Imperial College London

Usage of the CMS High Level Trigger as a Cloud Resource

David Colling

d.colling@imperial.ac.uk

Imperial College London

Acknowledgments

This work has been carried with contributions from several people (at different times):

- Adam Huffman (Imperial College, Grid/Cloud devops),
- Alison McCrea (CERN, CMS operations),
- Andrew Lahiff (RAL and CMS Operations),
- Mattia Cinquilli (CERN, IT Support for Distributed Computing),
- Stephen Gowdy (CERN),
- Jose Antonio, Coarasa (CERN, CMS Online devops),
- Anthony Tiradani (Fermilab, CMS glideinWMS),
- Wojciech Ozga (CERN and AGH University of Science And Technology in Krakow, CMS Online devops),
- Olivier Chaze (CERN, CMS Online devops),
- Daniela Bauer (imperial College)

+ Anybody I have missed out (apologies)

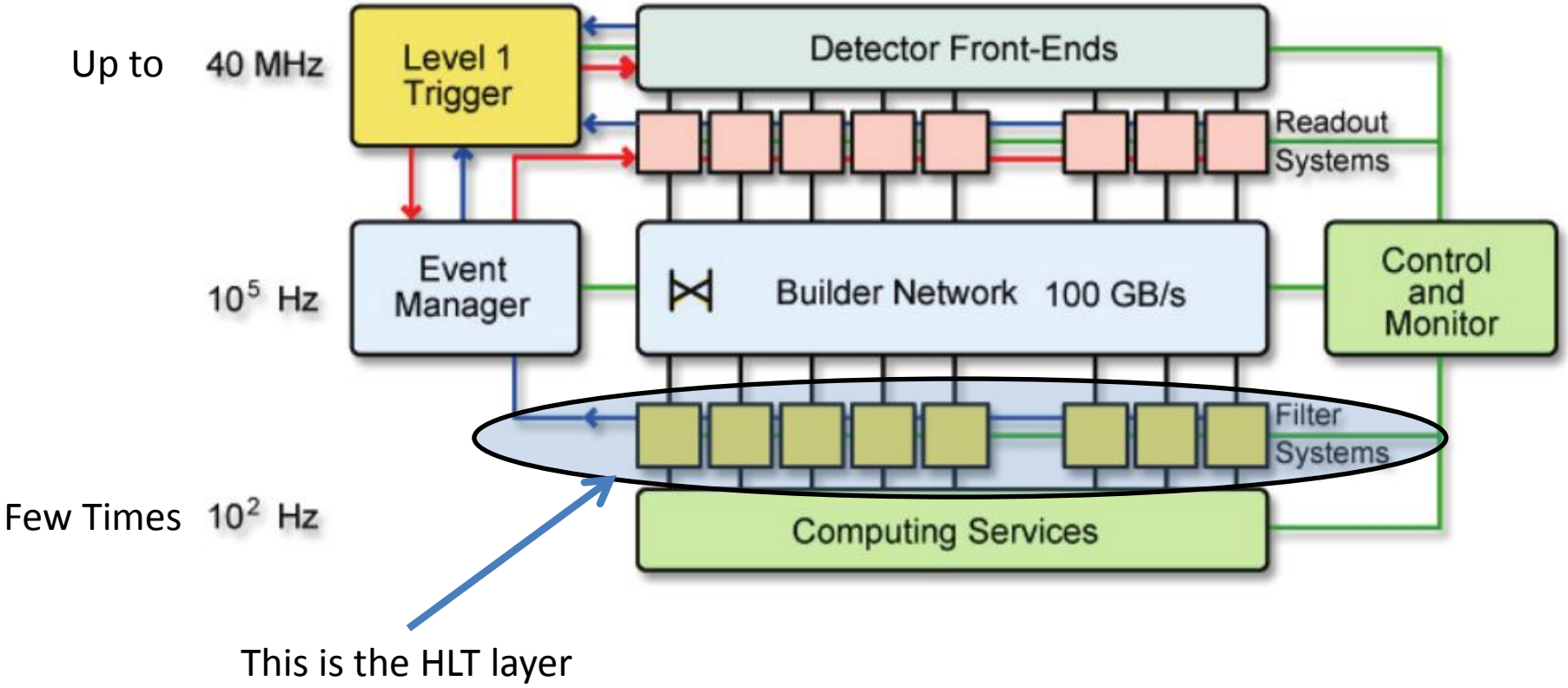
Imperial College London

Content

- What is the High Level Trigger (HLT)
- Why do we want to use it as a cloud resource and the constraints on using it as a cloud resource.
- The different components in using it as a cloud resource:
  - Changes that were made to HLT cluster
  - Data Access
  - Conditions Data Access
  - Access to CMS software
  - Job submission
- What happened when we tried to use the HLT as a cloud
- Status and on going activity

What is the HLT (for those not familiar)

The CMS trigger design in Run 1



HLT in terms of resources

At the start of 2013 the HLT consisted of:

cluster	Nodes	cores (HT on)/ node	cores	Memory (Gbyte/node)	Disk (Gbytes/node)
(1)	720	8	5760	16	72
(2)	288	12 (24)	3456	24	225
(3)	256	16 (32)	4096	32	451
(1)+(2)+(3)	**1264**		**13312**	**26 Tbytes**	**227 Tbytes**

- Three generations of hardware
- Variety of network links to CERN and outside world (1Gb/ and 20Gb/s)
- (Essentially) no storage
- Wholly owned by CMS
- Single use cluster – did one particular job very well indeed, not obviously suited to be a general purpose cluster

But look at the scale of the HLT…

In raw compute power, the HLT at the end of 2012 was ~195HS06 (only ~150HS06 easily usable). Compare with the CMS total requests:

HLT	Tier 0	Tier 1 (total)	Tier 2 (total)
195 (150)	121	150	399

So the resources of the HLT were as large as the total T1 request and larger than the Tier 0.

2015 HLT will be ~200KHS06 all easily usable.

Not a resource that we can afford to ignore

The Plan…

- Use the HLT as a production resource during LS1
- Then use it when ever possible during Run 2:
  - Machine breaks
