

GridPP

UK Computing for Particle Physics

**Squeezing more out of what we have,
trying to do things better and
helping to find new ways to work.**

- We nominally support ~ 20 VOs, with actually traffic from perhaps half that number, the vast majority being from ATLAS/LHCb.
- In rough figures, we have about:
 - 21,445 HS06
 - 2018 slots
 - 1.5 PB DiskLiverpool uses ARC/Condor and VAC. We have never used or tested Covfefe, and we have no intention of ever doing so.
- Our focus has been to provide an efficient service to our customers by maximising performance and quality at low cost; a three way trade-off. You can only have two of those.
- Efficiency: (of a system) - achieving maximum productivity with minimum wasted effort or expense.

- In 2016, we were asked to act as a VAC test site.
- In 2017, we were asked to set-up a version 7 cluster, to test UMD distribution with (in our case) Centos7 workernodes.
- We are currently in the midst of renewing our cooling technology, which is big disruption that needs a lot of babysitting. This has delayed other things, like putting the new storage online, IPv6 (which is always just coming in), VAC multicore etc. We'll get to those things in due course (I always say that.)
- The new storage, ~ 0.3 PB, is undergoing tests using ZFS on Centos7. Will say more later, once we know.

- No scalability issues found. Near full slot occupancy. Jobs from ALICE, ATLAS, LHCb and GridPP.
- Actual usage for ALICE, ATLAS and LHCb at least coarsely relate to the provided workload ratios.
- VAC may not completely eliminate the need for all middle-ware on the node; for example, APEL middle-ware to publish accounting records, file caching software, CVMFS .
- Nonetheless, we found that VAC largely meets its design goals. The total payload traffic from the experiments who participated was found to be consistently high, and the VAC system itself is highly reliable and straight-forward to configure and use. We can recommend VAC to sites that want to run virtual payloads with minimal ongoing maintenance effort.
- CHEP 2016:
<https://indico.cern.ch/event/505613/contributions/2230750/>

- Version 7 cluster is very small at the moment, 24 slots. Not much traffic, I wouldn't mind seeing some more jobs, and I'd put on more slots.
- Haven't tried mcore with it yet. Would like to try.
- Decent documentation is available on the install we used.
 - https://www.gridpp.ac.uk/wiki/Centos7_Adoption
- This documentation also :
 - a) provides tools to allow us to finally abandon Yaim altogether.
 - b) has a simple introduction to Puppet3 and Hieradata (which is about all I know.)
- More on that, if time allows.

- Complete renewal (more efficient, to save money.)
- Divide machine room into two halves with barrier.
- Maintain ops in one half, using ~ half the old cooling, on flat out.
- Tear everything out of the other half, install new cooling there.
- Move everything into the new “half”. Do up the other half if we need the space.
- Obviously, this is hugely disruptive, since we are operating on half power cooling, and the old cooling is flaky anyway.
- Maintaining about 75% to 90% power of cluster. Some flakiness is showing through, that needs careful control to avoid meltdown.
- Nodes scripted to shut themselves off (based on rate of change of CPU temp, and max value) as a last ditch measure to prevent the smoke coming out.

- We did some tuning on the VAC memory.
- Most efficient number of jobs is not necessarily equal to the number of hyper-threads a system can support.
- It is between core and cores * 2 (hyper-threads.) On VAC, this coarsely tallies with the number of VMs.
- VAC payloads are ostensibly variable in size; they actually appear to be consistent.
- On our VAC nodes, memory is the constraint - there is not enough memory to pick running jobs = hyper-threads.
- We have to under-subscribe the cores; cpu not fully utilised.
- So we want to bring VAC to the point where it is most efficient, i.e. as near as possible to the ideal number.
- We over-subscribe VAC in terms of memory, then back off the usage until we see high efficiency.

- On E5-2630 v2, sweet spot is 23 “jobs” (24 max hyperthreads). With VMs at the notional sweet spot wrt cpu, we see this (top -b -n 1 | grep qemu)

-

```
7521 qemu      20  0 4708m 2.6g 4468 S 100.5  5.5 232:55.14 qemu-kvm
15124 qemu     20  0 6610m 2.6g 2884 S 100.5  5.6 381:55.65 qemu-kvm
```

- And so on for 19 jobs or so, then rapid tail off due to use of swap (perhaps?) Can't fit 23 jobs/VMs.

```
16603 qemu     20  0 6562m 2.4g 2892 S 96.7   5.0 647:28.94 qemu-kvm
20147 qemu     20  0 6749m 2.4g 2864 S 94.8   5.0 868:24.76 qemu-kvm
 8161 qemu     20  0 4351m 1.7g 5524 S  0.0   3.6   3:36.50 qemu-kvm
22987 qemu     20  0 4621m 1.7g 5524 S  0.0   3.7   6:54.91 qemu-kvm
```


- I believe the slow payloads are the last to start, and are thus unable to build up their working set and are forced into swap. To check when they started (diff machine) ...

```
# for p in `top -b -n 1 | grep qemu | sed -e "s/qemu.*//"; do ps -eo pid,cmd,etime | grep $p | grep -v -e  
grep -e vhost; done | grep -e 24690 -e 13006 -e 15344
```

```
13006 /usr/libexec/qemu-kvm -name 06:56:11
```

```
15344 /usr/libexec/qemu-kvm -name 54:36
```

```
24690 /usr/libexec/qemu-kvm -name 01:32:26
```

- Yep. Two of the three slow ones started in the last hour or two; while the majority of the fast VMs started many hours before. So, reduce the VM count incrementally until things are right, i.e. opposite of tuning a guitar.

- With 22, all is well (top -b -n 1 | grep qemu)

```
1043 qemu    20  0 7461m 1.9g 848 S 100.5 4.1 32:49.05 qemu-kvm
2161 qemu    20  0 6907m 2.6g 868 S 100.5 5.4 28:46.85 qemu-kvm
3452 qemu    20  0 6574m 2.5g 864 S 100.5 5.4 34:24.79 qemu-kvm
4492 qemu    20  0 7316m 1.9g 888 S 100.5 4.0 32:46.86 qemu-kvm
```

- Etc...

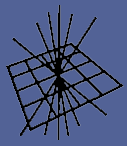
```
25638 qemu   20  0 6823m 2.2g 852 S 98.6 4.6 590:04.97 qemu-kvm
26392 qemu   20  0 7485m 2.6g 864 S 98.6 5.5 38:46.96 qemu-kvm
28161 qemu   20  0 6033m 1.0g 852 S 98.6 2.2 43:09.12 qemu-kvm
30022 qemu   20  0 6674m 1.5g 976 S 98.6 3.2 39:52.91 qemu-kvm
```

- Now they all get plenty of cpu. They may have huge VM, but it's swapped out or shared etc. and never used, hence the working set fits in RAM. No slowness. I can live with that.

