

Two Years of HTCondor at the RAL Tier-1

Andrew Lahiff, Alastair Dewhurst, John Kelly, Ian Collier
STFC Rutherford Appleton Laboratory

HEPiX Spring 2015 Workshop



Science & Technology
Facilities Council

- Introduction
- Migration to HTCondor
- Experience
- Functionality
- Monitoring
- Ongoing work & future plans

- RAL is a Tier-I for all 4 LHC experiments
 - Also support ~12 non-LHC experiments, including non-HEP
- Computing resources
 - 560 worker nodes, over 12K cores
 - Generally have 40-60K jobs submitted per day
- Torque / Maui had been used for many years
 - Many issues
 - Memory leaks, crashes, constant job submission failures from CEs, ...
 - Severity & number of problems increased as size of farm increased
 - Missing functionality
 - e.g. cgroups, hard to integrate with dynamic resources, ...
 - In 2012 decided it was time to start investigating moving to a new batch system
 - HTCondor was selected





Migration to HTCondor

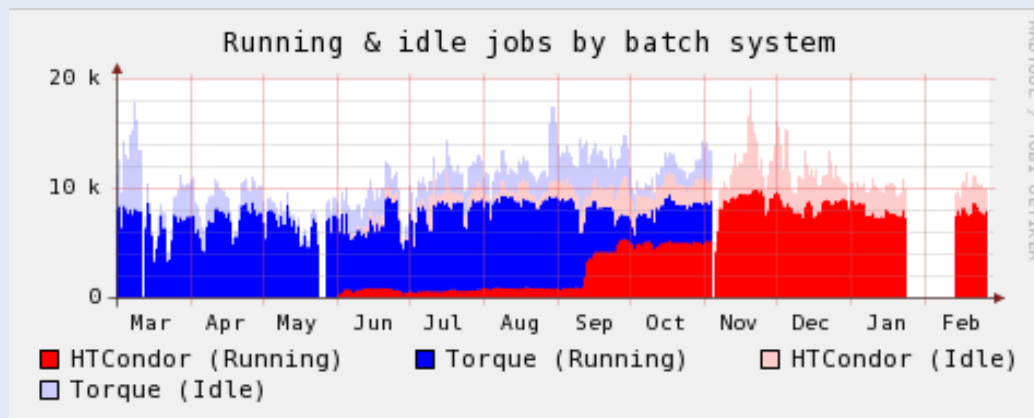
- Setup main components of new batch system
 - (in parallel to the old batch system)
 - Central managers
 - A few worker nodes (old hardware)
 - CEs
- After initial testing
 - Added one generation of older worker nodes (~1000 cores)
 - Capacity beyond LHC MoU commitment
 - Requested ATLAS start submitting to the new CEs
 - (in addition to the existing CEs associated with the old batch system)
 - Fixed any issues that came up
 - Later requested CMS start submitting to the new CEs

- While this testing was ongoing
 - Added monitoring
 - Nagios
 - Ganglia
 - Checked that APEL accounting was accurate & working
 - Wrote internal documentation
 - Service description, installation procedures, ...
 - On-call documentation
- Next steps
 - Testing with ALICE, LHCb, & selected non-LHC VOs
- Once migration to HTCondor approved by management
 - Migrated 50% of CPU to HTCondor
 - Within a few months migrated remaining CPU

- We combined
 - Migration from Torque to HTCondor
 - Migration from SL5 to SL6therefore re-installed workers nodes from scratch
- Alternatives
 - Remove Torque & add HTCondor
 - Add HTCondor, then remove Torque later
 - Can have them running at the same time on the same worker node
 - We initially did some testing with sleep jobs in HTCondor while production jobs were running under Torque on the same worker nodes

- Migration timeline

- 2012 Aug Started evaluating alternatives to Torque / Maui (LSF, Grid Engine, Torque 4, HTCondor, SLURM)
- 2013 Jun Began testing HTCondor with ATLAS & CMS
~1000 cores from old WNs beyond MoU commitments
- 2013 Aug Choice of HTCondor approved by management
- 2013 Sep HTCondor declared production service
Moved 50% of pledged CPU resources to HTCondor
- 2013 Nov Migrated remaining resources to HTCondor