  - May be in the gaps between fills???
  - Possibly even if running conditions mean HLT is not full (heavy ion running????????)
- Important that the HLT could run all CMS workflows not just (say) event generation.

Imperial College London

So why use it as a cloud?

- Primarily it is the HLT— can only be used for other things when not needed as a trigger
  - This means that Online Computing must retain control,
  - Only “spare” resources available to Offline computing.
  - Must make minimal changes to the underlying hardware.
  - Offline computing must be able to migrate on and (especially off) at very short notice (15 minutes cited)
  - Overlaying a cloud on top of the existing infrastructure seems ideal
- Virtualisation is a good way of dealing with heterogeneous resources – VMs will only run on resources capable of running them
- Finally, gaining experience of how to run clouds on other transient resources

Imperial College London

So a cloud was born ...

Decided to create the "CMS openstack, opportunistic, overlay, online-cluster Cloud" or CMSooooCloud for short.

Why openstack?

- Large, growing and dynamic development community
- Open source
- Widely believed to have a solid underlying architecture
- "speaks EC2"
- Being driven by a huge and growing user community

"**People Get Pissed Off About OpenStack. And That's Why It Will Survive"**
http://techcrunch.com/2012/08/13/people-get-pissed-off-about-openstack-and-thats-why-it-will-survive/

Imperial College London

Changes to the HLT

- After initial tests in late 2012 and early 2013, it seemed viable to run the HLT as a cloud for production.
- OpenStack (Essex) was overlaid on top of the HLT
- Crucially at this stage all data not directly from CMS data taking went into and out of the HLT over the 1Gb/s link
- But because the HLT had been a single purpose cluster we also had to introduce appropriate monitoring etc …

Services needed – Data Access

With the lack of disk at the HLT, this would once have been a real problem.

However CMS AAA (see https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookXrootdService) allows access to CMS data from anywhere with a network connection over xrootd.

Specifically, we decided to read and write data on EOS at CERN over xrootd.

Imperial College London

Services needed – conditions data

CMS conditions data is served from the Frontier caching service (based on squid).

This we thought this was simple, we just needed to point the VMs at a Frontier server.