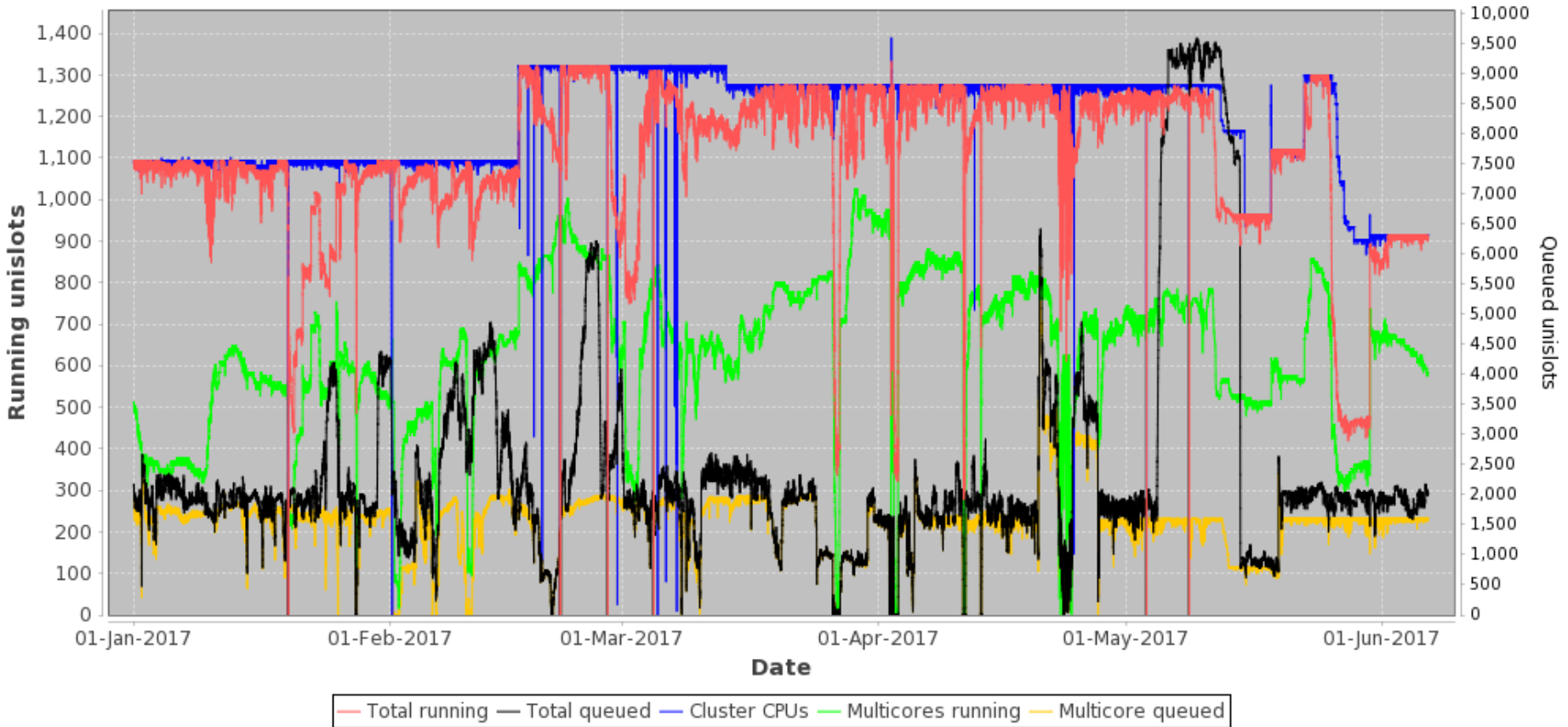
- Our approach to multicore on ARC/Condor_Cluster was original pinched from RAL, and it is extensively documented on the GridPP wiki.
- I've recently optimised this strategy a bit, having learned that other sites were getting better results.
- I'll use the next few slides to describe the method, and the new changes and I'll show some before and after cases.
- Thanks to also RALPP for making the work more general and for the ranking expression (TBD) that further improves the efficiency of the approach.
-

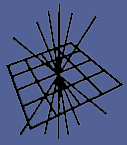
- Small print (for the record):
 - The provision of multicore slots depends on the Condor Partitionable Slots feature.
 - “Unislots” can “coalesce” to form wider slots, to run multicore jobs.
 - They coalesce “on their own”, but only once jobs are drained off.
 - Since jobs don't necessarily all finish at once, single core jobs have to be kept off the slots until they have coalesced into a wide slot and taken a wide job.
 - Hence, at least in the first place, some draining might be needed.
 - This leads to inefficiency (unused cores) while the wide slot is being prepared.
 - Another problem is what happens when a wide job ends? If a single core gets on, no new wide core can start...
- So that defines the problem. a) Find a way to hold single core jobs off nodes to get some wide slots in the first place, then b) keep them after a wide job ends by making sure you run another wide job in the freed up wide slot.

- In reverse order, we solve problem (b) (keeping a wide slot) at Liverpool in 2 ways.
 - First, once a wide slot is drained, we try to delay putting it back in use until it has had chance to run a wide job.
 - Then we assign wide jobs to a Condor accounting subgroup that has a priority factor set to a low number (low number = high priority...)
- And we solve problem (a) (making a wide slot in the first place) using a tool we developed called Fallow.
- We think Fallow has a better algorithm than the defragmentation tool supplied with HTCondor (but external scripting is not really a HTCondor way of doing things.)
- So perhaps we'll go back to the DEFrag daemon sometime and try it again.