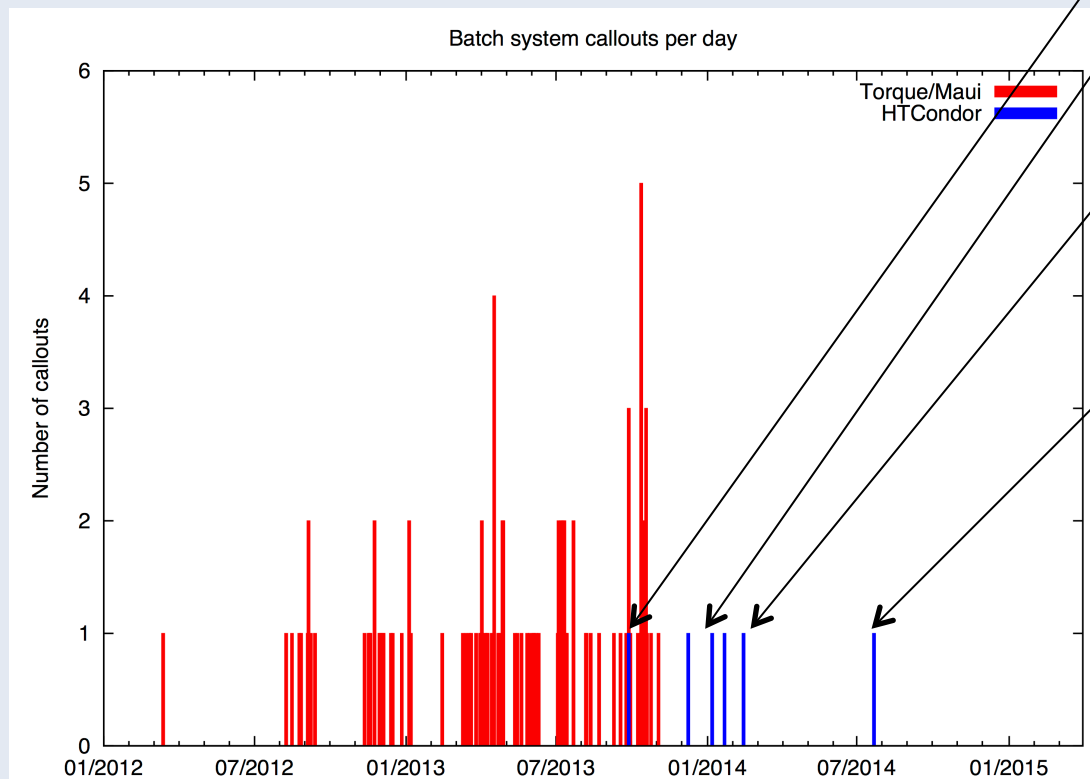




Experience

- Experience over past 2 years with HTCondor
 - Very stable operation
 - Generally just ignore the batch system & everything works fine
 - Staff don't need to spend all their time fire-fighting problems
 - No more days spent studying the Torque source code trying to understand obscure error messages
 - No changes needed as the HTCondor pool increased in size from ~1000 to >10000 cores
 - Job start rate much higher than Torque / Maui even when throttled
 - Farm utilization much better
 - Upgrades easy
 - Central managers/CEs: HTCondor restarts itself after detecting binaries have been updated
 - Worker nodes: configured to drain themselves then restart after binaries are updated
 - Very good support

- Significant reduction in numbers of callouts after migration to HTCondor
 - **None** of the callouts below were actually HTCondor's fault



CVMFS problem

Draining worker nodes

Slow migration of a VM

Network problems



Functionality

- Features in use at RAL

Beginning Hierarchical accounting groups

Partitionable slots

Highly available central managers

PID namespaces

Issue with ATLAS pilots killing themselves

July 2014 MOUNT_UNDER_SCRATCH (+ lcms-plugins-mount-under-scratch)

Jobs have their own /tmp, /var/tmp

July 2014 CPU cgroups

Jobs restricted to the number of cores they request, unless there are free cores available