Imperial College London

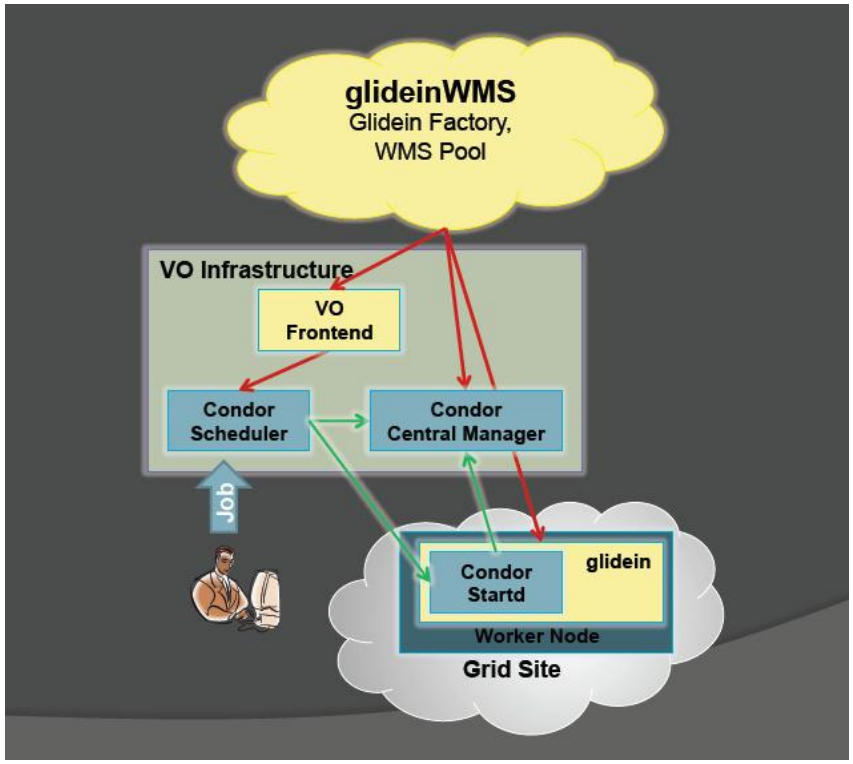
Services needed— CMS software

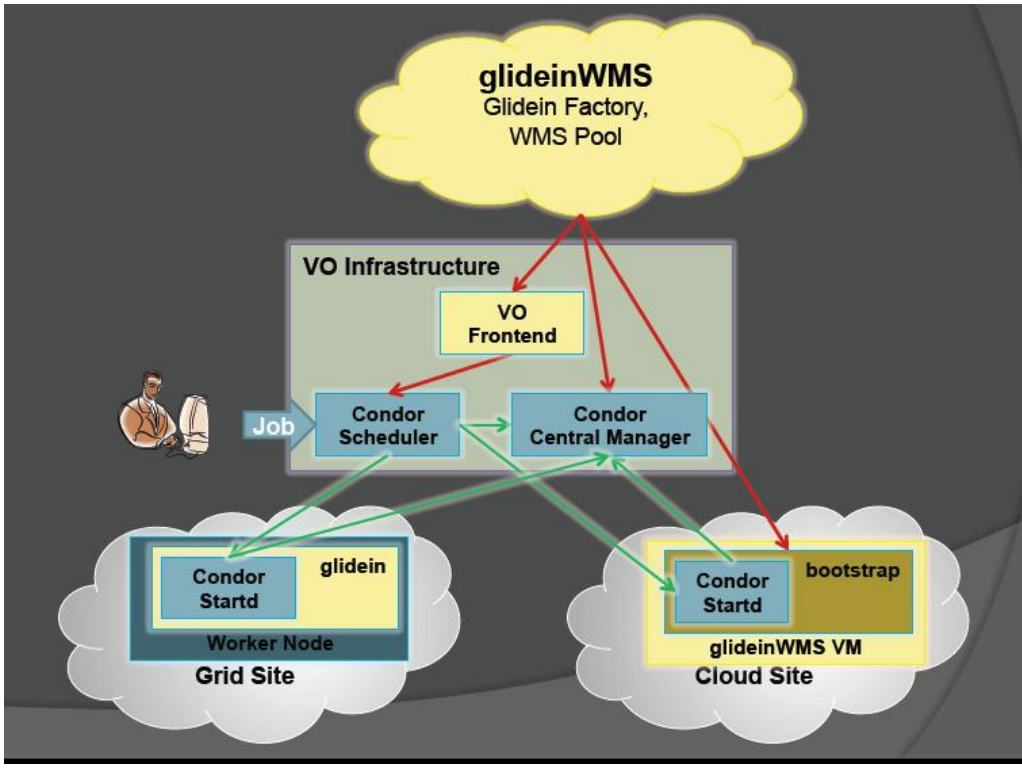
Could have built this directly in to the VMs.