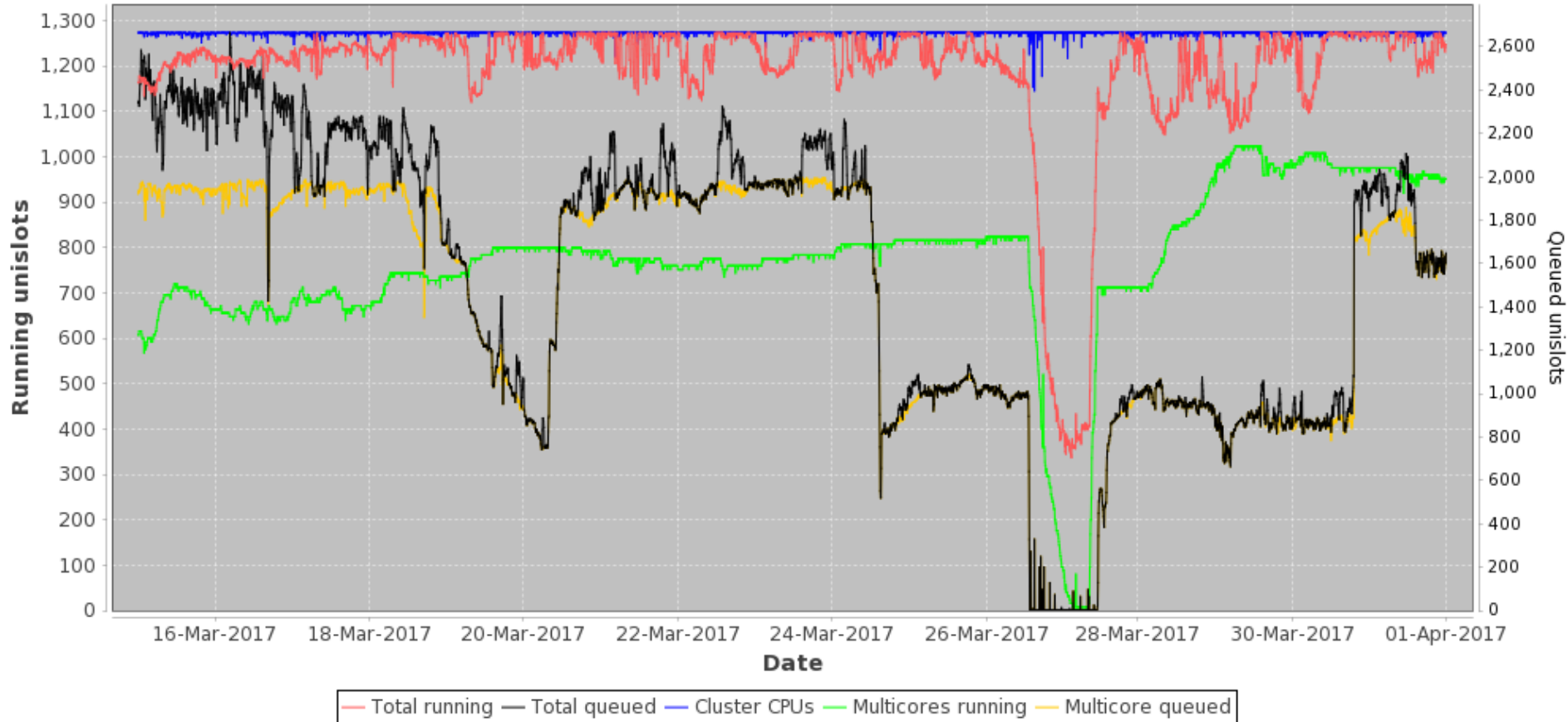


ARC/Condor Cluster Multicore Usage

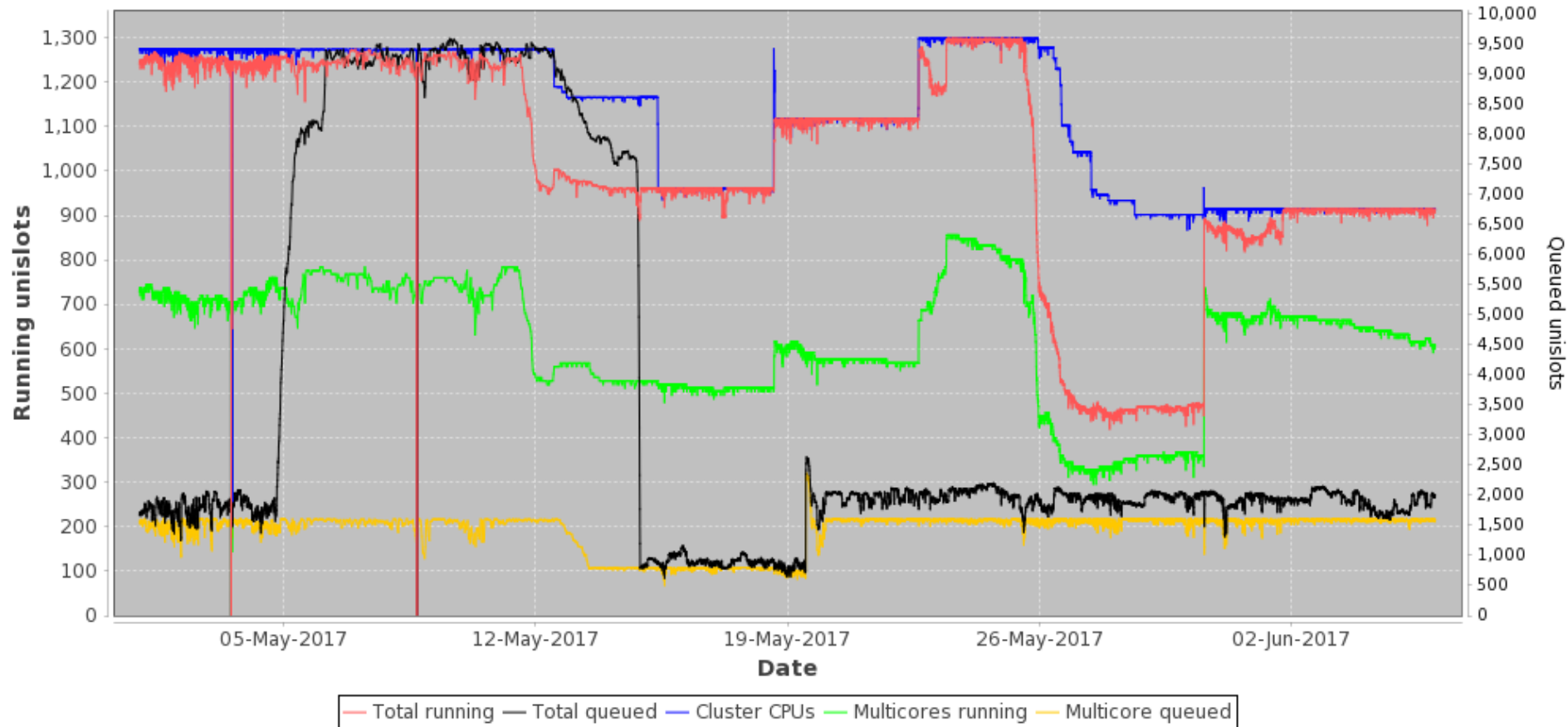




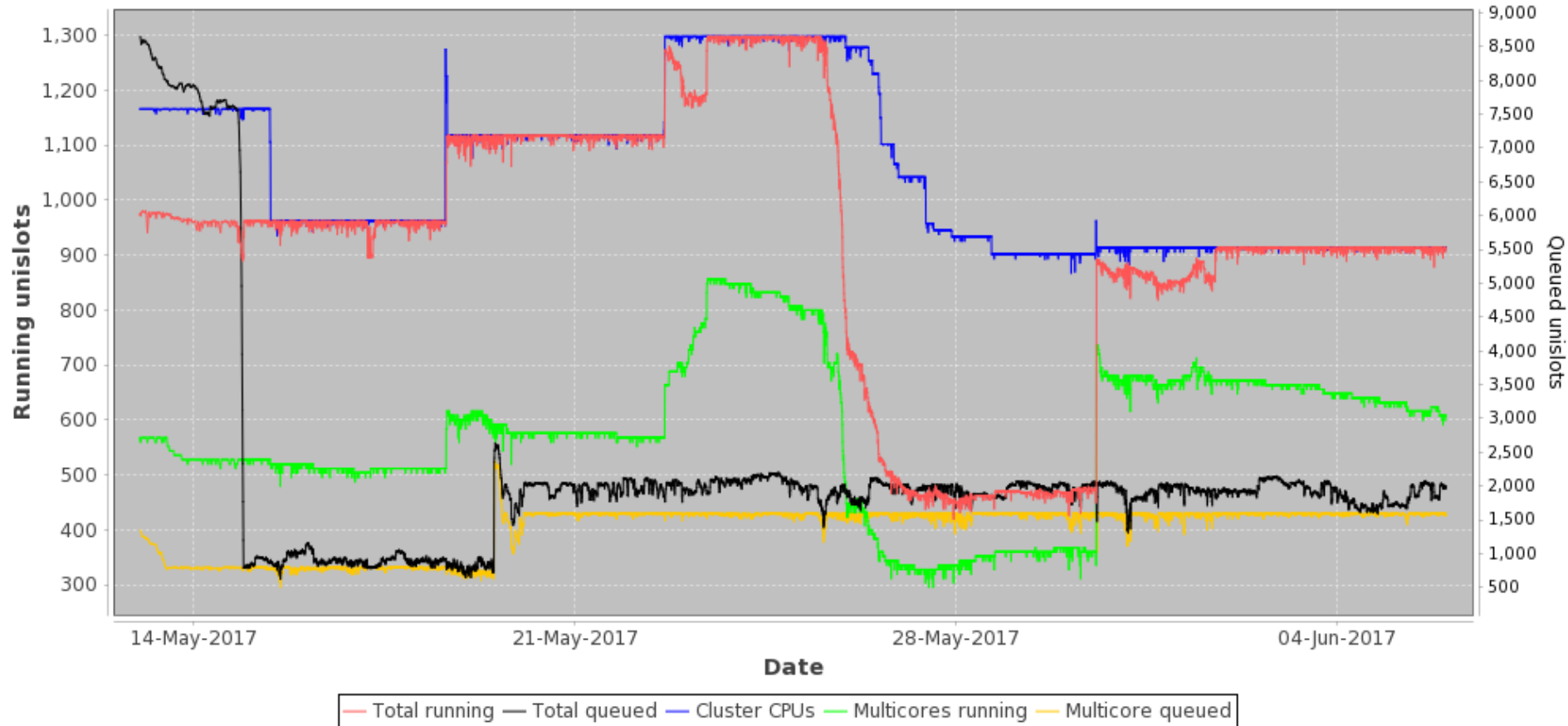
ARC/Condor Cluster Multicore Usage

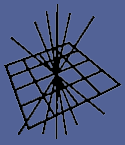


ARC/Condor Cluster Multicore Usage

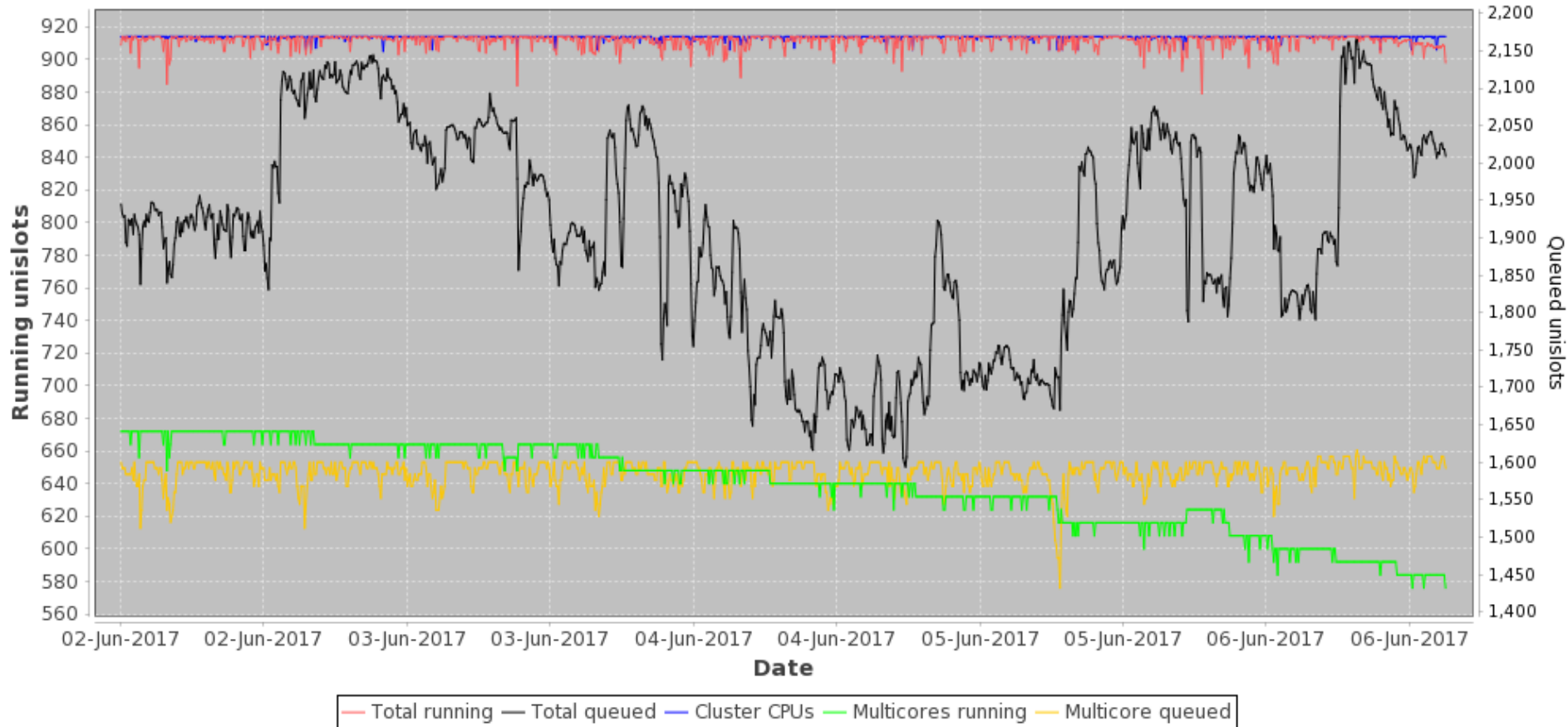


ARC/Condor Cluster Multicore Usage





ARC/Condor Cluster Multicore Usage



- To measure the wastage before the new control function, I take a period of calm operations with no incidents, before the cooling went flaky, i.e. from 15th March for 10 days straight, and use a tool to integrate the gap (delta between slots available and slots occupied). That gives:

```
# ./wastage.pl -f log.out -st "20170316 00:00:00" -d 864000  
CPUs potentially available -- 1274  
Average running cpus -- 1233  
Wastage -- 40, or around 3.14 %
```

- And I measure the calm period in early June as a comparison (see the final plot):

```
./wastage.pl -f log.out -st "20170602 00:00:00" -d 864000
```

```
CPUs potentially available -- 914
```

```
Average running cpus -- 910
```

```
Wastage -- 3, or around 0.36 %
```

- So it looks ~ ten times better now, using the Example ARC/Condor build with version 1.6.1 of Fallow:

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

http://hep.ph.liv.ac.uk/~sjones/fallow-1.6-1.x86_64.rpm

- The ratio I use is normally 66% mcore. I could probably go higher. I'm testing 76% to check the effect. See later...
- It may be possible to expand the scheme for more than one VO, if the mcore slot size is the same.
- As long as a good supply of score and mcore jobs come along, then one can get steady and highly efficient results using the standard techniques described.
- But it takes some time to recover from operational incidents, transient job droughts and so on.
- Another approach (RAL?) is to pre-empt jobs (kill them off). There is virtually no lag there, but it imposes a constraint on the type of job (they must not mind being killed!)