Feb 2015 Memory cgroups

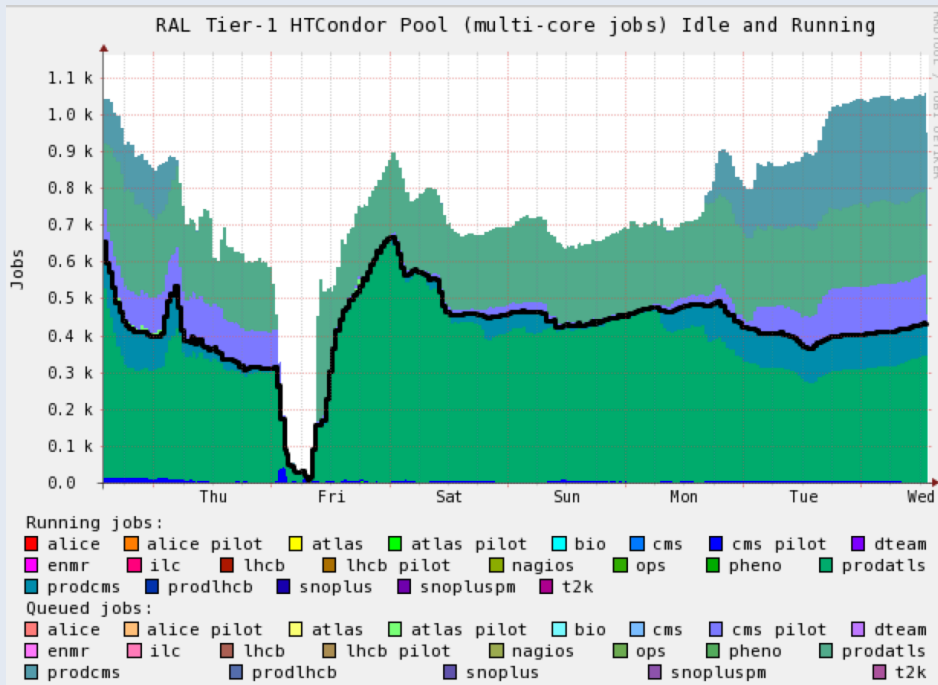
Using soft limits – jobs can exceed the memory they requested if there is memory available on the machine

- Originally
 - By default ARC CE constructs PeriodicRemove expression so that if the ResidentSetSize of the job exceeds the requested memory, the job is killed
- After enabling memory cgroups
 - Thought it would be good to only have the kernel manage memory, so stopped the ARC CE from including a memory limit in PeriodicRemove
- However, found
 - LHCb analysis jobs using > 80 GB RSS (requested 4 GB)
 - ATLAS analysis jobs using ~ 10 GB RSS (requested 3 GB)
- Therefore, re-enabled the “traditional” memory limits, but configured to kill jobs if 3x requested memory is exceeded
 - May reduce this further
- Issue with memory cgroups
 - Under specific circumstances, if one job uses too much memory all cgroups are killed on the worker node (reported to HTCondor developers)

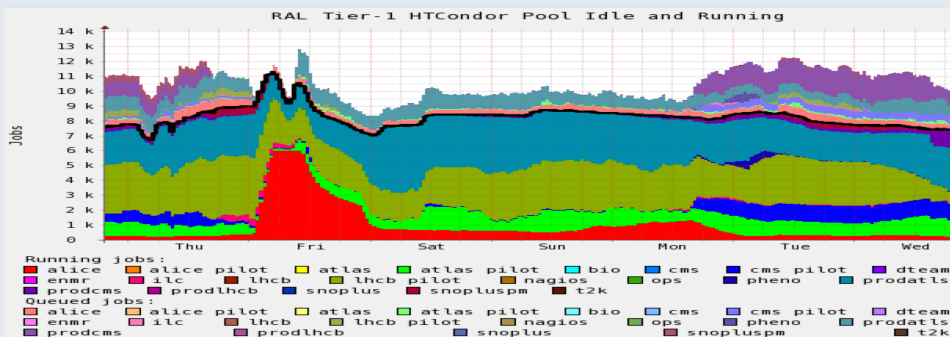
- Current situation
 - ATLAS have been running multi-core jobs since Nov 2013
 - CMS started submitting multi-core jobs in early May 2014
- Did a little tweaking early last year
 - Added accounting groups for multi-core jobs
 - Specified `GROUP_SORT_EXPR` so that multi-core jobs are considered before single-core jobs
 - Defrag daemon enabled, configured so that
 - Drain 8 cores, not whole nodes
 - Pick WNs to drain based on how many cores they have that can be freed
 - Demand for multi-core jobs not known by defrag daemon
 - By default defrag daemon will constantly drain same number of WNs
 - Simple cron to adjust defrag daemon configuration based on demand
 - Uses `condor_config_val` to change `DEFRAG_MAX_CONCURRENT_DRAINING`

