GridPP

UK Computing for Particle Physics

HTCondor at the RAL Tier-1

Andrew Lahiff, Alastair Dewhurst,
John Kelly, Ian Collier

pre-GDB on Batch Systems
11 March 2014, Bologna

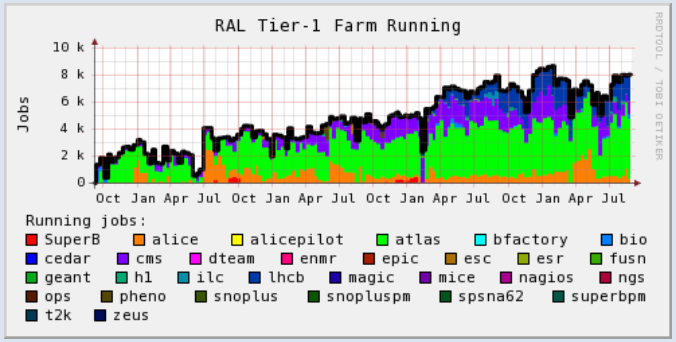
Outline

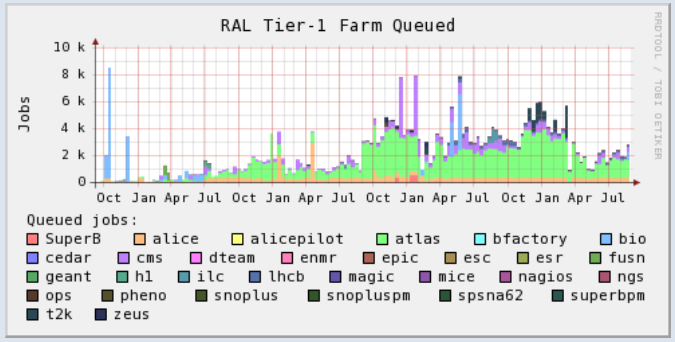
1. RAL batch system
   - Background
   - Why we migrated to HTCondor
   - Experience so far
2. In detail
   - Compatibility with middleware
   - Installation & configuration
   - Queues
   - Fairshares
   - Scalability
   - Support
   - Multi-core jobs
   - Dynamic WNs

RAL batch system

RAL batch system

- Batch system at the RAL Tier-1
  - 656 worker nodes, 9312 slots, 93000 HEPSPEC06
  - Growing soon to beyond 12000 slots
- VOs supported
  - All LHC experiments. RAL provides:
    - 2% of ALICE T1 requirements
    - 13% of ATLAS T1 requirements
    - 8% of CMS T1 requirements
    - 19% of LHCb T1 requirements (will grow to 30%)
  - Many non-LHC experiments, including non-HEP
  - No local job submission – only via grid





Torque + Maui

- Torque + Maui had been used for many years at RAL
- Many issues
  - Severity and number of problems increased as size of farm increased
- Problems included
  - pbs_server, maui sometimes unresponsive
  - pbs_server needed to be restarted sometimes due to excessive memory usage
  - Job start rate sometimes not high enough to keep the farm full
    - Regularly had times when had many idle jobs but farm not full
  - Regular job submission failures on CEs - *Connection timed out-qsub: cannot connect to server*
  - Unable to schedule jobs to the whole-node queue
    - We wrote our own simple scheduler for this, running in parallel to Maui
  - Didn't handle mixed farm with SL5 and SL6 nodes well
  - DNS issues, network issues & problematic worker nodes cause it to become very unhappy
- Increasing effort just to keep it working

Choosing a new batch system

- In August 2012 started looking for an alternative – criteria:
  - Integration with WLCG community
    - Compatabile with grid middleware
    - APEL accounting
  - Integration with our environment
    - E.g. Does it require a shared filesystem?
  - Scalability
    - Number of worker nodes
    - Number of cores
    - Number of jobs per day
    - Number of running, pending jobs
  - Robustness
    - Effect of problematic WNs on batch server
    - Effect if batch server is down
    - Effect of other problems (e.g. network issues)
  - Support
  - Procurement cost
    - Licenses, support
    - Avoid commercial products if at all possible
  - Maintenance cost
    - FTE required to keep it running
  - Essential functionality
    - Hierarchical fairshares
    - Ability to limit resources
    - Ability to schedule multi-core jobs
    - Ability to place limits on numbers of running jobs for different users, groups, VOs
  - Desirable functionality
    - High availability
    - Ability to handle dynamic resources
    - Power management
    - IPv6 compatibility