- What caused the improvement?
- First, I got some patches from Chris and Ian at RALPP. One of them allows Fallow to schedule more than one mcore on a node, which is handy for the ratios ATLAS is asking for now. The other patch ranks candidate nodes for draining based on an estimate of the amount of likely “lost-work” (formerly, the choice was a matter of luck.) I think these went into 1.5.1.
- And another good improvement came when I changed the control function quite a bit. The main change was to make Fallow check more circumstances when `_less_` draining is needed, whatever the reason. If that is the case, Fallow chooses the best nodes to put back on (those that are farthest from being drained).
-

A slot is a single/hyperthread/logical cpu.

A node is said to be "draining" when it only can run score jobs.

A node is said to be "drained" when it has 8+ slots slack.

- Get the whole set of nodes, and their individual states/properties.
- For any node that is draining and which has 8 slots slack, stop draining after one negotiation cycle.
- If nothing queued, quit draining all nodes (score draining on all nodes anyway.)
- If only mcore queued, quit draining all nodes (score draining on all nodes anyway.)
- If only score queued, quit draining all nodes (there's no point draining.)
- If you get this far, both mcore and score jobs are in the queue.

Calculate how many more nodes should be draining

- $mcoreDesired = setPoint / 8.0$
- $delta = mcoreDesired - mcoreRunning$
- $delta = delta - beingDrained$
- if delta +ve, we need more draining ...

Find a set (max size: delta) as follows.

Go over the nodes in "nearest to drained" order and select those which would have at least 8 slots of slack once all its score jobs have ended (rejecting nodes that are already being drained, or which already have 8+ slots of slack.)

Start to drain those nodes.

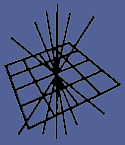
- But if delta -ve, we need less draining ...

Find a set (max size: abs delta) as follows.

