GridPP

UK Computing for Particle Physics

Arc/Condor

How Liverpool adopted
ARC / HTCondor Combo
to build a Grid Cluster

Stephen Jones

Overview of talk

- Introduction
- Pressures to change
  - Why we felt the need to change our setup
  - The solution we decided upon
- Adopting ARC/HTCondor
  - How we took up the solution
- Operational observations
  - Feedback about how it goes now

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 so it may be best to open the link. The document describes what we did 'warts and all'.
- If you just use the doc, it might work, but this talk covers how and why it should work.
- Much of the good work in it was provided by Andrew Lahiff, Simon Fay and many others.
- The bad parts (mistakes, conflicts, misunderstandings and unnecessary steps etc.) are my fault.
- Please let me know if you spot anything: sjones@hep.ph.liv.ac.uk

Introduction

- Follows on from Andrews work on integration by showing a concrete implementation of ARC/Condor.
- Since I split the talk into two parts, I'll skip the Multicore section.
- The versions of the software may have moved on by now.
- Any patches I mention may have been applied in the source.
- This talk explains the example build (which specifies almost all the config, but does not explain it.)
- Since YAIM has been (almost) dropped, some of the steps are manual; thus the process seems more onerous than it really is.

Pressures to change

- For years, we ran a ~1,000 slot SL HTC grid cluster that used the Torque batch system. It had been fitted with a CREAM front end. It worked OK, but...
- We had doubts about continuing with Torque.
  - The version used in our EMI release was miles behind.
  - maui (a scheduler) had lockups and delays, unsupported.
  - We spotted inconsistent behavior.
  - Workernode daemons (pbs_mom) crashed often after issuing commands.
  - We lost jobs.
- Around August 2014, sites were asked to prepare for ATLAS Multicore jobs.

Pressures to change

- We reviewed the software available and noticed HTCondor. It's advocates were saying nice things about it.
- We were pointed at a compatible CE from Nordugrid - Advanced Resource Connector, ARC. People were saying nice things about that as well. ARC is (or was) claimed to be the primary ATLAS grid tool for Northern Europe.
- In particular, the combination was said to be already capable of running ATLAS Multicore jobs.
- We decided to try out the ARC/HTCondor combo on a number of nodes to evaluate it, gain experience and write it up.
- The results are this talk and the example build it relates to.

Adopting ARC Condor - Literature

Some literature:

- The ARC System Admin Manual

  http://www.nordugrid.org/documents/arc-ce-sysadm-guide.pdf

- The Condor System Admin Manual

  http://research.cs.wisc.edu/htcondor/manual/v8.2/condor-V8_2_10-Manual.pdf

Adopting ARC Condor - Versions

Version of main middleware:

- ARC 4.1.0-1
  - Status: getting old. Should upgrade. Maybe to 5?
- HTCondor 8.2.7
  - Status: Should upgrade. There's 8.2.10, and well as 8.4 and 8.5 (dev) branches. 8.3 seems to be missing – presumed it was a dev build.

Adopting ARC Condor - HW & SW

- ARC/HTCondor head node is hungry for memory. Ours is a Virtual Machine with 10 gig and 5 cpus which should be fine for 1000's of jobs, I'm told.
- The cluster has types of worker node:
  - E5620, 24GB ram,1.5TB disk, 10 slots (max 16 hyperthreads)
  - X5650, 50GB ram,2TB disk, 16 slots (max 24 hyperthreads)
  - E5-2630, 50GB ram, 2TB disk, 14 slots(max 24 hyperthreads)
  - L5420, 16GB ram, 2TB disk, 8 slots (max 8 hyperthreads)
  - Total slots: 724
- It is based on Scientific Linux 6.4, kernel 2.6.32-573.12.1
- Repositories for the middle-ware software are specified in the example build.

Adopting ARC Condor - Head node

- The head node in our setup runs both ARC and HTCondor.
- Sites have local standards for grid nodes, so I won't go into all that. A baseline build might include the OS as well as networks, firewall, nagios, ganglia, ssh etc.
- Lots of middle-ware had to go on the **head node**; it's all specified in the example build, along with the yum repos and some extra directories that need to be created. But here's a slide holding the raw list of software components that we installed at Liverpool.
- Each of these components pulled in a heap of dependencies, so the use of yum is essential.