- Running & idle multi-core jobs

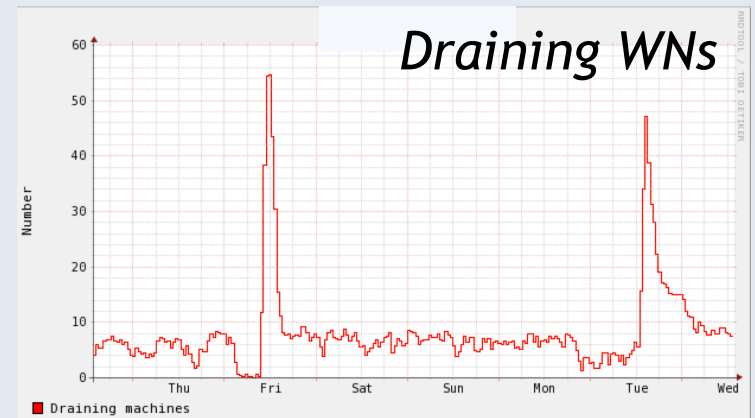
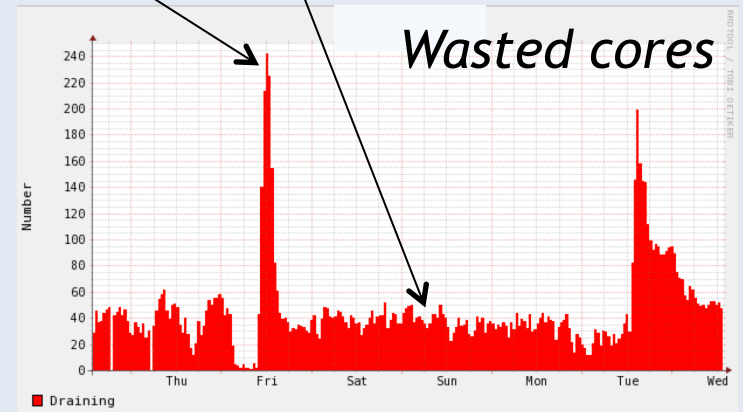
Multi-core jobs



All jobs



2.4% 0.4%



- Using HTCondor to manage virtual machines
 - Makes use of VM universe & some HTCondor features not-commonly used (e.g. job hooks, custom file transfer plugins, condor_chirp)
 - Requires libvirtd to be running on worker nodes
 - Site-instantiated VMs – “vacuum model”
 - Don't provide any way for users/VOs to instantiate VMs
 - See talk by Andrew McNab



Monitoring

- Monitoring used for the RAL HTCondor pool
 - Ganglia
 - Nagios
 - Elasticsearch
 - (HTCondor startd cron)

- Want to ignore worker nodes as much as possible
 - Any problems shouldn't affect new jobs
- Startd cron
 - Script checks for problems on worker nodes
 - Disk full or read-only
 - CVMFS
 - Swap
 - ...
 - Prevents jobs from starting in the event of problems
 - If problem with ATLAS CVMFS, then only prevents ATLAS jobs from starting
 - CVMFS usually “self-heals” eventually
 - Information about problems made available in machine ClassAds
 - Can easily identify WNs with problems, e.g.

```
# condor_status -const 'NODE_STATUS != "All_OK"' -af Machine NODE_STATUS
lcg0980.gridpp.rl.ac.uk Problem: CVMFS for alice.cern.ch
lcg0981.gridpp.rl.ac.uk Problem: CVMFS for cms.cern.ch Problem: CVMFS for lhcb.cern.ch
lcg1675.gridpp.rl.ac.uk Problem: Swap in use, less than 25% free
```

- `condor_gangliad`
 - Runs on a single host (any host)
 - Gathers daemon ClassAds from the collector
 - Publishes metrics to ganglia with host spoofing
 - Uses ganglia library rather than gmetric where possible
 - Examples

Central manager

HTCondor Accounting Groups metrics (92)

HTCondor Defrag metrics (8)

HTCondor Negotiator metrics (26)

HTCondor Pool metrics (9)

HTCondor Submitters metrics (164)