However this would have meant specific images only capable of using specific releases and would not scale generally.

- Instead decided to use CvmFS. Which serves software releases through squid caches (http://cernvm.cern.ch/portal/filesystem).

glideinWMS


glideinWMS
Glidein Factory,
WMS Pool
VO Infrastructure
VO
Frontend
Condor
Scheduler
Condor
Central Manager
Job
Condor
Startd
glidein
Worker Node
Grid Site


glideinWMS
Glidein Factory,
WMS Pool
VO Infrastructure
VO
Frontend
Job
Condor
Scheduler
Condor
Central Manager
Condor
Startd
glidein
Worker Node
Grid Site
Condor
Startd
bootstrap
glideinWMS VM
Cloud Site

Imperial College London

Initial results

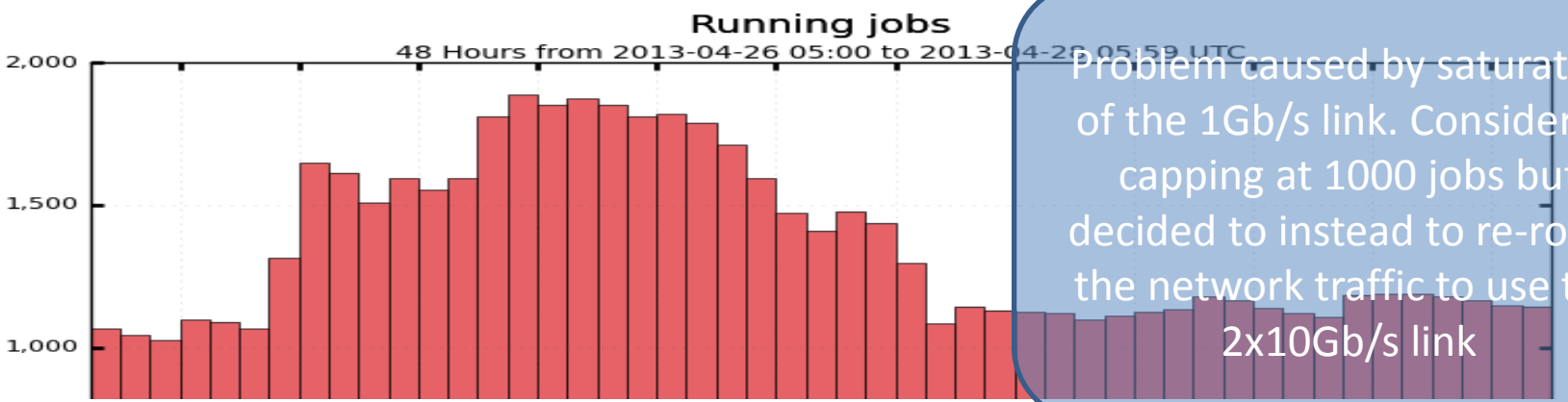
Found many, often minor but annoying, problems.