Choosing a new batch system

- Considered, tested & eventually rejected the following technologies:
  - LSF, Univa Grid Engine
    - Avoid commercial products unless absolutely necessary
  - Open-source Grid Engines
    - Competing products, not sure which has best long-term future
    - Communities appear less active than SLURM & HTCondor
    - Existing Tier-1s using Univa Grid Engine
  - Torque 4 + Maui
    - Maui problematic
    - Torque 4 seems less scalable than alternatives
  - SLURM
    - Carried out extensive testing and comparison with HTCondor
    - Found that for our use case:
      - Very fragile, easy to break
      - Unable to get to work reliably above 6000 jobs slots
- For more information, see

  http://indico.cern.ch/event/247864/session/5/contribution/21

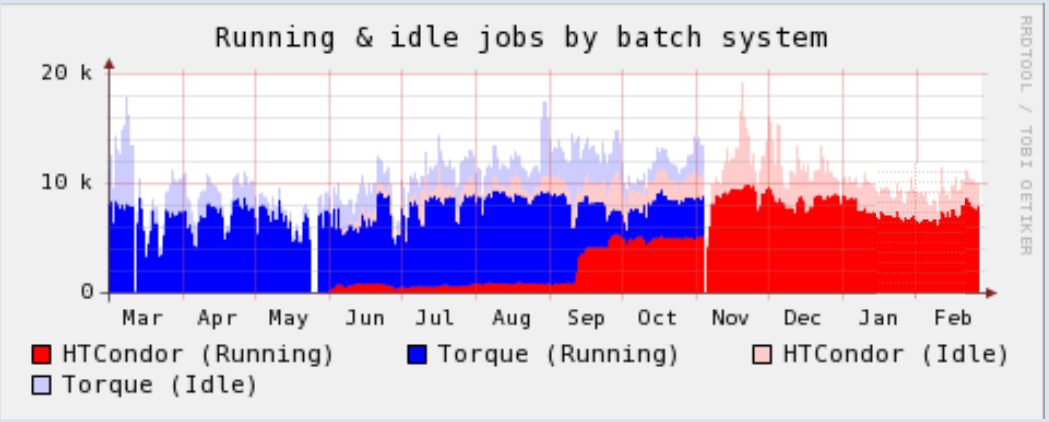
Choosing a new batch system

- HTCondor chosen as replacement for Torque + Maui
  - Has the features we require
  - Seems very stable
  - Easily able to run 16,000 simultaneous jobs
    - Prior to deployment into production we didn't try larger numbers of jobs
      - Didn't expect to exceed this number of slots within the next few years
    - Didn't do any tuning – it "just worked"

Migration to HTCondor

- Timeline

2012 Aug	- Started evaluating alternatives to Torque/Maui
2013 June	- Began testing HTCondor with ATLAS & CMS
2013 Aug	- Choice of HTCondor approved by RAL Tier-1 management
2013 Sept	- Declared HTCondor as a production service
	- Moved 50% of pledged CPU resources to HTCondor
	(upgraded WNs to SL6 as well as migrating to HTCondor)
2013 Nov	- Migrated remaining resources to HTCondor



Experience so far

- Current setup
  - 8.0.6 on central managers (high-availability pair), CEs
  - 8.0.4 on worker nodes
  - Using 3 ARC CEs, 2 CREAM CEs
- Experience
  - Very stable operation, no crashes or memory leaks
  - Job start rate much higher than Torque/Maui, even when throttled
  - Staff able to spend time investigating improvements/new features, not just fire-fighting

In detail

Compatibility with Middleware

- EMI-3 CREAM CE
  - HTCondor not officially supported
    - BLAH supports HTCondor
      - Job submission works!
    - HTCondor support in YAIM doesn't exist in EMI-3
      - We modified the appropriate YAIM function so that the blah configuration file is generated correctly
    - Script for publishing dynamic information doesn't exist in EMI-3
      - Wrote our own based on the scripts in old CREAM Ces
      - Updated to support partitionable slots
    - APEL parser for HTCondor doesn't exist in EMI-3
      - Wrote a script which writes PBS style accounting records from condor history files, which are then read by PBS APEL parser
  - Relatively straightforward to get an EMI-3 CREAM CE working
    - We will make our scripts available to the community
    - Milan Tier-2 also helpful

