GridPP

UK Computing for Particle Physics

Arc/Condor

Show how Liverpool runs ATLAS Multicore jobs

Stephen Jones

Introduction

The reference for this is here:

https://www.gridpp.ac.uk/wiki/Example_Build_of_an_ARC/Condor_Cluster

- I'll refer to this “example build” during the talk. The document describes what we did 'warts and all'.
- Much of the good work in it was provided by Andrew Lahiffe, STFC and others.

Introduction

- Around August 2014, sites were asked to prepare for ATLAS Multicore jobs.
- This describes how we fulfilled that request using ARC/Condor. The combination was said to be already capable of running ATLAS Multicore jobs.

Adopting ARC Condor - Topics

ATLAS Multicore

- I'll present the way it works presently at L'pool, alternatives we tried, and the problems we still have. See example build for details.
- Our workernodes are setup to use partitionable slots.
- # cat ./config.d/00-node_parameters

```
RalNodeLabel = E5620
RalScaling =  1.205
NUM_SLOTS = 1
SLOT_TYPE_1              = cpus=10,mem=auto,disk=auto
NUM_SLOTS_TYPE_1         = 1
SLOT_TYPE_1_PARTITIONABLE = TRUE
```

- After node type and normalization factor, it tells us we have 1 slot that is partitionable from 1 .. 10 cpus.

Adopting ARC Condor - Topics

ATLAS Multicore

- This workernode can run jobs that want 1 cpu, or 10 or anything between. We don't use all 16 cpus because that's not optimal due to hyper-thread tail-off.
- We only ever get jobs that want 1 or 8. So the possible usages are
  - 1 x 8 core and up to 2 single core
  - 0 x 8 core and up to 10 single core
- It is the responsibility of HTCondor to schedule the work.

Adopting ARC Condor - Topics

ATLAS Multicore

- But there's a scheduling problem.
- Consider a node with (say) eight slots, running eight single core jobs. One is the first to end; a slot becomes free.
- But say the highest priority queued job needs eight cores.
- The newly freed slot is not wide enough to take it, so it has to wait.
- Should the scheduler use the slot for a waiting single core job, or hold it back for the other seven jobs to end?
- If it holds jobs back, then resources are wasted.
- If it runs another single core job, then the multicore job has no prospect of ever running.
- Multicore jobs “need all lanes clear at the right time”.

Adopting ARC Condor - Topics

ATLAS Multicore

- The solution that Condor provides has two rules: periodically drain down nodes so that a multicore job can fit on them.
- It's also beneficent to start multicore jobs in preference to single core jobs so they get on the newly drained nodes.
- This is implemented using the Condor DEFRAG daemon, and various job priority parameters. The daemon has parameters which control the way nodes are selected and drained for multicore jobs.
- The DEFRAG daemon selects nodes by rate.