These include:

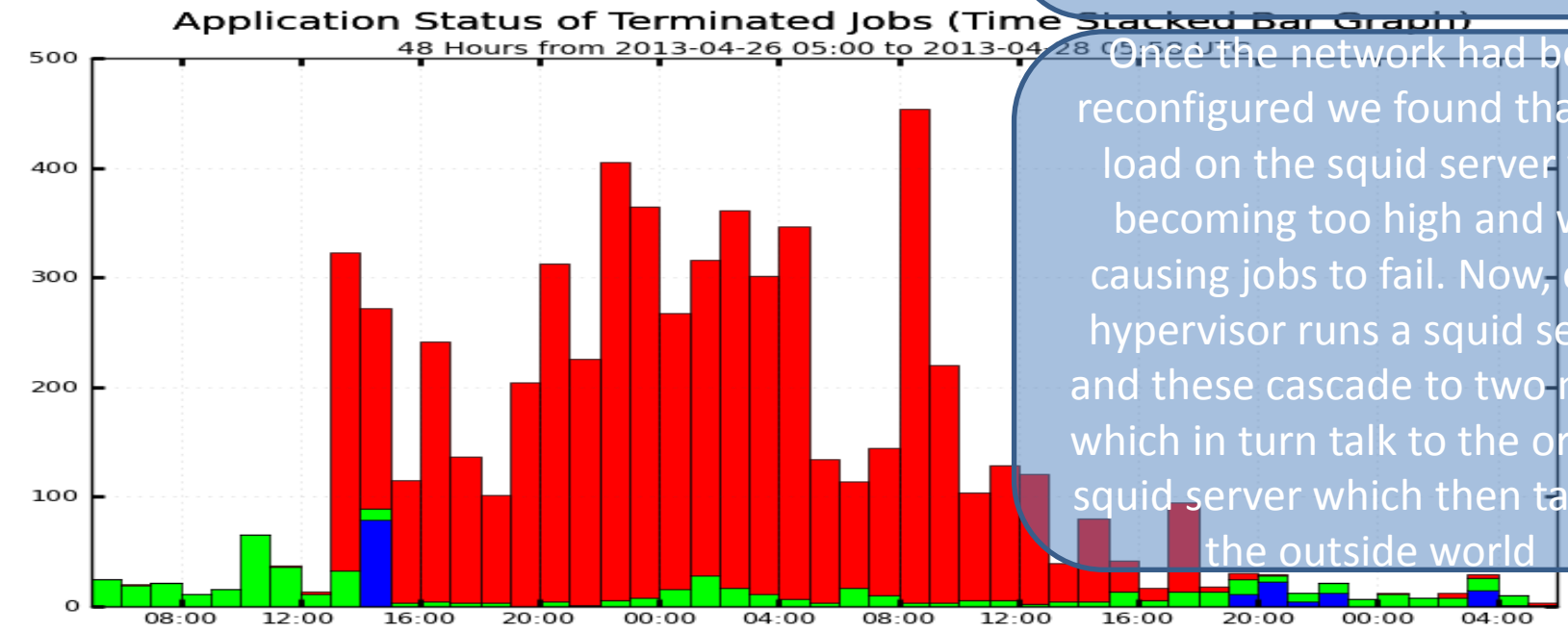
- Permissions problems with xrootd and EOS
- VMs dying because access to CvmFS was not available fast enough
- OpenStack EC2 not Amazon EC2 causing many minor problems all of which required modifications to the glideinWMS.
- Behaviour in clouds is different from behaviour in Grids so glideinWMS needed to learn how to handle the situations differently
- OpenStack controller can be “rather fragile” when asked to do things at scale so glideWMS learnt to treat it gently.
- glideinWMS losing track of jobs (often through fragility of OpenStack) and jobs ending up in “shutoff” state
- ...

Gradually been working our way through these with the set up becoming functional and then more robust as we go...