Compatibility with Middleware

- ARC CE
  - Successfully being used by some ATLAS & CMS Tier-2s outside of Nordugrid (with SLURM, Grid Engine, …)
    - LRZ-LMU, Estonia Tier 2, Imperial College, Glasgow
  - Benefits of ARC CEs
    - Support HTCondor better than CREAM CEs do
    - Simpler than CREAM CEs
      - No YAIM
      - No Tomcat
      - No MySQL
    - ARC CE accounting publisher (JURA) can send accounting records directly to APEL using SSM
      - APEL publisher node not required
  - The LHC VOs and ARC CEs
    - ATLAS and CMS fine
      - At RAL ATLAS & CMS only have access to ARC CEs
    - LHCb added ability to DIRAC to submit to ARC CEs
      - Not yet at the point of LHCb only using ARC CEs
    - ALICE can't use them yet, but will work on this

Initial install

- Most basic install + configuration is trivial
  - Time between basic SL6 machine & running jobs = time taken for yum to run

```
[root@lcg0732 ~]# yum install condor
…
[root@lcg0732 ~]# service condor start
Starting up Condor...    done.
[root@lcg0732 ~]# condor_status -any
MyType             TargetType         Name

Collector          None               Personal Condor at lcg0732.gridpp.rl.ac.u
Scheduler          None               lcg0732.gridpp.rl.ac.uk
DaemonMaster       None               lcg0732.gridpp.rl.ac.uk
Negotiator         None               lcg0732.gridpp.rl.ac.uk
Machine            Job                slot1@lcg0732.gridpp.rl.ac.uk
Machine            Job                slot2@lcg0732.gridpp.rl.ac.uk
Machine            Job                slot3@lcg0732.gridpp.rl.ac.uk
Machine            Job                slot4@lcg0732.gridpp.rl.ac.uk
Machine            Job                slot5@lcg0732.gridpp.rl.ac.uk
Machine            Job                slot6@lcg0732.gridpp.rl.ac.uk
Machine            Job                slot7@lcg0732.gridpp.rl.ac.uk
Machine            Job                slot8@lcg0732.gridpp.rl.ac.uk
[root@lcg0732 ~]# su - alahiff
-bash-4.1$ condor_submit condor.sub
Submitting job(s).
1 job(s) submitted to cluster 1.
-bash-4.1$ condor_q

-- Submitter: lcg0732.gridpp.rl.ac.uk : <130.246.216.4:34655> : lcg0732.gridpp.rl.ac.uk
 ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
   1.0   alahiff         3/4  10:25   0+00:00:02 R  0   0.0  sleep 60

1 jobs; 0 completed, 0 removed, 0 idle, 1 running, 0 held, 0 suspended
```

Configuration

- Use default config + /etc/condor/config.d/
  - Files read in alphanumerical order
  - Have each “feature” in a different file
    - Security
    - Fairshares
    - Assignment of accounting groups
    - Resource limits
    - …
- Configuration managed by Quattor
  - Use ncm-filecopy to write config files
  - Runs condor_reconfig as necessary so that changes are picked up
  - Don’t miss YAIM at all – better without
- A number of sites have written Puppet modules
  - Generally available in github

Queues

- HTCondor has no concept of queues in the Torque sense
  - We see no reason to have such queues
- Jobs can request what resources they require, e.g.

```
request_cpus = 8
request_memory = 16000
request_disk = 20000
```

- Jobs can also specify other requirements, e.g.

```
Requirements = OpSysAndVer == “SL5”
```

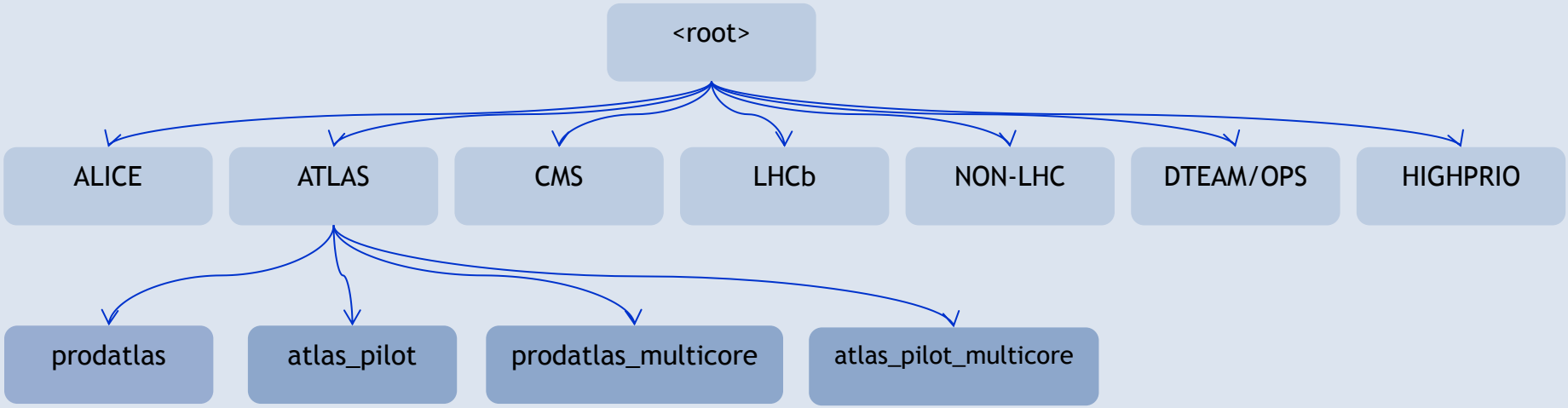
or

```
Requirements = Machine == "lcg1647.gridpp.rl.ac.uk"
```