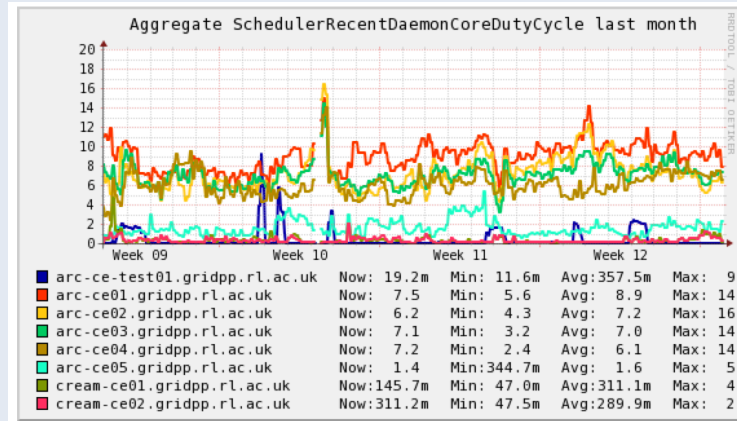
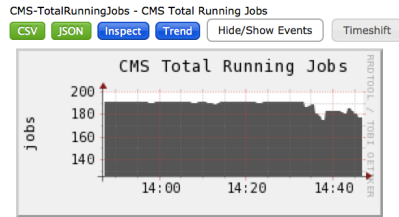
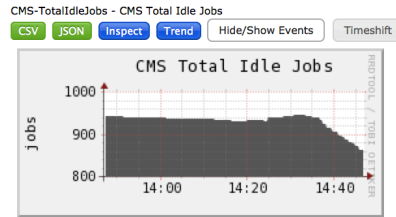
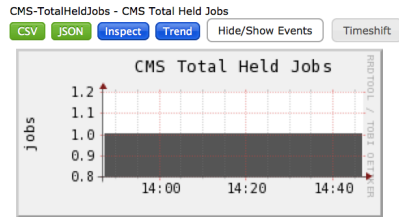
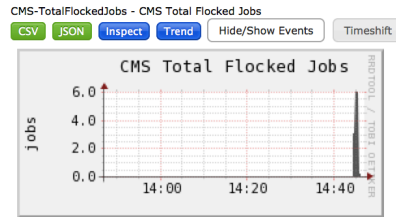
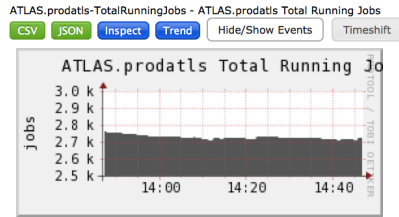
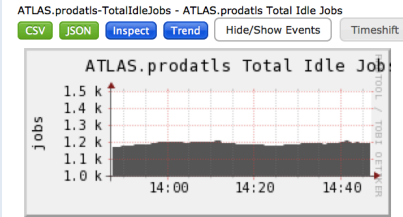
CE

HTCondor File Transfer metrics (12)

HTCondor Schedd metrics (49)