Initial results – basic limitations



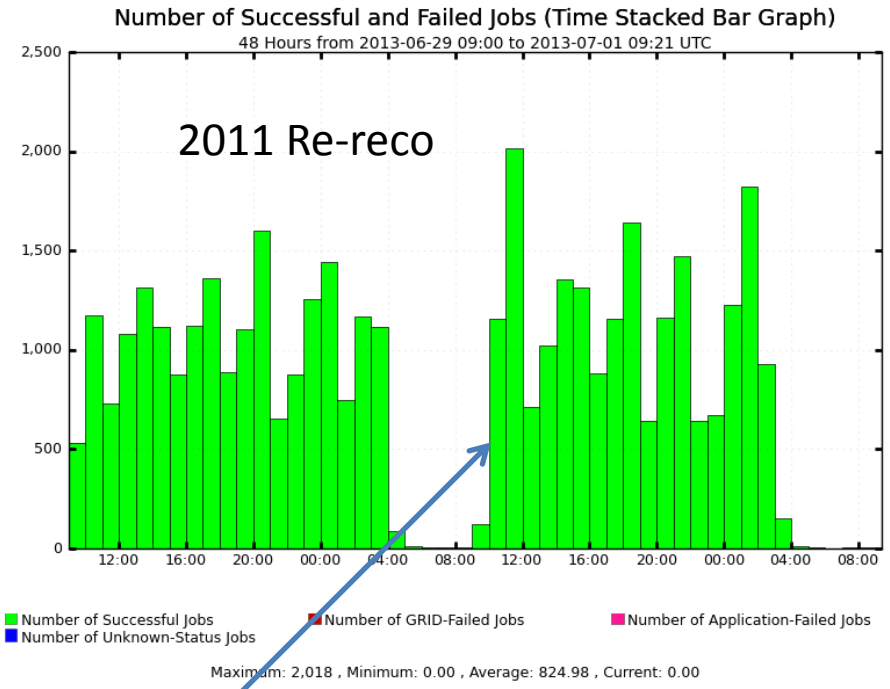
Problem caused by saturation of the 1Gb/s link. Considered capping at 1000 jobs but decided to instead to re-route the network traffic to use the 2x10Gb/s link

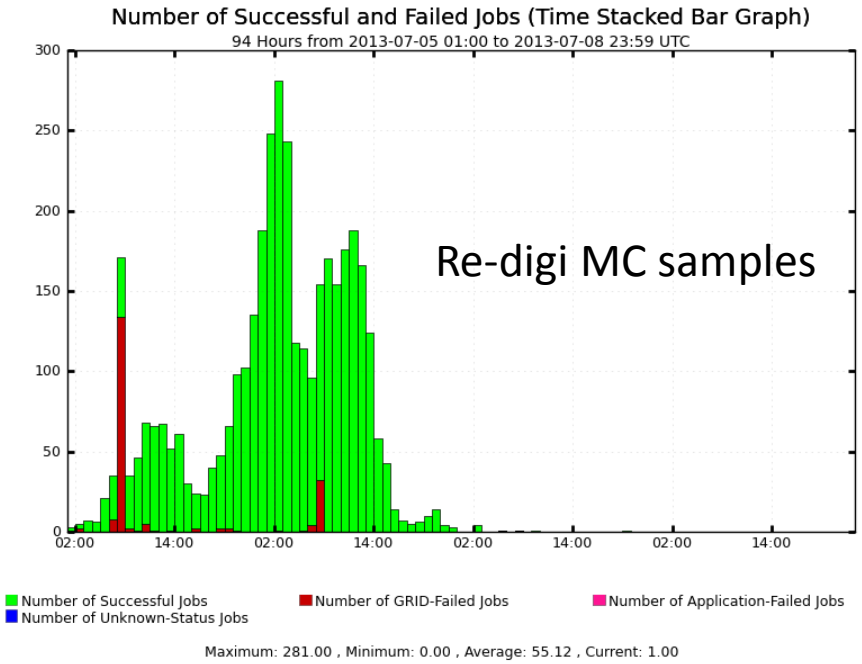


Once the network had been reconfigured we found that the load on the squid server was becoming too high and was causing jobs to fail. Now, each hypervisor runs a squid server and these cascade to two-nodes which in turn talk to the original squid server which then talks to the outside world