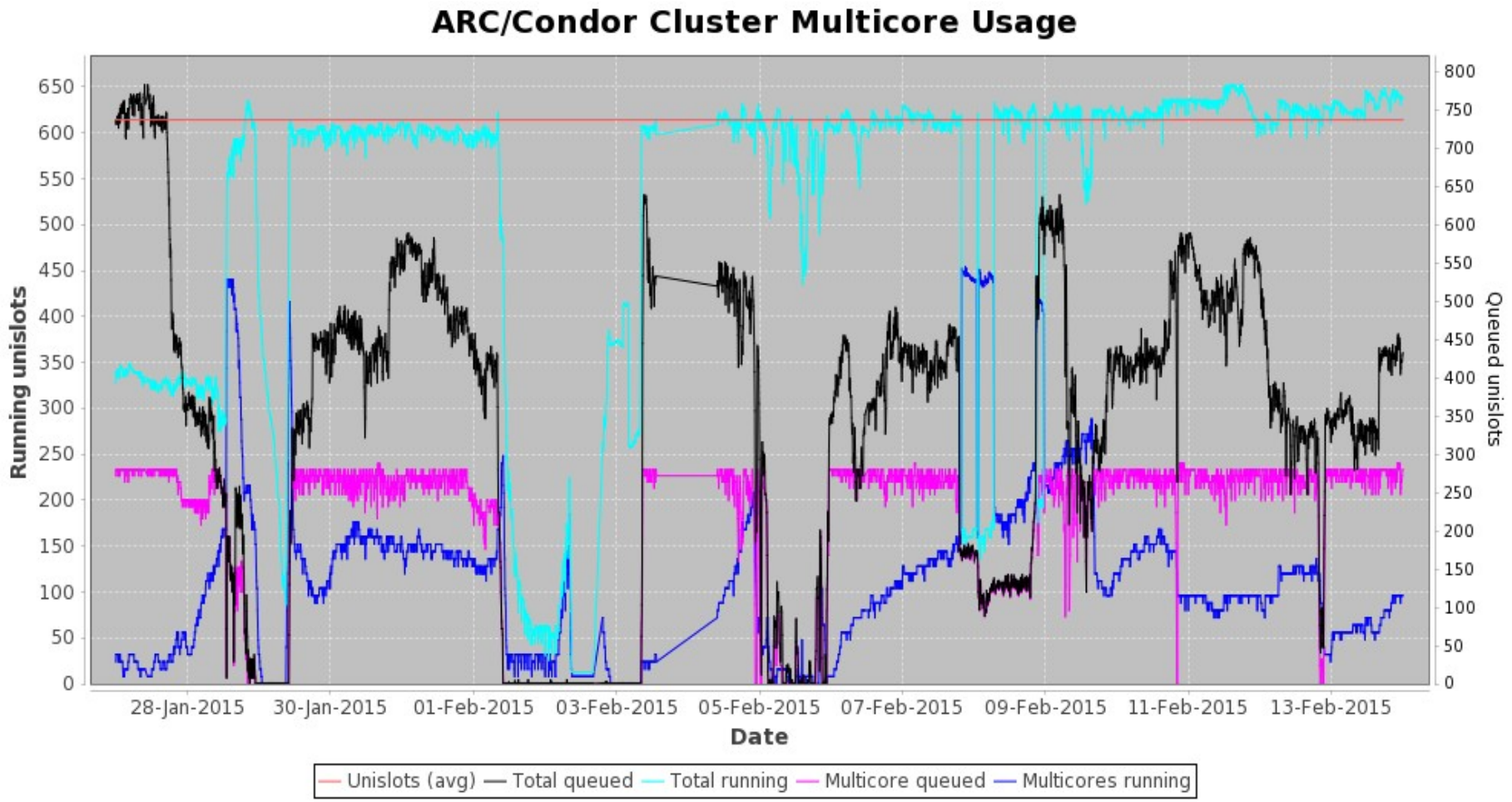
Adopting ARC Condor - Topics

DEFRAG Daemon

- **Our version 8.2.7 good ("vers < 8.2.2 buggy")**
- Main parameters:
  - MAX_CONCURRENT_DRAINING - Don't let more than this drain at once
  - DRAINING_MACHINES_PER_HOUR - Never start more than this many draining per hour
  - MAX_WHOLE_MACHINES - Don't bother draining if this many machines already have wide slot
- State constituting a WHOLE_MACHINE defined in an expression (classad)
- Tailor those constraints to get the drain rate you "want"; can be automated in (e.g.) cron.
- Note: no provision of Setpoint driven control in this version.