Adopting ARC Condor - Head node

- Essential:
  - ARC CE, HTCondor, Apel Accounting
  -
- Possibly needed:
  - CA Certificates, lcas components, lcmaps components,
  - globus-ftp for file transfer, globus-gsi for security,
  -
- Configuration tools:
  - VomsSnooper (set up the LSC (list of Certificates) files)
  - YAIM (used to make accounts)
  - Various helpers to yum

Adopting ARC Condor - Head node

- Since the features we want to discuss work in conjunction with both ARC and HTCondor, we need to present the basics for both systems first.
- The config file on the head node for ARC is /etc/arc.conf. I won't go through every detail, but I'll need to cover noteworthy parameters as we go.
- There are several config files for HTCondor, but from our point of view the main ones are in /etc/condor
  - condor_config.local - local settings
  - ./config.d/11fairshares.config - fair shares
  - ./config.d/14accounting-groups-map.config - assignment to groups
- There is also a set of RTE (environment) files.

Adopting ARC Condor - Head node

Head node User Accounts

- ARC maps the user of a job to a local account using ARGUS/lcmaps, so it's necessary to generate a set of accounts that correspond to users' credentials.
- Sites would have used (e.g.) YAIM to build them.
- Even though YAIM does not cover ARC/HTCondor, we borrow the functions of YAIM to do this job.
- We need users.conf, groups.conf and the site's standard site-info.def and vo.d (tools to gen users/groups.conf)
- We use this YAIM command to generate the users' config:

```
# yaim -r -s /root/glitecfg/site-info.def -n ABC -f config_users
```

- The actual mapping operation is done by ARGUS (below)

Adopting ARC Condor - Head node

Head node LSC, VOMSES, glexec

- ARC uses ARGUS to do authn/authz, so I don't know how much of the following is really necessary.
- To generate LSC files, I install VomsSnooper and use it as so.

```
cd /opt/GridDevel/vomssnooper/usecases/getLSCRecords
sed -i -e \"s/ vomsdir/ \/etc\/grid-security\/vomsdir/g\" ...
    getLSCRecords.sh
./getLSCRecords.sh
```

- To generate VOMSES, we use this YAIM command:

```
yaim -r -s /root/glitecfg/site-info.def -n ABC -f config_vomses
```

- To configure the vomsdir, we use this YAIM command:

```
yaim -r -s /root/glitecfg/site-info.def -n ABC -f config_vomsdir
```

- And to configure glexec:

```
yaim -c -s /root/glitecfg/site-info.def -n GLEXEC_wn
```

Adopting ARC Condor - Headnode

Headnode lcmaps/ARGUS

- We use an central ARGUS server for authn/authz/mapping.
- The CE passes credential to ARGUS server, which returns mapping to a local account.
- The arc.conf in example build contains copious notes on the ARC Interface to lcmaps (which we determined with strace because it is otherwise poorly or wrongly documented).
- The lcmaps.db file has coordinates of local ARGUS server, which must have policies that serve requests from ARC.
- Such policies are practically identical to the CREAM polices (only the resource id changed.)

Adopting ARC Condor - Workernode

- Workernodes are based on the local site WN standard which includes CVMFS.
- Packages (main)
  - HTCondor
  - emi-wn (for glite-brokerinfo at least),lcg-util, lcg-util-libs,lcg-util-python, lfc-devel, lfc, lfc-perl, lfc-python, uberftp,voms-clients3,
  - voms, gfal2-plugin-lfc,HEP_OSlibs_SL6, yaim
- Packages (libs)
  - libXft-devel, libxml2-devel, libXpm-devel
- Packages (for VOs)

  bzip2-devel,compat-gcc-34-c++,compat-gcc-34-g77 gcc-c++,gcc-gfortran,git,gmp-devel, imake, ipmitool, libgfortran, liblockfile-devel,ncurses-devel, python
- Also CVMFS and software mount points, as per any WN.