June and July

- used as a production resource





~6000 Jobs running
If only 1000 finishing/hour

Two workflows that were run

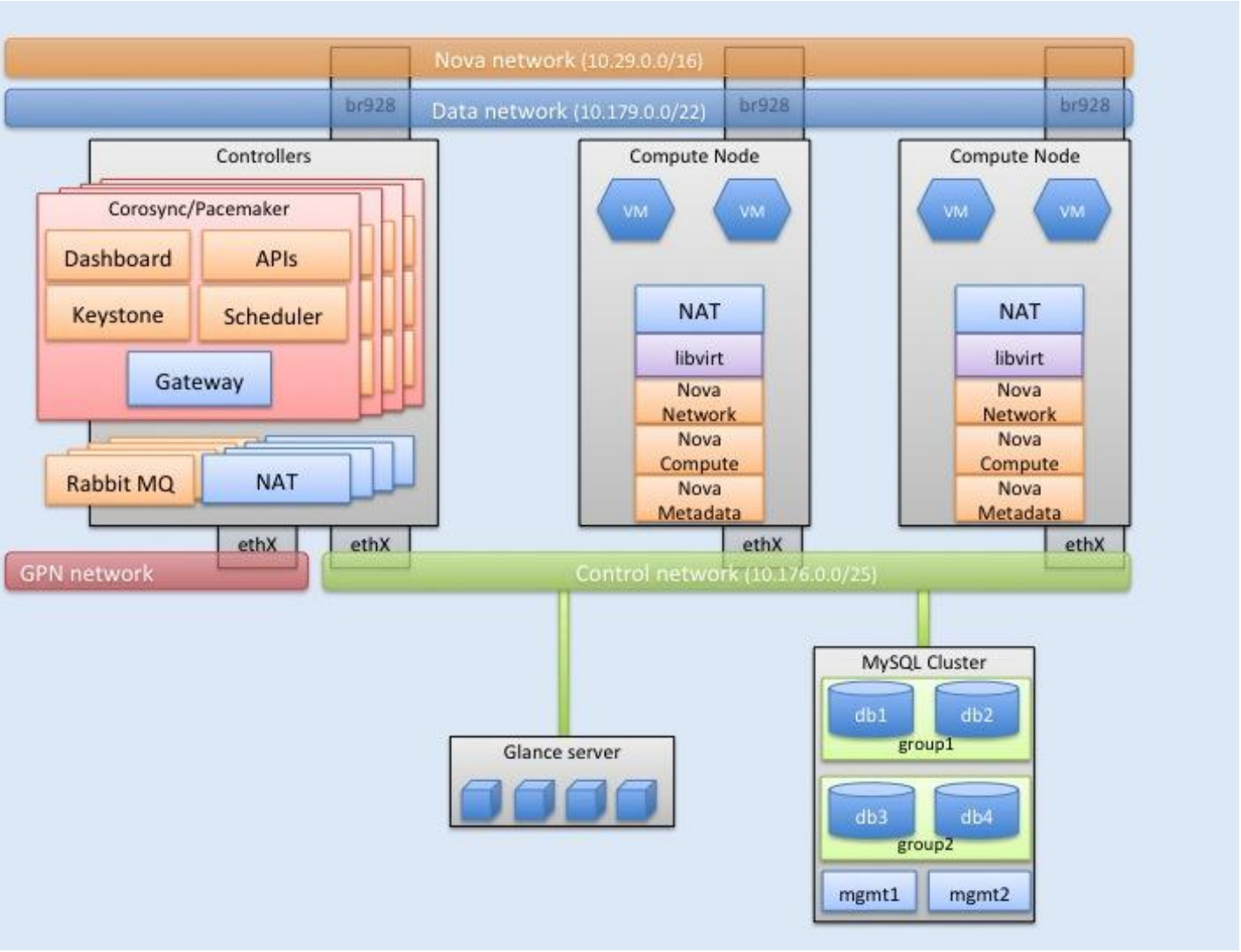
Current Status

Over the summer the service degraded

- Lack of work meant that it wasn't constantly utilised.
- People upgraded components (both software and hardware)
- Distributed management of different components, along with changes of manpower and holidays meant delays in debugging.
- Eventually (recently) formed distributed working group to resolve problems.