Adopting ARC Condor - Topics

DEFRAG Daemon (~ 2.5 weeks)



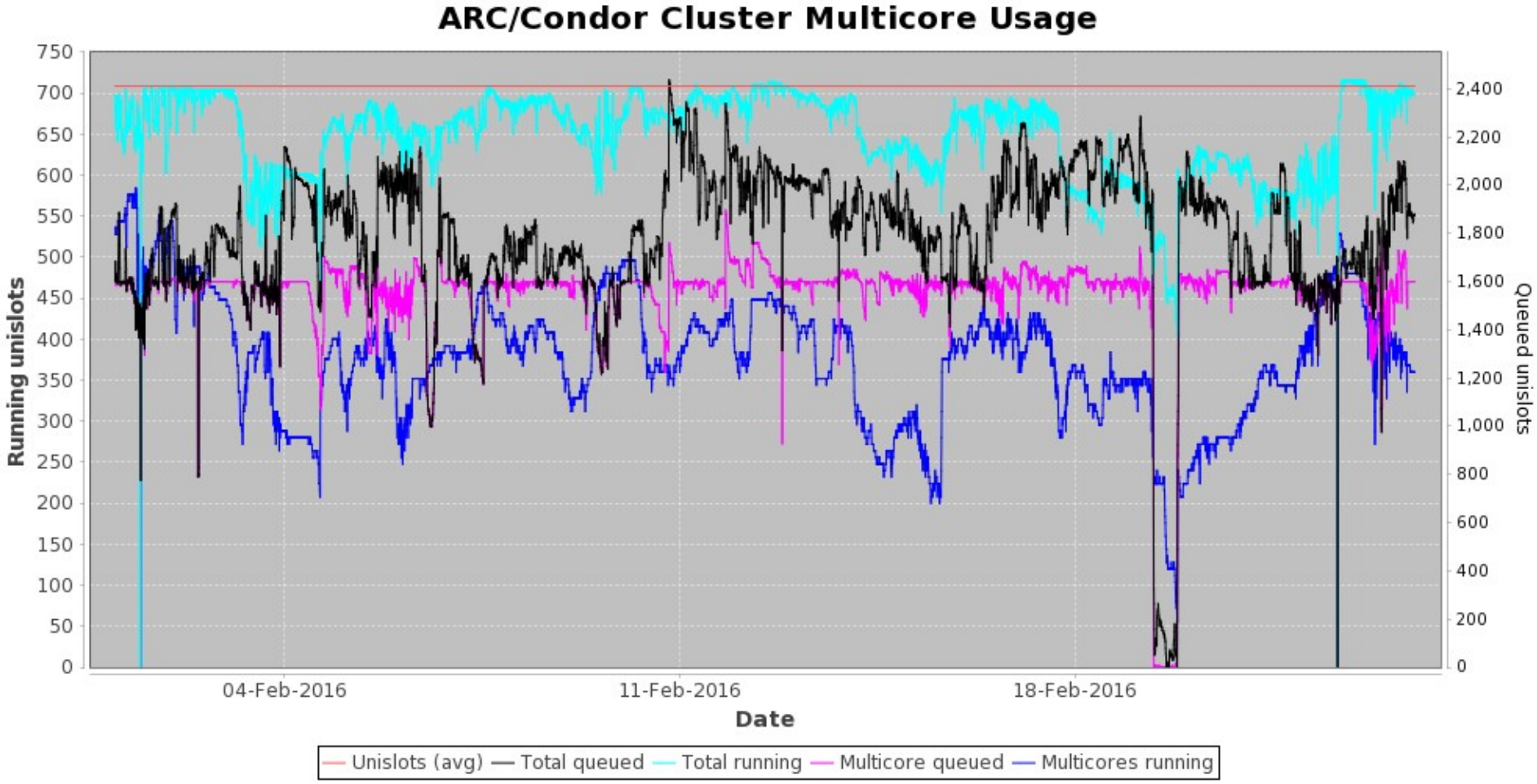
Adopting ARC Condor - Topics

DEFRAG Daemon

- It seems a bit scrappy... but I can't blame the daemon – I was modifying it, trying different rates, different limits, automatic adjustments and the job traffic was stopy/starty.
- But is has no provision to drive the number of MC jobs to a setpoint.
- We wanted to drain the “right” number of nodes to keep a constant number of MC jobs running as long as queues primed, and to “do the right thing” when MC or SC queue became empty.
- To test the ideas, I wrote a script, DrainBoss, which I simplified and now call Fallow, which we run at present.
- It seems to do a fair job, but I'd like to see how the requirements that Fallow fulfils can be integrated into the core HTCondor toolkit, e.g. DEFRAG Daemon so the solution becomes general.