Adopting ARC Condor - Workernode

- We also needed to add a bunch of directories that the packages didn't automatically create.
  - /etc/arc/runtime/ENV
  - /etc/condor/ral
  - /etc/lcmaps/
  - /root/glitecfg/services
  - /root/scripts
  - /var/spool/arc/debugging
  - /var/spool/arc/grid
  - /var/spool/arc/jobstatus

Adopting ARC Condor - Workernode

General Workernode Config

- The example build contains the config for a workernode as a reference
- The following files are supplied, with further details on some below
  - A perl script to set up the node parameters.
  - A condor_config.local file that shows all the main parameters, including a persistent variable used to control node called StartJobs.
  - A testnode script with a scheme to allow the node to self check.
  - Environment scripts.
  - Crons.
  - Yaim input files (site-info.def, vo.d, users.conf, groups.conf)
- Refer to the example build for details on all this material.
- The next slides will explain some of these topics and explain them in more detail.
- Worker nodes also need a set of user accounts; we use the same scheme as we did on the head nodes with Yaim.

Adopting ARC Condor - Workernode

Workernode Parameters

- Each worker node can hold its slot config in a file (which also holds a scaling factor to normalise run times, covered later).
- A good way to to install a file with the appropriate settings is to use a Puppet template with Hiera. But we use set_node_parameters.pl to sense the node type and create the file (see example build). By way of example, here's the file from r21-n01.ph.liv.ac.uk.

```
# cat  /etc/condor/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
# processors : 16
# numberOfCpus : 2
```

Adopting ARC Condor - Workernode

Workernode Control

- We needed a scheme to make a new node come up Off, and to turn it on and off to drain. In the worker node config, we put a section for a persistent varibale, StartJobs, like this

```
ENABLE_PERSISTENT_CONFIG = TRUE
PERSISTENT_CONFIG_DIR = /etc/condor/ral
STARTD_ATTRS = $(STARTD_ATTRS) StartJobs, ...
STARTD.SETTABLE_ATTRS_ADMINISTRATOR = StartJobs, ...
StartJobs = False
```

  And we set the START variable to depend on StartJobs.

```
START = ((StartJobs == True) && blah blah blah
```

- One can then manipulate the nodes from the head node with the condor_config_val tool. A similar scheme is used be Fallow to defrag the nodes.

Adopting ARC Condor - Workernode

Workernode Tests

- Workernode faults can make it blackhole. Jobs which run on a blackhole node fail. Node immediately takes new job. Whole queue is rapidly consumed by one broken node.
- Condor has a STARTD_CRON feature, which runs some arbitrary script. The output of script is assumed to be ClassAds (i.e. variables) which are made available to the STARTD.
- Run testnode script with STARTD_CRON , to detect blackhole faults.
- The example build describes such a scheme. It makes use of a persistent variable, RalNodeOnline, which is transmitted to STARTD.
- RalNodeOnline used in the START conditional. If not True, no starts.
- The current version of the testnode script checks file systems writable, VO directories present, disk space, rpm versions, disk health, yaim status, rpc_statd, kernel version, cvmfs online, CA certs, home dirs present etc. If we find a blackhole cause, we add a test for it.

Adopting ARC Condor - Topics

Scaling and Publishing

- For accounting, capacity and scaling , L'pool uses this:
  - https://www.gridpp.ac.uk/wiki/Publishing_tutorial
- Allows heterogeneous set of workernodes. CPU Times normalized by scaling factor before sent to APEL accounting.
- Procedure refers to CREAM/Torque set-up; tweaks (prov. by A Lahiff))are needed for ARC/Condor.
- arc.conf specifies an authplugin called 'scaling_factors_plugin.py'. It runs when a job ends, and it alters the run times to normalize the accounting. Factor is setup in advance in the workernodes and passed to the script.
- Benchmark value to which nodes are normalised is set in arc.conf, jobreport_options parameter, and [infosys/glue12] section.

Adopting ARC Condor - Topics

Accounting - Use of Jura

- Jura is installed in the ARC package.
- It's a C++ compiled binary that reads the ARC logs, formats up the accounting data into the required APEL (format), sends it via SMS queue to some accounting gateway, then cleans up the old files. ARC runs it periodically.
- It's configured with the jobreport* parameters in arc.conf.
- It seemed OK. But at month end, we saw discrepancies between “Record Count In Your Database” and “Record Count What You Published”.

  http://goc-accounting.grid-support.ac.uk/rss/ ...

  UKI-NORTHGRID-LIV-HEP_Sync.html
- The differences were smallish (1% to 3%, from memory)

Adopting ARC Condor - Topics

Accounting - Use of Jura

- I never got to the very bottom of the bug.
- But the problem never came back after I
  - Edited the jura src to stop it deleting the sent log files.
  - Cleaning up the log files by hand after.
- I suspect Jura was cleaning up files that had not yet been successfully transferred.
- The patch is unofficial but I can share it should anyone else suffer small glitches when publishing APEL accounting.
- ARC dev should take note of this (will write ticket soon).