- Therefore
  - CE needs to pass on job requirements to HTCondor

Fairshares

- Using similar hierarchical fairshares to what we used in Torque/Maui
- Accounting group setup (only ATLAS sub-groups shown)



- Configuration
  - Negotiator configured to consider DTEAM/OPS and HIGHPRIO groups before all others
  - VO CE SUM test jobs forced to be in HIGHPRIO group

Other features

- High availability of central manager
  - Using 2 central managers
  - Shared filesystem not required
  - Default configuration from documentation works fine for us
- Startd cron
  - Worker node health-check script
  - Information about problems advertised in WN ClassAds
  - Prevents new jobs from starting in the event of problems
    - Checks CVMFS, disk, swap, …
    - If problem with ATLAS CVMFS, only stops new ATLAS jobs from starting
- Cgroups
  - Testing both cpu & memory cgroups
  - Help to ensure jobs use only the resources they request

Scalability

- Initial testing
  - Prior to deployment into production
  - 110 8-core worker nodes, high number of slots each
  - Easily got to 16,000 running jobs without tuning
- More recent testing
  - 64 32-core worker nodes, high number of slots each
  - So far have had over 30,000 running jobs successfully

Support

- Support options
  - Free, via mailing list
  - Fee-based, via HTCondor developers or third-party companies
- Our experience
  - So far very good
  - Experienced issue affecting high-availability of central mangers
    - Fixed quickly & released in 8.0.2
  - Experienced issue caused by network break between CEs and WNs
    - Problem quickly understood & fixed in 8.1.4
  - Questions answered quickly
- Other support
  - US Tier-1s have years of experience with HTCondor & close ties to developers
    - They have also been very helpful

Multi-core jobs

- WN configuration
  - Partitionable slots: WN resources (CPU, memory, disk, …) divided up as necessary for jobs
- Partitioning of resources
  - We're using dynamic allocation of multi-core jobs
  - Easily could partition resources
    - Since ATLAS usage has been ~stable, this wouldn't waste resources
- condor_defrag daemon
  - Finds WNs to drain, drains them, then cancels draining when necessary
  - Works immediately out-of-the-box, but we're tuning it to:
    - Minimize wasted CPUs
    - Ensure start-up rate of multi-core jobs is adequate
    - Maintain required number of running multi-core jobs

Dynamic WNs

- HTCondor was designed to make use of opportunistic resources
  - No restarting of any services (like Torque would require)
  - No hard-wired list of WNs
  - No pre-configuration of potential WNs
    - WNs advertise themselves to the collector
    - With appropriate security permissions, can join the pool and run jobs
- Dynamic provisioning of virtual WNs
  - Common to use simple scripts to monitor pools & instantiate VMs as necessary
  - Alternatively, can use existing power management functionality in HTCondor
  - condor_rooster
    - Designed to wake-up powered down physical WNs as needed
    - Can configure to run command to instantiate a VM
  - Easy to configure HTCondor on virtual WNs to drain then shutdown the WN after a certain time
  - Tested successfully at RAL (not yet in production)
    - CHEP 2013 http://indico.cern.ch/event/214784/session/9/contribution/205
    - HEPiX Fall 2013 http://indico.cern.ch/event/247864/session/4/contribution/53

Summary

- Due to scalability problems with Torque + Maui, migrated to HTCondor
- We are happy with the choice we made based on our requirements
  - Confident that the functionality & scalability of HTCondor will meet our needs for the foreseeable future
- We have both ARC & CREAM CEs working with HTCondor
  - Relatively easy to get working
  - Aim to phase-out CREAM CEs

Questions? Comments?