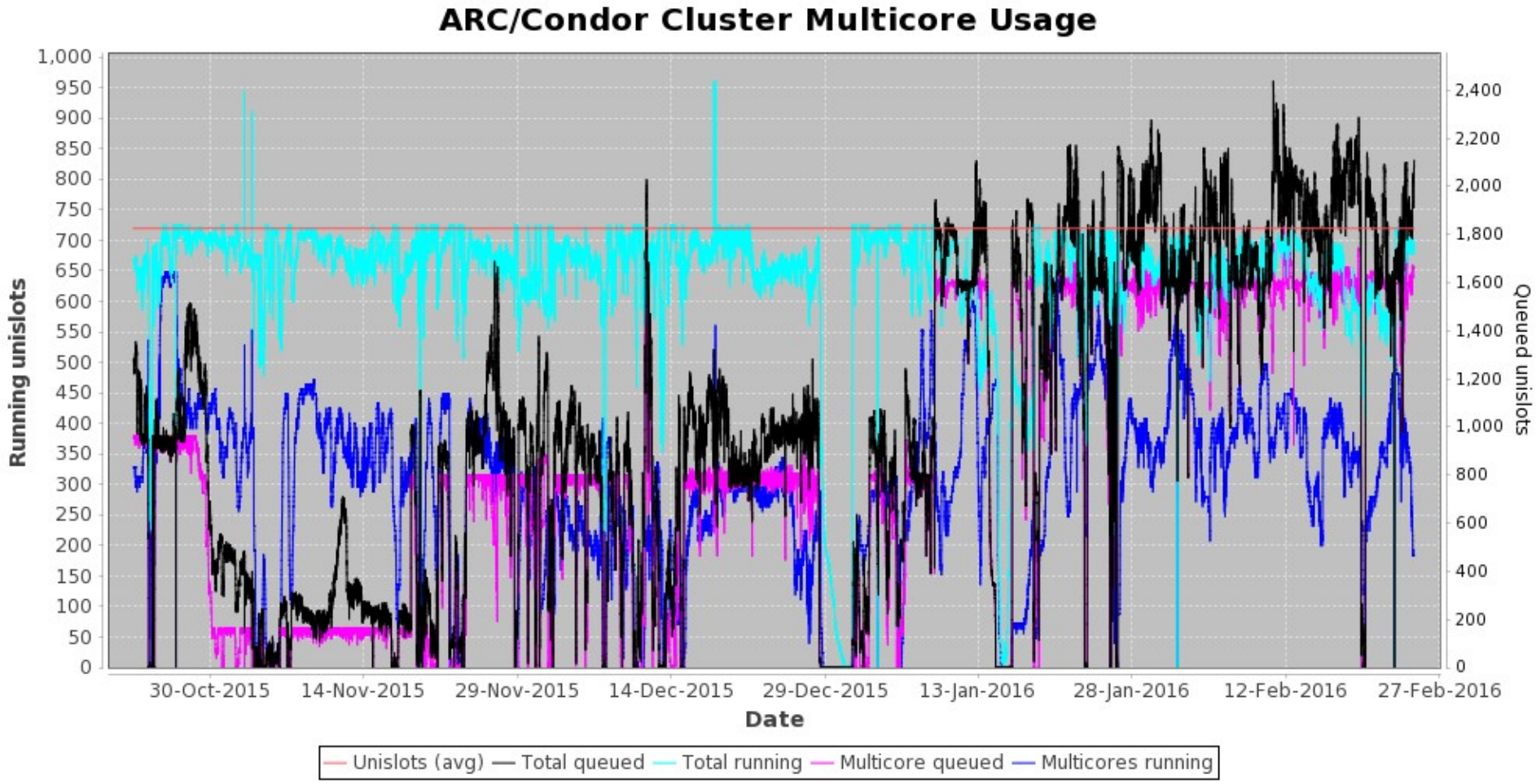
Adopting ARC Condor - Topics

Fallow (3 weeks)