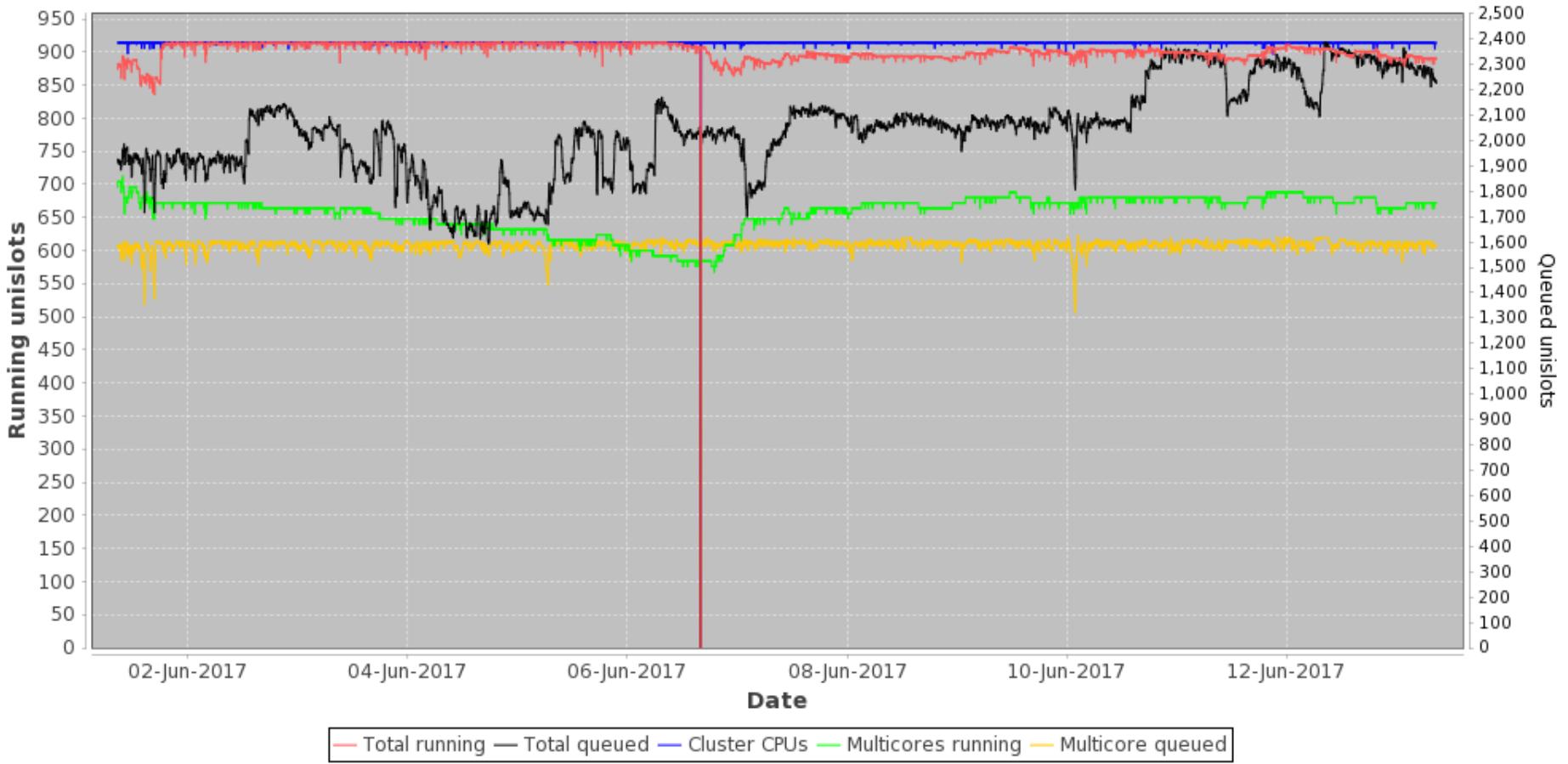
Go over the nodes in "farthest from drained" order and select those that are draining.

Stop draining those nodes.

- And what does “nearest to drained” and “farthest from drained” mean? This is the ranking function supplied by Chris Brew and Ian Loader. When you are selecting a node to drain, you want to chose one that :
 - a) has some free slots already, so that it drains sooner and
 - b) is a node with many slots in the system, since, probabilistically, jobs on such a node end more frequently than they would on a node with few slots.
- Hence the patch “ranks” the list of nodes using those criteria, combined into one value field. Fallow hence selects nodes that are nearest or farthest from being drained by sorting the list of candidate nodes on that rank field, minimising loss of work.
- So when you want to start to drain nodes, you sort in one direction and chose the ones with the least draining to do, and when you cancel a drain, you sort the other way and chose ones that have the most draining to do (that's the best sacrifice).



ARC/Condor Cluster Multicore Usage



- Wastage of ~ 2 %. Much worse (but not too bad.)
- So the highest I can go is actually ~ 680 mcore slots used jobs in 915 slot cluster, or ~ 75%.
- If I try to get a better ratio than that, I get more draining, but the ratio does not improve.
- Why? One theory is that we have a lot of nodes with 10 slots. They can only take 1 mcore no matter how much draining we do!
- I'll look at this and see if we can rearrange the slots per node figures. It might be worth getting less hepspec06 overall, as long as we can get more mcore... worth looking at maybe (in infinite free time).

- A while back I wrote a wiki page on how to get the publishing right.
https://www.gridpp.ac.uk/wiki/Publishing_tutorial
- And I wrote up how use the outputs of this procedure in ARC/Condor arc.conf file.
 - https://www.gridpp.ac.uk/wiki/Example_Build_of_an_ARC/Condor_Cluster#Notes_on_Accounting.2C_Scaling_and_Publishing
- The standard “gLite-style” BDII publishing is used by the APEL accounting to get a site standard benchmark value. And it's has information that can be combined to find a site's installed capacity, for VOs who are interested in that.

- But VAC doesn't have a BDII.
- This doesn't matter to the APEL system, because VAC accounting records contain the HS06 benchmark used on the factory system. They are self contained.
- But it does matter to the installed capacity. Basically, VAC is not counted. So there is a choice to make.
 - a) Live with it. VOs can't determine the installed capacity at a site. That's the VAC ethos; systems are opportunistic, even if they are not. Does anyone really care? What about VAC only sites?
 - b) Configure ldif records (or a special BDII) to hold the VAC figures and transmit them. A fair bit of work, not a standard service node, and it goes against the “VAC ethos”. Will the BDII even matter soon?
 - c) Clump the installed capacity values into the standard BDII, and munge the values to make them “valid”. We do this.

- Munging is made easier with an app; Site Layout DB
- The AGGREGATE cluster set is used to do the munging.