Adopting ARC Condor - Topics

Accounting - Resubmission

- It often becomes necessary to republish accounting records after some mishap.
- Jernej Porenta has provided a script to do this.
- The script, and instruction on how to use it, are provided in the example build.
- Note: it's only possible if the archiving option in the arc.conf was set during the period in question.

Adopting ARC Condor - Topics

RTE - Run Time Environments

- Condor uses “run time environments”. We use GLITE rte.
- An authplugin line in arc.conf associates a plugin script with the GLITE RTE. The GLITE RTE, the plugin script (and a debug version) is supplied in the example build.
- When a job is run, it picks up the GLITE RTE.
- The GLITE RTE contains a set of essential shell variables that each job must inherit.
- In particular, each VO's software directory is set up here, which is either an NFS share (sw dir) or a reference to a repo in CVMFS.
- Same mechanism used to transmit SW tags.

Adopting ARC Condor - Topics

Ports

- On Condor headnode and workernodes, we define LOWPORT = X, HIGHPORT = Y; firewalls must allow this traffic. You also need to allow other ports for various services, locked down where pos. to nodes or networks. On our systems:

  Head Node: 22 (ssh), 123 (time), 443 (ssl/arc), 2135 (bdii/arc), 2170 (bdii pos. red.),
  2811 (gsiftp/arc), 5666 (nagios), 8649 (ganglia),
  9001, 9002, 9091 (pos redundant?) , 9618 (condor), X:Y (gsiftp, condor)

  Worker Node: 22 (ssh), 123 (time), 2811 (gsiftp/arc),
  5353 (unk), 5666 (nagios), 5900:5910 (unknown), 8649 (ganglia),
  9618 (condor), X:Y (gsiftp,condor)

Adopting ARC Condor - Topics

Patches

We collected a nice set of patches for ARC as we went through this, all in the example build. Here's the list:

- glue-generator.pl, rem urn:ad (sj)
- glue-generator.pl, compute job breakdown in bdii (al)
- glue-generator.pl, extra fields (al)
- glue-generator.pl, correct cores parsing (sj)
- /usr/libexec/arc/smssend, use_ssl parameter (sj)
- jura, potential race condition in cleanup (sj) (tbd)

(thanks again to A Lahiff for his share)

Adopting ARC Condor - Topics

GOCDB and Registration

Add new service entries for the head node in GOCDB for the following service types.

- gLite-APEL
- gLExec
- ARC-CE

- It is safe to monitor all these services, once they are marked in production.
- Also contact representatives of experiments and tell them about the new CE.
- Important: Ask Atlas to add the new CE in its analysis, production and multicore job queues.

