)

Those sums?

- Three main steps
  - Job throttling
  - Job priority calculation
  - Job eligibility (resource requirements)
- Generally works by iterations
  - Each iteration, evaluate valid jobs
  - Calculate a priority value for each valid job
  - Walk down the list, starting each job whose resources can be met

Those sums?

- Three main steps
  - Job throttling
  - Job priority calculation
  - Job eligibility (resource requirements)
- Generally works by iterations
  - Each iteration, evaluate valid jobs
  - Calculate a priority value for each valid job
  - Walk down the list, starting each job whose resources can be met
- Also resource distribution
  - What job to what slot, more important when quieter, or heterogenous

Except...

- The probabilities of meeting the resources needed for a multi-core job are low
  - Very, very low; unless empty

Except...

- The probabilities of meeting the resources needed for a multi-core job are low
  - Very, very low; unless empty
- So practical multi-core schedulers use advance reservations to generate a suitable number of slots for multi-core jobs
  - This reduces throughput, but allows multicore jobs to be mixed

Except...

- The probabilities of meeting the resources needed for a multi-core job are low
  - Very, very low; unless empty
- So practical multi-core schedulers use advance reservations to generate a suitable number of slots for multi-core jobs
  - This reduces throughput, but allows multicore jobs to be mixed
- These reservations keep job slots clear for a while.

Except...

- The probabilities of meeting the resources needed for a multi-core job are low
  - Very, very low; unless empty
- So practical multi-core schedulers use advance reservations to generate a suitable number of slots for multi-core jobs
  - This reduces throughput, but allows multicore jobs to be mixed
- These reservations keep job slots clear for a while.
- Jobs shorter than the time till the start of the reservation can be fitted in front of the reservation if there's a gap
  - This is called 'Backfill'

Credentials

- Restrictions and calculations can be based on multiple factors – many concern the “owner” of the job.
- Maui calls these 'credentials' and recognises that a job can belong to several of these
  - User
  - Group (meaning unix group)
  - Account (rarely used in HEP or Grid environments)
  - QOS (Quality of Service class)
  - Queue (Maui calls this 'class')
- And one more pseudo class, SYSTEM, for everything.

Throttling

- Limits are set per credential
- Limits (hard and 'soft') on
  - Jobs running (MAXJOBS), Processors (cores) used (MAXPROCS)
    - Can be different with multi-core jobs
  - Nodes (MAXNODES)
  - CPUTime / Wall time of credentials scheduled jobs (MAXPS / MAXWC)
  - Memory allocated to credentials job (MAXMEM)
- Less often used ones:
  - Limits on number of queued jobs / cores
    - Doesn't stop Torque accepting new jobs, but prevents them from being considered at all – no accumulation of queue time etc
    - Can use all the various throttling types on queued jobs
- Same step also excludes un-runable jobs
  - Job holds, invalid states (Running, Deferred etc), pending stage-in and job dependancies

Priority

- Job priority is a number calculated from a number of factors about the job
- Typical factors include
  - Credential based stuff
  - Job resources
  - Resources recently consumed by the credentials
    - i.e. the Fairshare
  - Current service levels
    - Queuetime etc
  - Target service levels
  - Usage

Priority - Fairshare

- Fairshare may be specified per Credential
- Fairshare calculations use: (target – current) * weight
- Typically across multiple credentials
- With maui, diagnose -f to see fairshares
- Usage considered for fairshare is help over a rolling window
  - N period of X duration, with possible decay
- Decay is useful, as it gives a gentle tail off, which prevents bang-bang regulation
  - i.e. all Altas one day, all LHCb the next.

```
FSPOLICY        DEDICATEDPES
FSDEPTH         6
FSINTERVAL       4:00:00
FSDECAY         0.95
```

Resource requirements

- By this point, handling resource requirements is near trivial
- Walk down the priority list, starting the jobs in priority order that can meet requirements
- In event of a conflict, higher priority job gets the needed resource
- Not going to go into detail on the reservation algorithm for multicore jobs...
  - But, essentially, when a multicore job comes in, it puts in reservations so that it can start when all the jobs ahead of it in priority are done.
  - Maui will hold RESERVATIONDEPTH (default 1) reservations

- And that's the basics of how it all works.
  - Note that it primarily affects job starts.
- Now to talk about some tuning, and tweaking data.