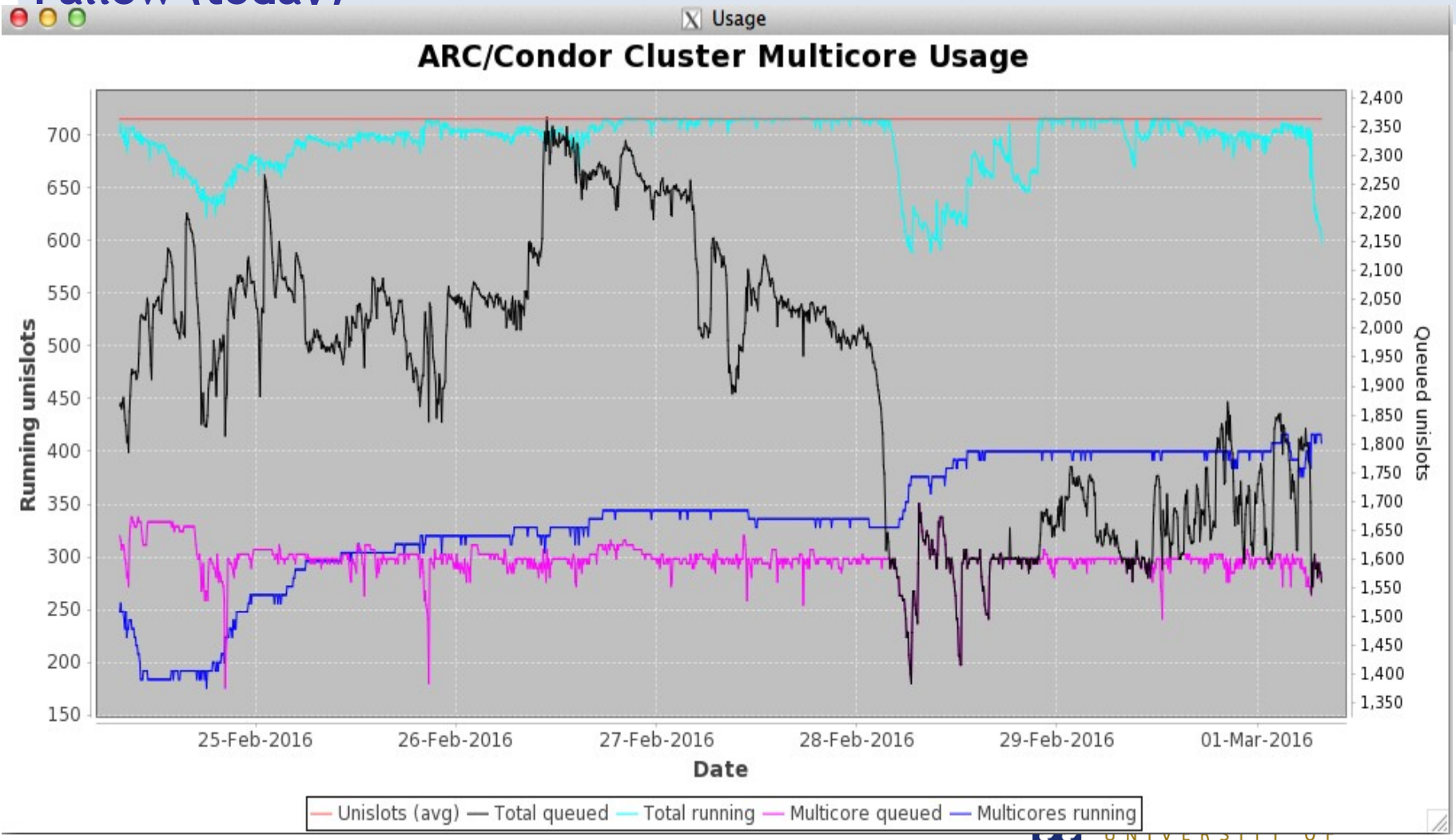
Adopting ARC Condor - Topics

Fallow (4 months)