Adopting ARC Condor - Topics

ATLAS Multicore

- That this is WIP. 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
# processors : 16
# numberOfCpus : 2
```

- 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 (see extra slide on this).
- 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, and 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.
-

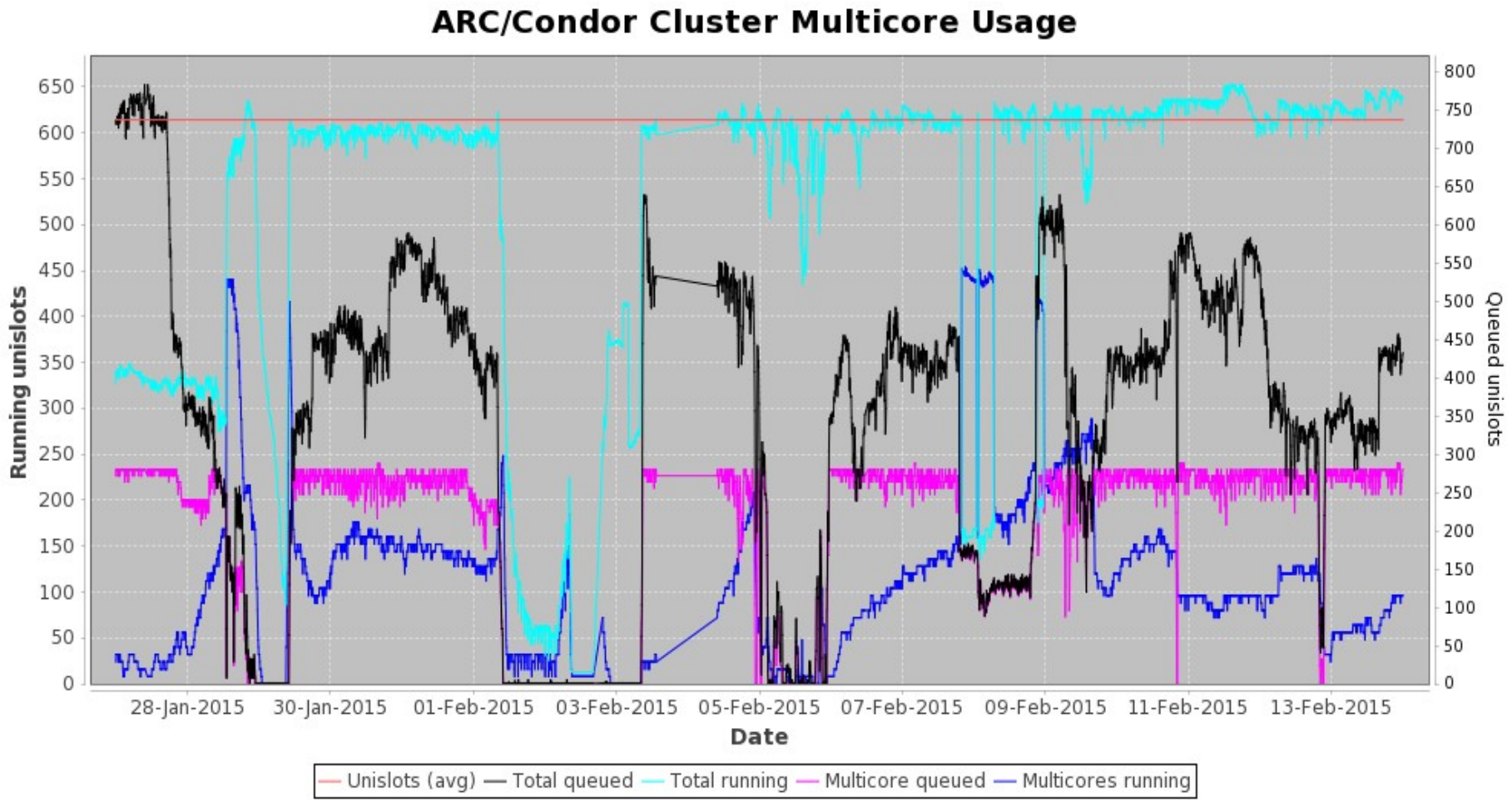
Adopting ARC Condor - Topics

DEFRAG Daemon

- **Our version 8.2.2 good (less 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)



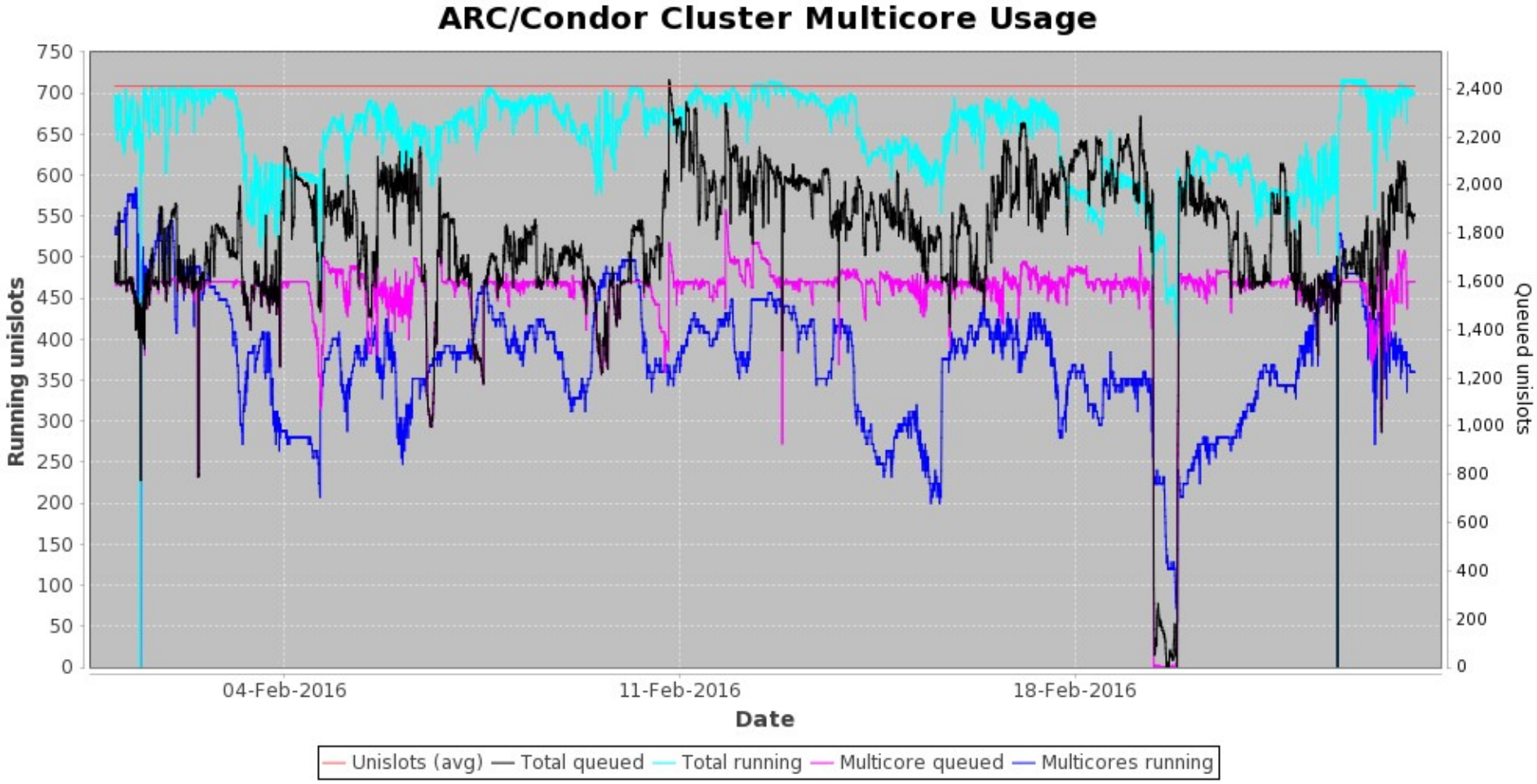
Adopting ARC Condor - Topics

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 provsion 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.

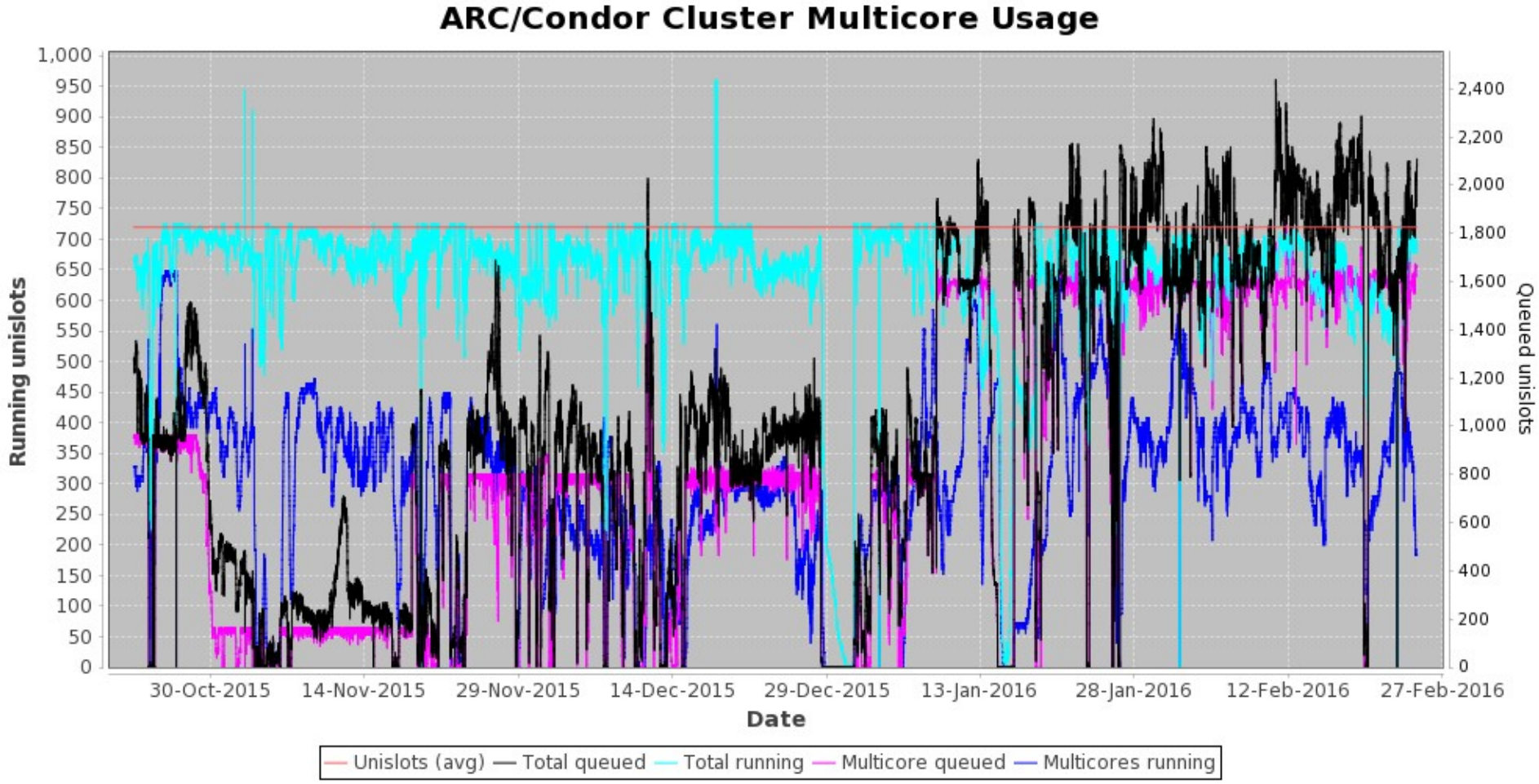
Adopting ARC Condor - Topics

Fallow (3 weeks)



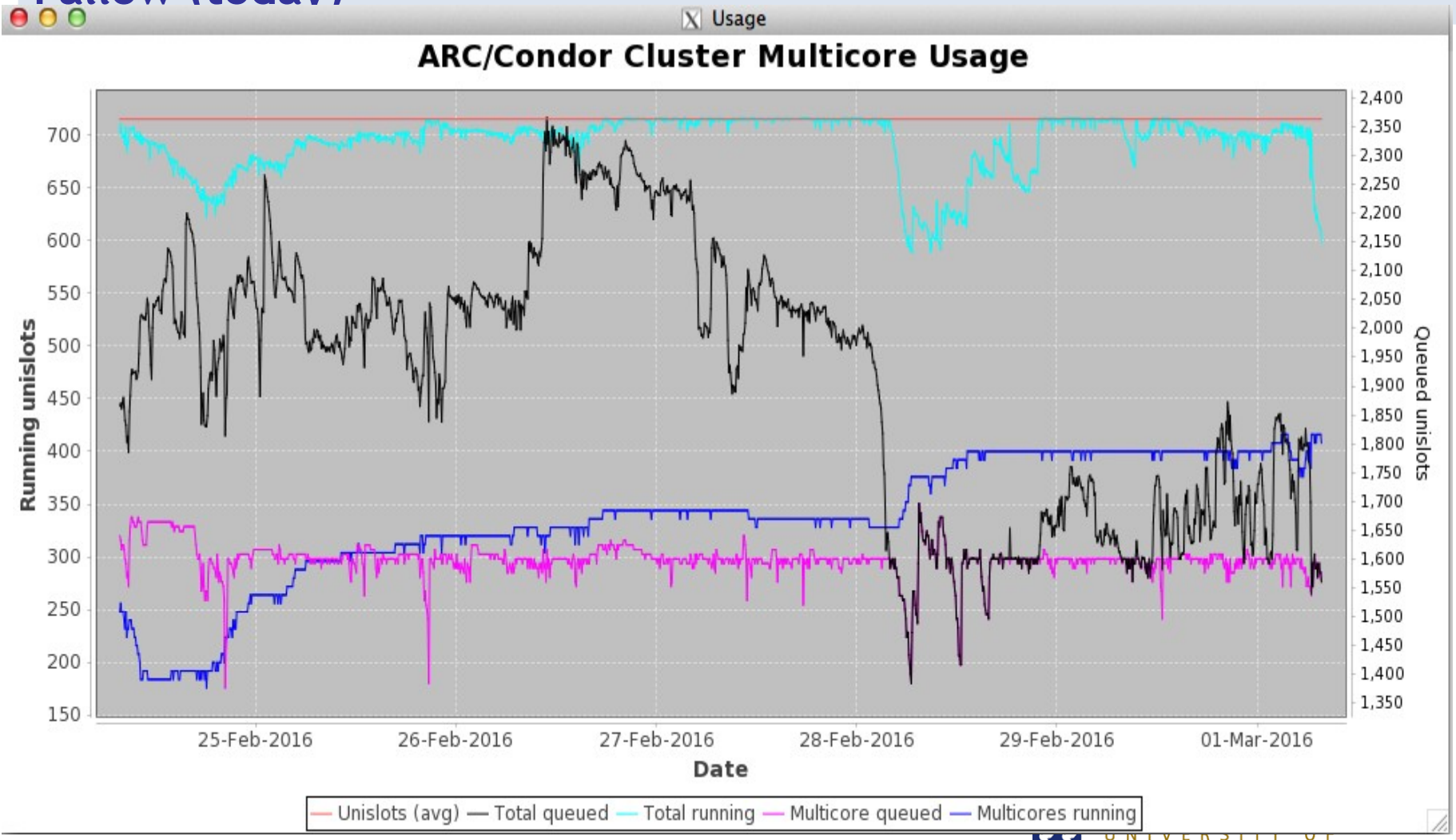
Adopting ARC Condor - Topics