Cluster - Mozilla Firefox

File Edit View History Bookmarks Tools Help

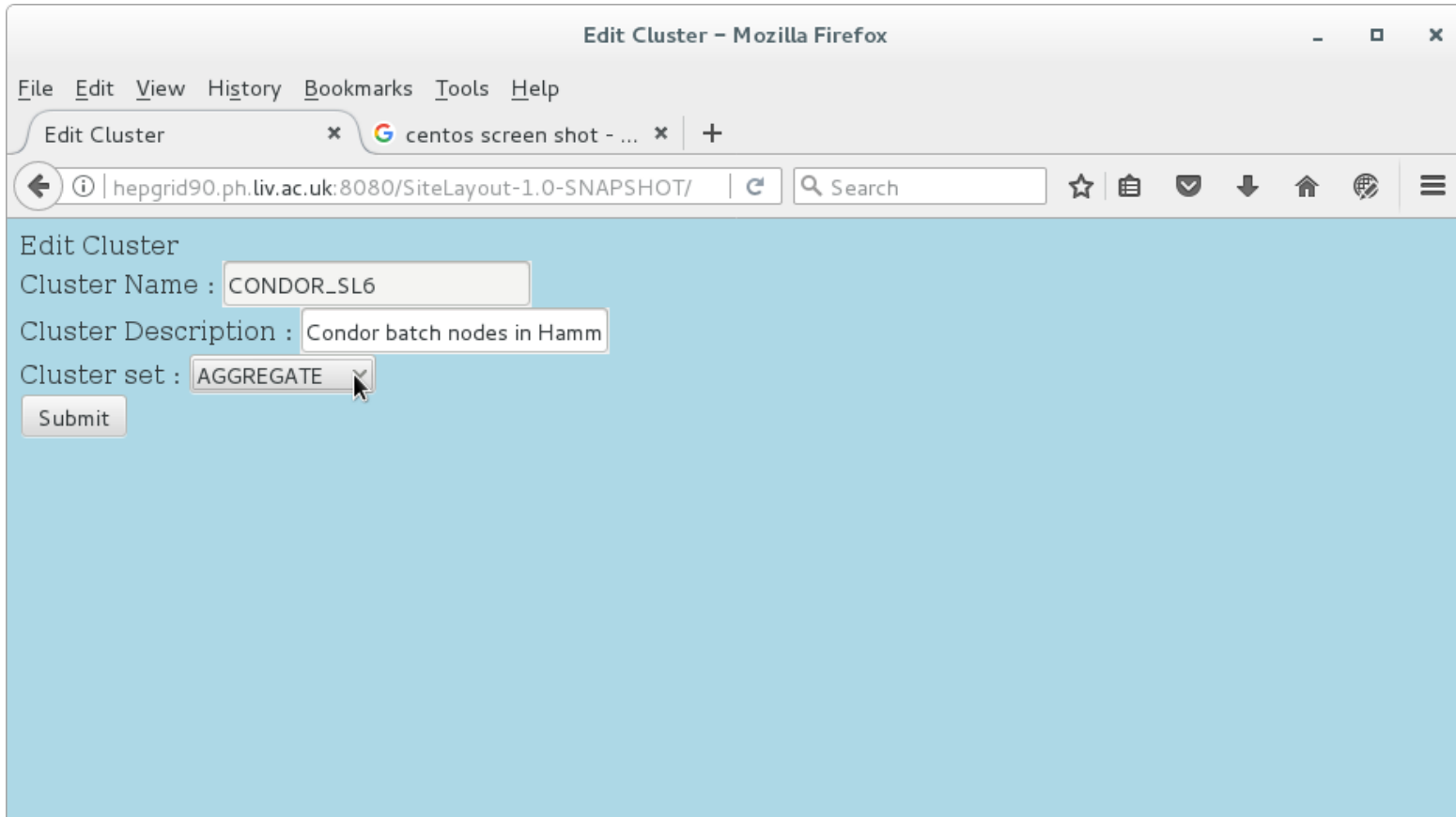
Cluster centos screen shot - ...

hepgrid90.ph.liv.ac.uk:8080/SiteLayout-1.0-SNAPSHOT/

Cluster

Cluster name ↑	Description ↑	Cluster set name ↑		
AGG	AGG	AGGREGATE	Del	Edit
CONDOR_C7	Centos7 Test	CONDOR_C7	Del	Edit
CONDOR_SL6	Condor batch nodes in Hammer server room	CONDOR_SL6	Del	Edit
VAC_SL6_LOCAL	Vac cloud nodes in the Hammer server room	VAC	Del	Edit
VAC_SL6_REMOTE	Vac cloud nodes in the Chadwick server room	VAC	Del	Edit
Back	New			

- Add the clusters to the aggregate cluster set, one by one (a “cluster set” is analogous to a Site.)



Edit Cluster – Mozilla Firefox

File Edit View History Bookmarks Tools Help

Edit Cluster × centos screen shot - ... × +

hepgrid90.ph.liv.ac.uk:8080/SiteLayout-1.0-SNAPSHOT/ Search

Edit Cluster

Cluster Name :

Cluster Description :

Cluster set :

- Like this ...

Cluster - Mozilla Firefox

File Edit View History Bookmarks Tools Help

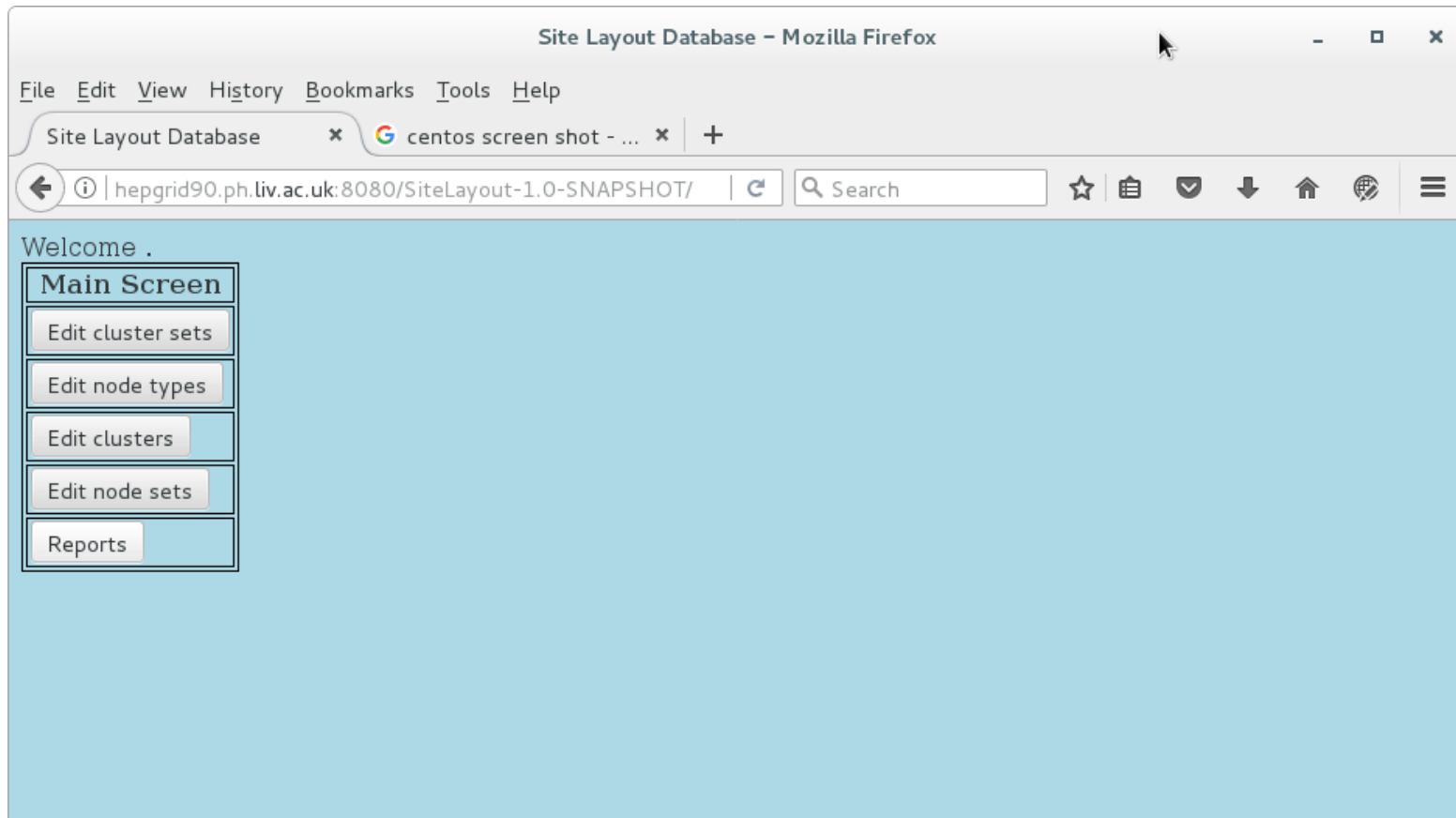
Cluster centos screen shot - ...