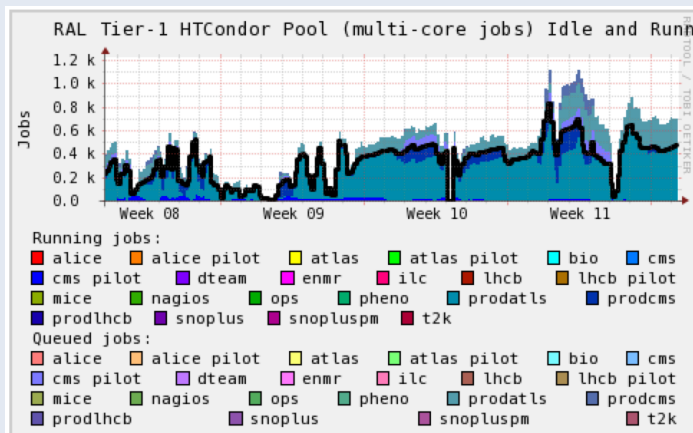
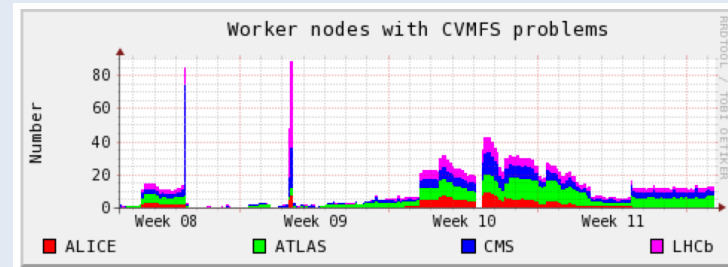
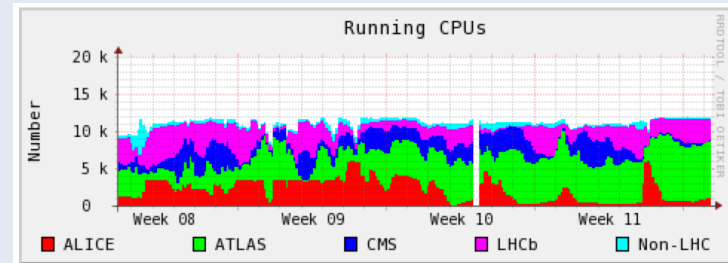
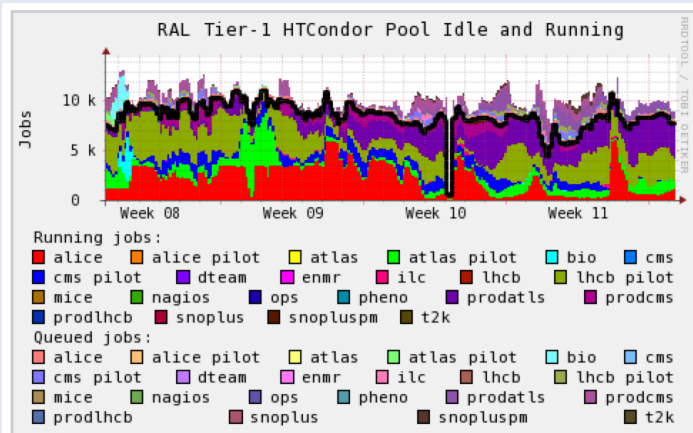
HTCondor Submitters metrics (108)