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 how many 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.

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 integerate 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.

Observations

Good things

- HTCondor is a highly configurable, general, capable and very stable batch system.
- ARC is easy to configure and works well with Condor.
- SW and doc maintenance is responsive and professional.
- ARC/Condor combo runs ATLAS Multicore with a low overhead.

Concerns

- Not ARC or HTCondor's fault, but it takes effort to set up as Grid Cluster - lots of small things (hence this talk). Condor config needs a lot of DIY design, i.e. user responsibility for policies, in Condor-speak.
- condor_* commands are quite verbose. A fair amount of use case scripting is required or you'll need to memorize them (see Todd's talk)
- Unlike torque, do not restart the workernode daemons while jobs are running - they'll die.
- BDII and Jura within ARC need maintenance (boring but needs doing).

Observations

- **VOs enabled**
  - About 25.
- **Recent usage**
  - atlas
  - lhcb
  - biomed
  - ilc
  - pheno
  - snoplus
  - mice
  - lsst
  - t2k
  - gridpp (various small VOs encapsulated under that VO)

Observations

Future Plans

- Get rid of that giant ifthenelse in the group assignments
- 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').
- Figure out how to do redundancy, e.g. two headnodes, one a hot spare.
- Tighten up multicore, eliminate the last of the leaks.
- Get cgroups working properly.
- Move it all over to IPv6.
- Upgrade all the versions to make it current.
- Tidy up the install
- Move it to puppet 3 and use Hiera for overides.
- Get those patches in the version, or agree what to do next.

Observations

Future Plans

- Arc:
  - Work to get the BDII patches put in.
  - Fix up Jura.
  - Find and remove redundant steps, esp. wrt RTE, + authn/authz
  - Maybe modularise the install with Puppet (big and difficult job?)
  - Reverse YAIM aspects and _do something_, i.e. write it down, script it or put it in an rpm... whatever.
- CONDOR:
  - Do tests on Fallow, DrainBoss and DEFRAG daemon. Convey reqs (e.g. setpoint, job/slot stickiness/affinity) to HTCondor dev team.
  - Build fallow with library access/API. Not necessary if DEFRAG had (e.g.) setpoint-like functionality.
  - CGROUPS and Memory reqs. (don't yet understand)

Observations

- Some further resources:
  - https://www.gridpp.ac.uk/wiki/ARC_HTCondor_Basic_Install
  - https://www.gridpp.ac.uk/wiki/Imperial_arc_ce_for_cloud
  - https://www.gridpp.ac.uk/wiki/ARC_CE_Tips
  - http://www.slideserve.com/hanzila/multi-core-jobs-at-the-ral-tier-1
  - https://www.gridpp.ac.uk/wiki/ARC_HTCondor_Accounting

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

Extra topics

Snaky, voconfig.py, condor_* wrapper scripts.