hepgrid90.ph.liv.ac.uk:8080/SiteLayout-1.0-SNAPSHOT/

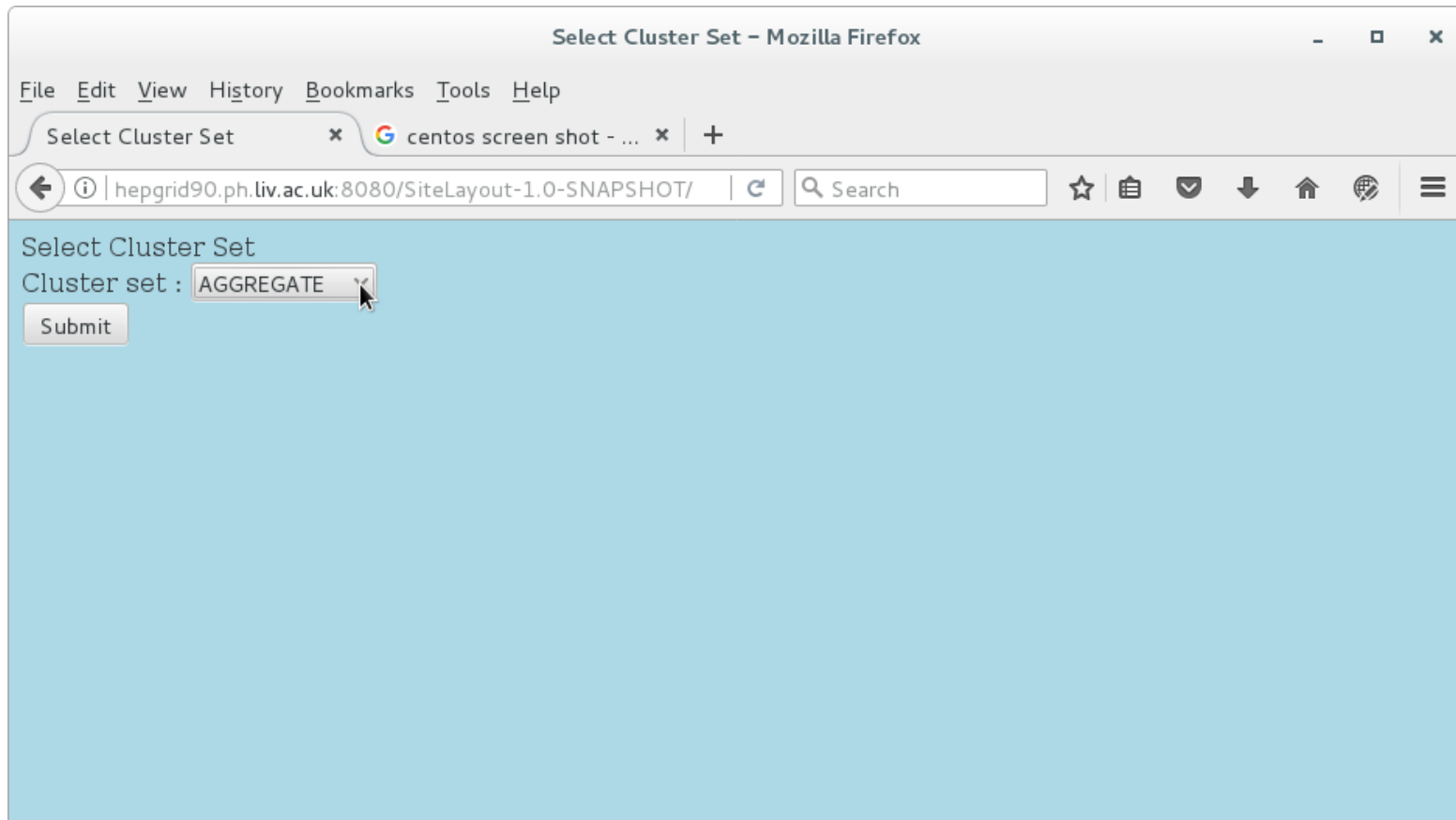
Cluster

↓ Cluster name ↑	↓ Description ↑	↓ Cluster set name ↑		
AGG	AGG	AGGREGATE	Del	Edit
CONDOR_C7	Centos7 Test	CONDOR_C7	Del	Edit
CONDOR_SL6	Condor batch nodes in Hammer server room	AGGREGATE	Del	Edit
VAC_SL6_LOCAL	Vac cloud nodes in the Hammer server room	AGGREGATE	Del	Edit
VAC_SL6_REMOTE	Vac cloud nodes in the Chadwick server room	AGGREGATE	Del	Edit
Back	New			

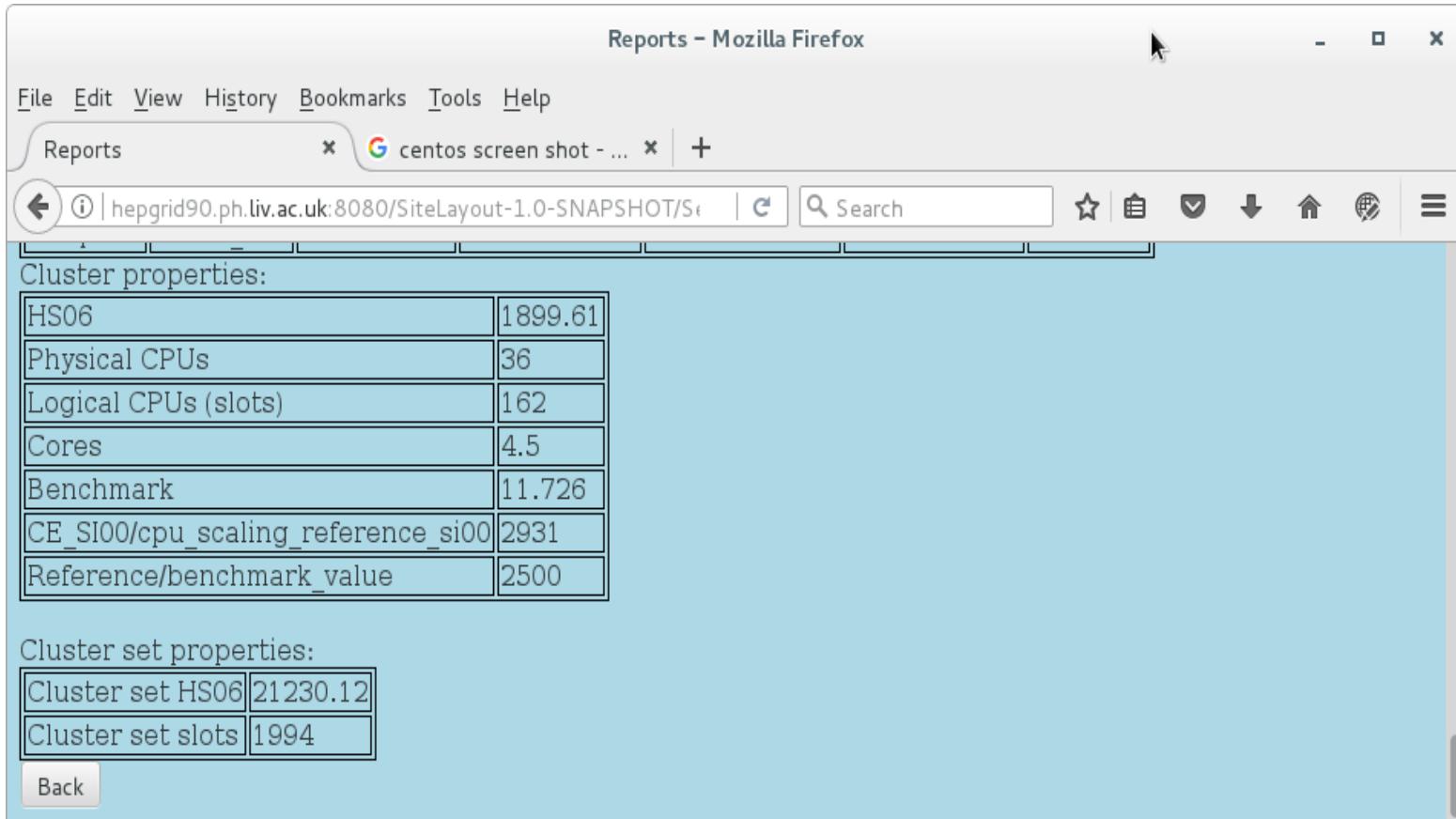
- Then use the Reports menu...