Adopting ARC Condor - Topics

Fallow (today)



Adopting ARC Condor - Topics

Fallow

- Simplified version of precursor; DrainBoss
- The objectives are to maximise the usage of the cluster and get good mix of both single-core and multicore jobs.
- Senses the condition of the cluster and adjusts how nodes are drained and put back to obtain a certain amount of predictability.
- Has a simple state logic to try to minimise negative corrections and deal with irregular delivery of multicore and single core jobs.
- Implemented as a script, not a daemon.
- Works in conjunction with a mechanism to start multicore jobs in preference to single core jobs, to be discussed later.

Adopting ARC Condor - Topics

Fallow Principles of operation

- Fallow takes a parameter that tells it how many unislots (single cores) should be used ideally by multi-core jobs. This is called the setpoint.
- Fallow detects whether multi-core and single-core jobs are running and queued, and uses the OnlyMulticore attribute (see below) to control whether nodes are allowed or not to run single-core jobs. A node that is not allowed to run single-core jobs is, effectively, draining.
- It does nothing if there are no jobs or if there are only multi-core jobs. This is OK because the cluster is already effectively draining if there are no single-core jobs in the queue, and it's pointless doing anything if there are no jobs at all in the queue.

Adopting ARC Condor - Topics

Fallow Principles of operation

- If there are only single-cores in the queue, Fallow sets OnlyMulticore on all nodes to False, allowing all nodes to any type of job. This is OK because there are no multi-core jobs waiting, so no reservations are wanted.
- If there are multi-core and single-core jobs in the queue, Fallow uses the following logic.
- Fallow works out how many multi-core (8 core) slots are needed to achieve the “setpoint”, i.e. the desired number of MC jobs.
- Fallow exits if there are already enough running (Fallow never stops a running job to achieve the setpoint.)

Adopting ARC Condor - Topics

Fallow Principles of operation

- Fallow then subtracts the running jobs from the desired to find how many newly drained nodes would be needed to reach the desired state. It then discounts nodes that are already OnlyMulticore but not yet with an 8 core slot; these are in progress. This gives the number of new nodes to set OnlyMulticore. Fallow then tries to find a set of nodes that are not OnlyMulticore, and sets them OnlyMulticore, starting the drain. Following this algorithm, the system should eventually converge on (or over) the correct number of multi-core jobs desired.
- Newly drained nodes must be put back online. Fallow scans all the nodes, finding ones that are OnlyMulticore but which have already got an 8 core slot. It turns OnlyMulticore off for those nodes, putting them back into service.

Adopting ARC Condor - Topics

Fallow Principles of operation

- Fallow also takes a parameter to delay allowing single core jobs on a node even once it has 8 slots available. This is experimental, but the delay gives condor more time to put an 8 core job on the node.
- Hopefully (I have no statistics on this) this decreases the risk that a newly drained node will be taken by one or more SC jobs, spoiling the drain.