- And that's the basics of how it all works.
  - Note that it primarily affects job starts.

Large cluster tuning

- NODEPOLLFREQUENCY (default 0; glasgow has 3)
  - How many schedular iterations before telling torque to poll nodes
- JOBAGGREGATIONTIME (default 0; glasgow has 4)
  - Delay between scheduling iterations when new jobs present
- RMPOLLINTERVAL (default 60; glasgow has 60)
  - Delay between scheduling iterations when no new jobs
- RESDEPTH (default 24)
  - Number of reservations on a node
    - "To be on the safe side, this value should be approximately twice the average sum of admin, standing, and job reservations present."
  - Multi core jobs increase demand on this
  - Appears to be non-fatal if it exceeds -might impact multicore scheduling
  - Appears to have no detriment, beyond a few kilobytes of ram

Partitions

- Maui won't schedule one job across multiple partitions
  - Useful for scheduling multicore with heterogenous nodes
- We have 2 partitions, used by everything by default
  - GROUPCFG[DEFAULT] PLIST=DEFAULT:cvold
- We keep analysis off the oldest class of nodes
  - GROUPCFG[atlas] PLIST=DEFAULT&
  - GROUPCFG[atlaspil] PLIST=DEFAULT&
- The '&' means to AND with other partition access, without it the default is a logical OR

- Partition definition:
  - NODECFG[node052] SPEED=1.82
  - NODECFG[node060] SPEED=0.94 PARTITION=cvold

Priority tuning

- Priority is a value from 0 to 1000000000
  - This leaves some headroom for setpri, for manually running jobs
- Most configurations don't use this range, instead a tiny part of it
- Make sure that your weights are large enough to use a good part of that space – no point cramming everything in a 0-50 range!
- There are a range of places where a weighting can be increased – the best ones to use will vary by the priority algorithm
- I throughly recommend writing out the precise priority formula for your site, containing only the options you use

Glasgow's current algorithm

Priority = CREDWEIGHT (1)
* GROUPWEIGHT (1000) * GROUP-PRIORITY (5 – 2000)
+ FSWEIGHT (1)
* FSGROUPWEIGHT (80) * GROUP-FS (0.01 – 37)
+ XFACTORWEIGHT (200) * XFACTOR (1+)

```
CREDWEIGHT              1
GROUPWEIGHT             1000

GROUPCFG[DEFAULT]       PRIORITY=5
FSWEIGHT                1
FSGROUPWEIGHT           80

GROUPCFG[atlasprd]      FSTARGET=35     MAXPROC=2200,2200
GROUPCFG[atlaspil]      FSTARGET=35     MAXPROC=400,400
GROUPCFG[atlassgm]      PRIORITY=1000   MAXPROC=10,10
GROUPCFG[biomed]        FSTARGET=0.1    MAXPROC=200,2000

XFACTORWEIGHT           200
```

Analysis

- In the normal case...

Priority = CREDWEIGHT (1)
* GROUPWEIGHT (1000) * GROUP-PRIORITY (5 – 2000)
+ FSWEIGHT (1)
* FSGROUPWEIGHT (80) * GROUP-FS (0.01 – 37)
+ XFACTORWEIGHT (200) * XFACTOR (1+)

Analysis

- In the normal case...
- Min Priority
  - = 1000 * 5 + 1 * 80 * -65 + 200
  - = 5000 – 4480 + 200
  - = 720

Priority = CREDWEIGHT (1)
* GROUPWEIGHT (1000) * GROUP-PRIORITY (5 – 2000)
+ FSWEIGHT (1)
* FSGROUPWEIGHT (80) * GROUP-FS (0.01 – 37)
+ XFACTORWEIGHT (200) * XFACTOR (1+)

