Condor at the RACF

Migration to 7.4, Group Quotas, and More

William Strecker-Kellogg
Brookhaven National Lab
RHIC/ATLAS Computing Facility Overview

- Physics Dept. at Brookhaven National Lab—provides computing and storage to active RHIC experiments
- Serves as a Teir-1 for ATLAS computing
- Uses Condor to manage RHIC and ATLAS compute clusters
- 14.4k cores running SL5.3 currently
- With new hyper-threaded 12-core Westmere systems, to grow to over 20k cores
One instance for ATLAS
- 5100 slots
- 2 submit nodes manage all production/analysis jobs
- Other smaller queues managed on 3 other submit nodes

Instance each for STAR and PHENIX experiments
- 4300, 4500 slots resp.
- 20 submit nodes each
- “General Queue”—flocking between RHIC pools
- Smaller experiments grouped into another instance
New Since Last Year

- New condor administrator
- Migration to 7.4.2
  - Up from 6.8.9—long overdue
- Move ATLAS to Group Quotas
  - Easier configuration—from 16 configuration files to 5—90% of slots use just 1)
  - Management via web-interface
  - Some problems we’ve had to solve...more later
Upgrade to 7.4.2

- Get rid of suspension model
- Undesirable to have slots != real cores
- Simplify START expression
- Better negotiator performance, results later
- Bugfixes all around

Example START Expression

Start = (((PrimExpt == 'atlas') && (VMachineID > 7) && (TARGET.RAFC_Group == 'short')) ||
TARGET.RAFC_Group == 'prod' ||
TARGET.RAFC_Group == 'distr_analysis' ||
TARGET.RAFC_Group == 'short') ||
TARGET.RealExperiment == 'atlas' && false) ||
TARGET.RAFC_Group == 'prod';

From this

To this
Group Quotas

- ATLAS only, for now
  - PHENIX to follow suit in a few months
  - No plans for STAR
- What groups buy us:
  - Manage ATLAS production/analysis jobs separately from many other smaller queues
  - Unify configuration files—one config for vast majority of ATLAS nodes
ATLAS Group Quotas

Reallocation of resources between queues managed via web interface

A Day in the Life of ATLAS
Issues Moving to Group Quotas

- Backwards compatibility with our accounting and monitoring systems
  - Solution: Retain previous job-type flag that used to hard-partition slots
- How does it interact with flocking?
- Fairness / Enforcement of group memberships
  - “We rely on societal enforcement”
- Not good enough...solution for ATLAS
  - ATLAS uses PANDA, we control local submission
  - Other users few enough to monitor individually
Issues Moving to Group Quotas

- PHENIX: two classes—user and special jobs
  - Special jobs submitted from few machines, separate users
  - User jobs from 20 submit nodes
- Two solutions
  - Submit node based partition: regex-match GlobalJobID ClassAd against list of valid sources in START expr.
  - Coexist with users: three configs, user nodes w/ no AccountingGroup flag, shared nodes that run anything but are ranked last by user and special jobs, and special nodes requiring AG flag
Group Priorities

- ATLAS setup: three main groups
  1) Production: highest prio., hard quota, no preemption
  2) Short analysis: medium prio., auto-regroup on, preemption enabled
  3) Long analysis: lowest prio., auto-regroup on, preemption enabled

- Idea—short and long analysis spill over into each other as needed and not be squeezed out by production
- Problem—sometimes short and long will “eat into” production even when they are over-quota and production is under its quota
ATLAS Priority Inversion

- Group-priority affects only order of negotiation
- When an analysis queue starts up after a quiet period, production starts to lose out. **Even though production is below its quota it loses slots to analysis jobs because they get negotiated for first.**
- Negotiation should stop for a queue that is over quota (w/ auto-regroup on) and there are other queues with waiting jobs below their quotas.

Problem area
ATLAS Priority Inversion

- Solution? *Increasing the spread of priority factors as more lots get added to production.* Required spread scales with size of the largest queue, and if another queue quiesces for long enough it will outrank production.

- E.g. Production goes from 3k to 4k slots: usage increases 33% making its priority that much worse and an inversion that much more likely to occur...
Negotiator Performance

Cycle occasionally blocks waiting for a schedd (many successes near the default 30s timeout)

In case where many submitters are on many machines each, much wasted time
Issues with Scheduler/Negotiator

- User frequently polling large queue
  - Schedd would fork a child which would use 5-10s of CPU time to answer query (1.6Gb Size!)
- Auto-clustering sometimes doesn’t skip similar jobs
- Globally-scoped ClassAds would be nice, e.g. for the usage of a shared scratch NFS filesystem
Puppet-ize Configuration Files

- New Puppet-based centralized configuration management system of general-purpose servers
- Will templatize condor configuration

```ruby
class ldap {
  # first make sure nscd is included:
  Include nscd
case $nldap_domain {
  $ldap_home_dir = 'homeDirectoryAtlas'
  }

  RHCI: {
  }
  default: []
  }

case $operatingsystem {
  default: {
case $operatingsystem release {
    */2/:
    $package_names = [ 'nss LDAP' ]
    $ldap_template = '/etc/ldap/ldap.conf.rhel5.erb'
    $ldap_conf_path = '/etc/ldap.conf'
    $nsswitch_file = 'puppet:///modules/nsswitch.conf.rhel5'
    $service_names = []
    }
    #!!!: can use either nss-pam-ldap or sssd!
    $package_names = [ 'nss-pam-ldap', 'sso' ]
    $ldap_template = '/etc/nslcd.conf.rhel6.erb'
    $ldap_conf_path = '/etc/nsswitch.conf' # nsswitch.conf
    $nsswitch_file = 'puppet:///modules/nsswitch.conf.rhel6' # Replace ldap with sss in this file!
    $service_names = [ 'nslcd' ] # sssd
    }
    */3/: {
    #!!!: can use either nss-pam-ldap or sssd!
    $package_names = ['']
    $ldap_template = ''
    $ldap_conf_path = ''
    $nsswitch_file = ''
    $service_names = []
    }
  }
}
```
Motivation to use Puppet

- Configuration is similar in structure between experiments
  - Memory Limits for regular and flocked jobs
  - Preemption/Retirement-time on a per-job-type basis
  - Policy expressions (RANK/START/etc...)
  - List of currently blocked users
- Recent blocking/unblocking of users took editing 6 different files and a reconfig everywhere
- Using Puppet would separate each logical entity, making it easy to change things on a per-entity basis, and would automate pushing of changes and reconfiguration. All changes versioned in git—accountability and reliability
Thanks!

Questions? Comments?
CRS Job System

- Written in Python, submits to condor
- Asynchronous IO for staging data from tape
- Stages in and out are done outside of condor
  - Previously done with extra slots, not good aesthetically and otherwise
  - Can combine stage requests for jobs intelligently
- Abstraction layer for IO, similar to plugins
- Own basic workflow tools—DAGs not suitable
ATLAS Shared Pool

- Allow smaller ATLAS queues and OSG grid jobs to run in their own queue that can utilize a small shared pool of resources.
- Implemented with Group Quotas.
- Jobs “compete” for extra slots made available by ATLAS.
- Necessitates adding AG flag by users (small enough it works).