- condor_gangliad



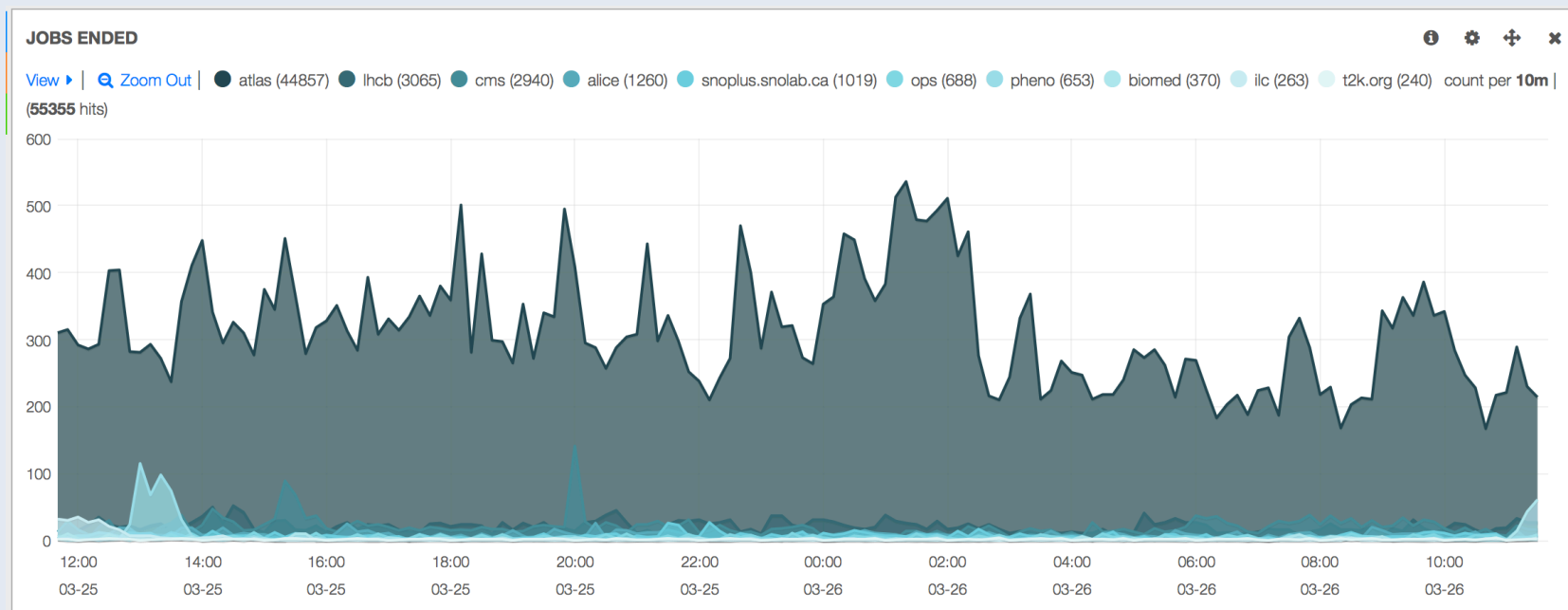
- Custom ganglia plots

- gmetric scripts running on a central manager + Perl scripts on ganglia server
 - If doing this again we would use metrics from condor_gangliad as much as possible rather than making our own



- Central managers
 - Process check for condor_master
 - Check for number of collectors visible in the pool
 - Check for 1 negotiator in the pool
 - Worker node check
 - Need a minimum number of worker nodes advertised & willing to run jobs
- CEs
 - Process check for condor_master
 - Check for schedd being advertised
- Worker nodes
 - Process check for condor_master (won't trigger pager alarm)

- Elasticsearch ELK stack at RAL, mostly used for CASTOR
- Adding HTCondor
 - First step: information about completed jobs
 - Wrote config file for Logstash to enable history files to be parsed
 - Added logstash to all machines running schedds



- ‘Minimal’ resources used
 - Generally < 80,000 documents, < 500 MB per day

Index	# Docs	Primary Size	# Shards	# Replicas	Status
condor-history-2015.03.25	65,533	359.3MB	5	1	open
condor-history-2015.03.24	55,178	299.3MB	5	1	open
condor-history-2015.03.23	24,017	134.3MB	5	1	open
condor-history-2015.02.20	31,325	172.1MB	5	1	open
condor-history-2015.02.19	44,923	245.5MB	5	1	open
condor-history-2015.02.18	42,947	231.5MB	5	1	open
condor-history-2015.02.17	79,410	422.9MB	5	1	open

- Search for information about completed jobs (faster than using condor_history) ARC job id

Condor History
a day ago to a few seconds ago

QUERY ▾

🔍 +

FILTERING ◀

ALL EVENTS

Fields

All (172) / Current (150)

- @timestamp
- @version
- _id
- _index
- _type
- AccountingGroup
- Arguments
- BlockReadKbytes
- BlockReads
- BlockWriteKbytes
- BlockWrites
- BufferBlockSize
- BufferSize
- BytesRecvd
- BytesSent
- ClusterId
- Cmd
- CommittedSlotTime

0 to 1 of 1 available for paging

GlobalJobId	x509UserProxyVOName	RequestCpus	RequestMemory	ResidentSetSize_RAW	LastRemoteHost
arc-ce02.gridpp.rl.ac.uk#2159795.0#1427359616	atlas	8	16000	5467608	slot1@lcg1703.gridpp.rl.ac.

0 to 1 of 1 available for paging



Ongoing work & future plans

- Integration with private cloud
 - OpenNebula cloud setup at RAL, currently with ~1000 cores
 - Want to ensure any idle capacity is used, so why not run virtualized worker nodes?
 - Want opportunistic usage which doesn't interfere with cloud users
 - Batch system expands into cloud when batch system busy & cloud idle
 - Batch system withdraws from cloud when cloud becomes busy
 - Successfully tested, working on moving this into production
 - See talk by George Ryall
- Upgrade worker nodes to SL7
 - Setup SL6 worker node environment in a chroot, run SL6 jobs in the chroot using NAMED_CHROOT functionality in HTCondor
 - Will simplify eventual migration to SL7 – can run both SL6 and SL7 jobs
 - Successfully tested CMS jobs

- Simplification of worker nodes
 - Testing use of CVMFS `grid.cern.ch` for grid middleware
 - 540 packages installed vs 1300 for a normal worker node
 - HTCondor can run jobs:
 - In chroots
 - In filesystem namespaces
 - In PID namespaces
 - In memory cgroups
 - In CPU cgroups
 - Do we really need pool accounts on worker nodes?
 - With the above, one job can't see any processes or files associated with any other jobs on the same worker node, even if the same user
 - Worker nodes and CEs could be much simpler without them!



Questions?



Backup slides