Analysis

- In the normal case...
- Min Priority
  - = 1000 * 5 + 1 * 80 * -65 + 200
  - = 5000 – 4480 + 200
  - = 720
- Max Priority
  - = 1000 * 5 + 1 * 80 * 35 + 400
  - = 5000 + 2800 + 400
  - = 8200

Priority = CREDWEIGHT (1)
* GROUPWEIGHT (1000) * GROUP-PRIORITY (5 – 2000)
+ FSWEIGHT (1)
* FSGROUPWEIGHT (80) * GROUP-FS (0.01 – 37)
+ XFACTORWEIGHT (200) * XFACTOR (1+)

Overfairshare issues

- Consider a case of:
  - GROUPCFG[atlasprd] FSTARGET=35
  - GROUPCFG[biomed] FSTARGET=0.1
- What should happen when the current usage for each VO is 5 over fairshare target, and only jobs from those groups queued?
- That depends on the site poilcies
  - But it probably shouldn't be a free for all!
- The fairshare calculation does:
  - FSWEIGHT * (currentUsage – FSTARGET)
  - So overtarget has a negative contribution

Overfairshare issues 2

- If jobs are sufficiently over fairshare that the total job priority hits 0 …
  - Then it's random distribution at that point
- This is an issue if one is very over fairshare, and one is just over fairshare
- Simple fix is to make sure that the minimum priority is higher than 0!

```
FSWEIGHT
 * (FSUSERWEIGHT        * (FSTARGET – 100)
   + FSGROUPWEIGHT     * (FSTARGET – 100)
   + FSQOSWEIGHT       * (FSTARGET – 100) )
                    >=
MAX(All other priority components)
```

Priority capping

- Sometimes we want to limit how much a particular component can add
  - e.g. maximum benefit from XFACTOR or QUEUETIME
- Maui calls these >FOO>CAP
  - XFACTORCAP for example

- Capping FAIRSHARE (FSCAP) can help distribution between large and small FS users.

Priority capping

- Sometimes we want to limit how much a particular component can add
  - e.g. maximum benefit from XFACTOR or QUEUETIME
- Maui calls these >FOO>CAP
  - XFACTORCAP for example

Preemption

- What if you want long running low priority jobs?
- Without impeding short term analysis?
  - Which is impossible, without interrupting the longer jobs
    - Or being precognitive...
- So the only practical option is to interrupt long jobs with the shorter ones
- Not so great a plan for Grid jobs – which tend to assume network availability

Preemption - 2

- PREEMPTPOLICY (default: REQUEUE)
  - Probably best to be SUSPEND; CHECKPOINT is hard to support
- Need to use QOS features, even if they have no effect on priority
- Set some(s) (presumably long) queue to a QOS class with PREMPTEE set
- Set another(s) (presumably short) queue to a QOS class with PREMPTOR set
- Pour beer

Throttling users

- In particular, job start rate
- Not a direct option with Maui
- MAXIJOBS / MAXIPROCS will have this effect, assuming you don't have QUEUETIME / XFACTOR play a strong role
- As they apply before the priority calculation, the ineligible jobs miss a scheduling iteration
  - And hence the maximum started per scheduling iteration is the total eligable
- The actual effect depends strongly on what JOBAGGREGATIONTIME and RMPOLLINTERVAL are set to

GROUPCFG[alice] MAXIJOBS=1

Maui's FEEDBACKPROGRAM

- FEEDBACKPROGRAM <progname>
- Runs the program on completion of a job
- One presumes that it passes some data to the program, but I can't find any details on that
- It's on my list to check precisely what it does

Glossary

- Credential
  - The 'owner' of a job – generic term encompassing the User, Group, Account, QOS and Class. Also implies a pseudo user, SYSTEM, for defaults.
- Class
  - Torque (and everyone else!) calls these queues.