- And select the report for the AGGREGATE cluster set



- And I get the values I need to put in the arc.conf.



Reports - Mozilla Firefox

File Edit View History Bookmarks Tools Help

Reports centos screen shot - ...

hepgrid90.ph.liv.ac.uk:8080/SiteLayout-1.0-SNAPSHOT/S... Search

Cluster properties:

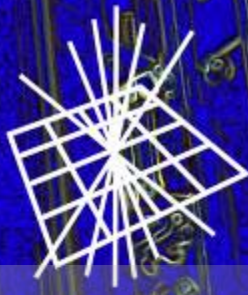
HS06	1899.61
Physical CPUs	36
Logical CPUs (slots)	162
Cores	4.5
Benchmark	11.726
CE_SI00/cpu_scaling_reference_si00	2931
Reference/benchmark_value	2500

Cluster set properties:

Cluster set HS06	21230.12
Cluster set slots	1994

Back

- Obviously I have to reset it all after.
- Could easily be done any way you like, of course. Spreadsheet, maybe?
- But the SLDB has lots of other nice to have thingies.
- I tend to use it like this:
 - Plan design changes for the site.
 - Update the SLDB to model the changes and see what's what,
 - Then actually update the racks, nodes, operating systems, networks, middleware, install new nodes, memory, whatever. Move the nodes around to create new clusters, dismantle old ones...
 - Then update the BDII with the values already in the SLDB.
- So it's a life cycle tool. I don't know now how I coped without it. It's either more accurate or easier, or maybe both at once.
- Discussion on how sites manage, and how they use their SLDBs.
- I.e. what are the other requirements for automation etc.
- Could I run an entire site scripted entirely from the SLDB, so I no longer need to use vi?
What else is out there?



GridPP

UK Computing for Particle Physics

How to “efficiently” build Centos7
HTCondor workernodes, without Yaim,
for UMD release testing.

- In 2017, we were asked to set-up a version 7 cluster, to test UMD distribution with (in our case) Centos7 workernodes.
- We have had jobs arriving from the atlas, gridpp, dteam, t2k, lhcb, ilc and ops VOs.
- V 7 cluster is very small. Not much traffic. LHCb long lasting jobs. ATLAS not picking up pilots. Haven't tried mcore.
- Decent documentation is available on the install we used.
 - https://www.gridpp.ac.uk/wiki/Centos7_Adoption
- This documentation also :
 - a) provides tools to allow us to finally abandon Yaim altogether.
 - b) has a simple introduction to Puppet3 and Hieria (which is about all I know.)

- How we build our Centos7 HTCondor workernodes, without Yaim (I only give the salient points.)
https://www.gridpp.ac.uk/wiki/Centos7_Adoption
- A repository of UMD middle-ware code has been ported to Centos7 and is available for testing and early adoption (Andrea Manzi)
 - Note: wn-emi will now be called wn (which might be the shortest package name ever.)
- The headnode for this mini-cluster is a normal HTCondor headnode running on SL6. The build for that is documented in the GridPP wiki:
https://www.gridpp.ac.uk/wiki/Example_Build_of_an_ARC/Condor_Cluster
- Basis of build is a plain Centos7 worker-node build to local site standards using “CentOS Linux release 7.3.1611 (Core)”, and the xfs filesystem. Build is completed by Puppet and Hiera.

- High-level build system
- Puppet gives declarative means to define the end-state
- Hiera gives a way to define parameters used during the puppet build; useful for making modules portable.
- Puppet modules contain parameters that will be resolved by Hiera when the node is built. Hiera looks them up in a user-defined hierarchical database.
- Hence general, default settings can be generally defined for “all” nodes by default, while more specific settings can be defined where a particular node in the set needs something special.
- More general → more specific implies a Hierarchy...
- Our v7 workernode are defined in Hiera in a “nodetype” file (much code omitted, related to security, ganglia etc.)

```
# cat environments/production/hiera/nodetype/condor-c7-L5420.yaml
---
classes:
  - grid_users
  - condor_c7
  - cvmfs
  - grid_hosts

condor_c7::params::ral_node_label: "L5420"
condor_c7::params::ral_scaling: "0.896"
condor_c7::params::num_slots: "1"
condor_c7::params::slot_type_1: "cpus=8,mem=auto,disk=auto"
condor_c7::params::num_slots_type_1: "1"
condor_c7::params::slot_type_1_partitionable: "TRUE"
condor_c7::params::processors: "8"
condor_c7::params::numberOfCpus: "2"
condor_c7::params::headnode: "igrid5.ph.liv.ac.uk"

cvmfs::cvmfs_quota_limit: '20000'
cvmfs::cvmfs_http_proxy: 'http://hepsquid1.ph.liv.ac.uk:3128|
http://hepsquid2.ph.liv.ac.uk:3128'
cvmfs::cvmfs_cache_base: '/var/lib/cvmfs'
```

```
yum_repositories:  
  local:  
    descr: "local RPMs"  
    baseurl: "http://map2.ph.liv.ac.uk/yum/local/rhel/7/x86_64"  
    enabled: "1"  
    protect: "1"  
    gpgcheck: "0"  
    priority: "1"  
  epel:  
    descr: "Extra Packages for Enterprise Linux 7 - $basearch"  
    baseurl: "http://map2.ph.liv.ac.uk/yum/ONLINE/pub/epel/7/$basearch"  
    enabled: "1"  
    gpgkey: "http://dl.fedoraproject.org/pub/epel/RPM-GPG-KEY-EPEL-7"  
    gpgcheck: "1"  
    priority: "99"
```

An actual node, say r26-n15.ph.liv.ac.uk, is associated with this particular nodetype via another Hiera yaml file, thus defining a concrete node.

```
cat environments/production/hiera/node/r26-n15.ph.liv.ac.uk.yaml
---
classes:
-
nodetype: "condor-c7-L5420"
# condor_c7::params::someparm : "somevalue"
```

- The trouble is that Hieradata doesn't actually have a "nodetype" level in its hierarchy. We need it to avoid lots of identical node definitions containing duplication.
- So we insert one as follows:

```
# cat /etc/puppet/hiera.yaml
---
:backends:
  - yaml
:hierarchy:
  - node/%{::clientcert}
  - nodetype/%{::nodetype}
  - os/%{::operatingsystem}/%{::operatingsystemmajrelease}
  - os/%{::operatingsystem}
  - site-info
  - common
```

Etc. etc.

- And we add a line to a Hiera config file to make Hiera itself lookup the value of “nodetype”.
- `vi /etc/puppet/environments/production/manifests/site.pp`
- And explicitly add this line to that file to read and set the nodetype variable.
`$nodetype = hiera('nodetype')`

- The important modules are described in this section:

https://www.gridpp.ac.uk/wiki/Centos7_Adoption#Puppet_module_layout

- The main Puppet modules used in the procedure are provided in a tar ball.

<http://hep.ph.liv.ac.uk/~sjones/Centos7/modules.tar>

- They consist of:
 - `grid_users`: replaces Yaim (`config_users`) and runs 100 times faster.
 - `condor_c7`: install HTCondor on the worker node and makes it ready to use.
- Details are given in the document referenced.

- Is there such a thing as a normal T2 now?
- Publishing calculations efficiency/accuracy. Does the BDII matter. Do we care about installed capacity?
- VAC efficiency/tuning
- Mcore efficiency/tuning
- Puppet/Hiera admin efficiency/tuning