Adopting ARC Condor - Topics

Prepare for Fallow

- OnlyMulticore is a persistent, settable attribute that can be altered by the script. The START classad is consulted before a job is started, and will only yield True for a specific job if OnlyMulticore is False, or OnlyMulticore is True and the job needs 8 cpus. Thus the node can be barred from running single-core jobs by making OnlyMulticore true.
- Lines in the /etc/condor/condor_config.local file need to be amended to hold the OnlyMulticore attribute, as show here.
  - ENABLE_PERSISTENT_CONFIG = TRUE
  - PERSISTENT_CONFIG_DIR = /etc/condor/ral
  - STARTD_ATTRS = $(STARTD_ATTRS) StartJobs, RalNodeOnline, OnlyMulticore
  - STARTD.SETTABLE_ATTRS_ADMINISTRATOR = StartJobs , OnlyMulticore
  - OnlyMulticore = False

Adopting ARC Condor - Topics

Prepare for Fallow

- And the START classad, in the same file, has to be modified to use the OnlyMulticore attribute, as follows.
  - START = ((StartJobs =?= True) && (RalNodeOnline =?= True) && (ifThenElse(OnlyMulticore =?= True,ifThenElse(RequestCpus =?= 8, True, False) ,True)))

Adopting ARC Condor - Topics

Install Fallow

- Yum repo (sysadmin.hep.ac.uk.repo)
  - [sysadmin.hep.ac.uk]
    name=sysadmin.hep.ac.uk
    baseurl=http://www.sysadmin.hep.ac.uk/rpms/ ...
      fabric- management/RPMS.vomstools/
    enabled=1
- yum install fallow
- Edit the ~/scripts/runFallow.sh to set the setpoint
- service fallow start
- Write messages to /root/scripts/fallow.log

Adopting ARC Condor - Topics

Fallow - Preferring Multicore Jobs

- That's the logic we implemented outside HTCondor in a script (Fallow) to drain the “right” number of nodes to maintain an MC setpoint, while considering the condition of the cluster. But draining nodes is futile if newly prepared nodes are grabbed by SC jobs. Fallow tries to reduce the chance of such leaks.
- As a node drains, once 8 slot comes free, system puts 8 core jobs in them. The next run of Fallow will reallow single core, but it is (hopefully) too late. More details in the example build.
- Fallow has the -n option to increase the overlap period (delay the start of single core) where a node has enough slots to run a multicore. This is experimental.
- A Condor parameter that could influence this is GROUP_SORT_EXPR, but we use the following method instead.

Adopting ARC Condor - Topics

Fallow - Preferring Multicore Jobs

- Chance of leak further reduced using user priorities and accounting groups (not APEL.)
- In the example build (14accounting-groups-map.config), any job is assigned to a user in an accounting group with reference to his credential; as per these set of rules (sorry about ifs!)

```
# Basic group (one line, many VOs omitted)
LivAcctGroup = strcat("group_",toUpper(
  ifThenElse(regexp("sgmops11",Owner),"highprio",
  ifThenElse(regexp("^alice", x509UserProxyVOName), "alice",
  ifThenElse(regexp("^atlas", x509UserProxyVOName), "atlas",
  ifThenElse(regexp("^biomed", x509UserProxyVOName), "biomed",
  ifThenElse(regexp("^zeus", x509UserProxyVOName),
"zeus","nonefound")))))))
```

Adopting ARC Condor - Topics

Fallow - Preferring Multicore Jobs

- Subgroup = owner name + Multi-core or Single-core tag (underlined)
  - # Also show whether multi or single core.
  - LivAcctSubGroup = strcat(regexps("([A-Za-z0-9]+[A-Za-z])\d+", Owner, "\1"),
  - ifThenElse(RequestCpus > 1,"_mcore","_score"))
- Assemble whole group
  - AccountingGroup = strcat(LivAcctGroup, ".", LivAcctSubGroup, ".", Owner)
- Add to the job via submit expression
  - SUBMIT_EXPRS = $(SUBMIT_EXPRS) LivAcctGroup, LivAcctSubGroup, AccountingGroup