This worked and, only recently, back to being production quality

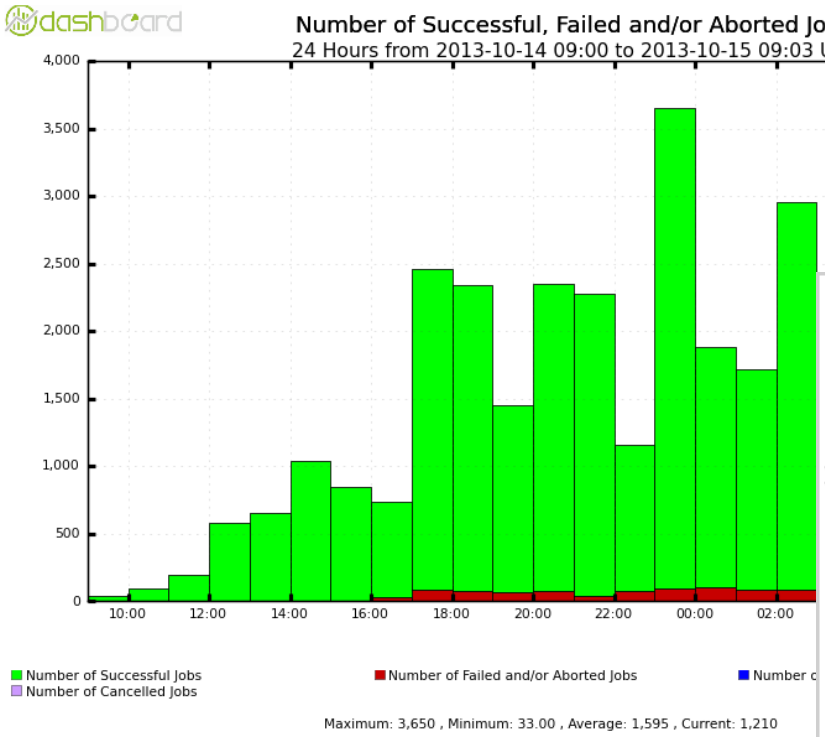
Current Status

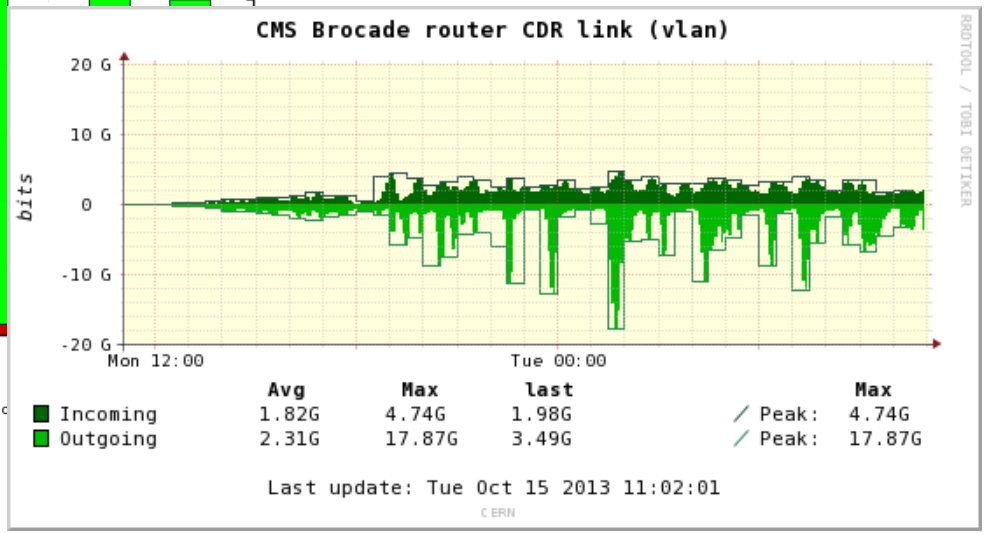


Moving to Grizzly and this configuration

Current Status

Running production workflow as final test before restarting real work on the HLT cloud …





Looking to the future

Now that the resource is working again.

- Network will need to be upgraded – currently run at a (few) thousand jobs. Need to be able to run ~15000 jobs concurrently by 2015
- Increasing monitoring of VMs themselves (especially) e.g. hammercloud testing when not actually being used in production.
- Starting to look at migration strategies so that capable of running during data taking.
- Important to keep good communications between different teams running the different components

Imperial College London

Conclusions

- The HLT is a resource that we need to use
- The HLT "naturally" should work as a cloud
- The HLT does work as a cloud resource
- The HLT cloud should be able to run all CMS workflows
- We have learnt a lot by running the HLT as a cloud. Much of this is applicable to running over other cloud resources.
- Human communications between different groups are important in providing the HLT as a cloud resource.