Adopting ARC Condor - Topics

Fallow - Preferring Multicore Jobs

- Sub accounting group always _mcore or _score. Running condor_userprio (for e.g. ATLAS), priority factor is last col

  group_ATLAS 0.65 Regroup 1000.00

  pilatl_score.pilatl08@ph.liv.ac.uk 500.00 1000.00

  atlas_score.atlas006@ph.liv.ac.uk 500.33 1000.00

  prdatl_mcore.prdatl28@ph.liv.ac.uk 49993.42 1.00

  pilatl_score.pilatl24@ph.liv.ac.uk 96069.21 1000.00

- Priority factor for the _more subgroup has been set to 1

  condor_userprio -setfactor prdatl_mcore.prdatl28@ph.liv.ac.uk 1

- Default priority factor is 1000; so this makes mcore jobs much more likely to be selected to run than score jobs.

Adopting ARC Condor - Topics

Fallow - Preferring Multicore Jobs

- Some leaks will always occur if (say) MC job stream dried up.
- But some avoidable leaks still occur.
- Hypothesis: Manipulation of the User Priority more or less guarantees that multicore jobs will be preferred when ATLAS VO is being served, and since ATLAS VO has by far the biggest fairshare, multicore get preference in the large majority of cases. But the userprio scheme has no bearing on other VOs, notably LHCb. Perhaps single core LHCb jobs are being scheduled because LHCb is starved? This seems to occur even with a 30 minute multicore only window. I'd have to look at scheduling algorithm to be sure.
- Or perhaps I'm missing something very obvious but I don't know where to look. Perhaps I can get some tips at this conference on how to prefer MC jobs. Latest news: Todd has given a link to his tips webpage. See his talk.

Adopting ARC Condor - Topics

ATLAS Multicore Conclusions

- We're reasonably happy with the performance of Fallow for ATLAS Multicore but we'd like to stop the leaks completely.
- Would like to look again at the DEFRAG daemon and maybe compare performance analytically.
- Would want to integrate work/knowledge acquired into HTCondor Core tools (e.g. DRAG daemon), esp. set point driven draining.
- Could make controller more sophisticated with respect to lags.
- Example build contains more information on our experiences with Fallow, DrainBoss and DEFRAG Daemon.

Adopting ARC Condor - Topics

ATLAS Multicore Conclusions

- Two important goals:
  - Build the right system
  - Build the system right
- At present, for use, the DEFRAG daemon is built right.
- Fallow is not built right, but for us it is the right system.
- Suggest add Setpoint-like feature to drive draining to a value then quit, or show how current daemon can be used to achieve the same goal.

Observations

Future Plans

- Get rid of that giant ifthenelse in the group assignements.
- To do that, we need full set of string manipulation functions, e.g. Python-like, Regex like or Perl/C like. etc.Figure out if we can use the DEGRAG daemon to capture the logic of Fallow (i.e. use HTCondor 'principles').
- Tighten up multicore, eliminate the last of the leaks.
- Do tests on Fallow, DrainBoss and DEFRAG daemon. Convey reqs (e.g. setpoint, job/slot stickiness/affinity) to HTCondor dev team.

Observations

- Further resources:
  - http://www.slideserve.com/hanzila/multi-core-jobs-at-the-ral-tier-1

Extra slide

Maximise throughput but respecting memory constraints

For each type of node, run an instance of the benchmark for every number between cores and cores*2, to cover the whole area, from ignoring hyper-threads to using all hyper-threads. Compare all the results and select the number of benchmark instances that gave the maximum applied computing power overall from these scenarios and use that as the number of slots (i.e. logical cpus) for this node type, i.e. chose the sweet spot. Where the sweet spot is flat (e.g. the same overall throughput is obtained with 12, 13 or 14 instances), choose the lowest, because this combines the highest throughput with the most memory available per job. And, in any-case, always chose a number that at least provides adequate memory per job.

This approach maximises use of cpu power in a fully loaded cluster while giving adequate memory for each job.

That's all folks!

The End

But please let me know if anything is left out that should be covered, if there's anything obviously wrong, if there's anything that could be done better or anything else. Thanks.

sjones@hep.ph.liv.ac.uk