CPU resource provisioning towards 2022: implementations

Andrew McNab
University of Manchester
LHCb and GridPP

Overview

- Goals going forward
- Simplify the “Grid with Pilot Jobs” model
- Virtual grid sites
- The Vacuum Model
- Vac, Vcycle, HTCondor Vacuum
- Containers
- Volunteer, BOINC, @home
- Opportunistic HPC

(In these slides, “T4:519:Tue” = CHEP Track 4, Talk 519, on Tuesday)

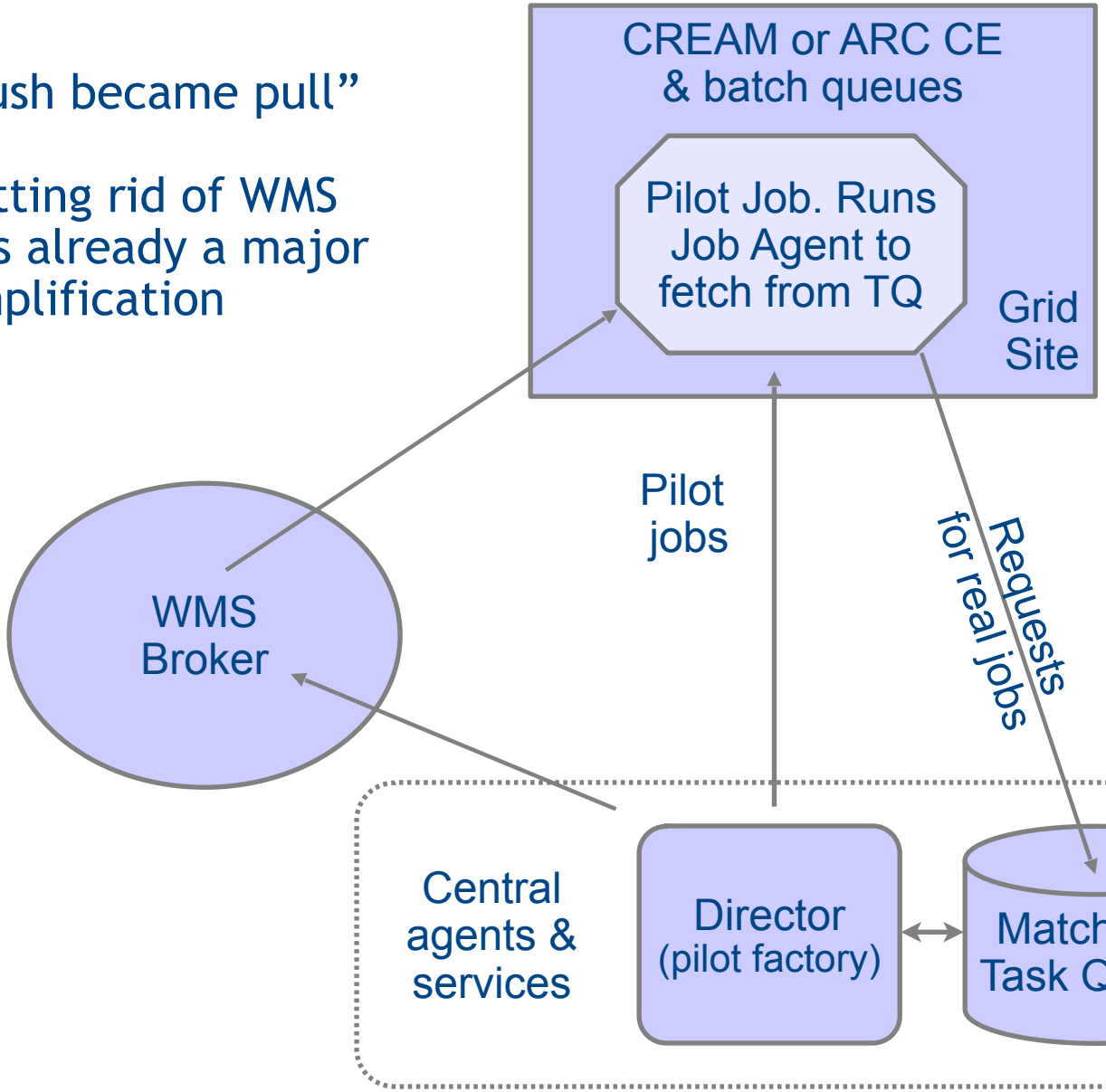
Goals going forward

- Landscape for the next few years shaped by data, technology and money
- Higher event rates mean more data, and more CPU
- Flat-cash funding or outright cuts mean less people
- “How can we do more with less?”
- Simplify what we have, to work more efficiently whilst retaining the functionality we really need
- Themes:
  - Refactoring existing grid
  - Virtualization
  - Use mainstream technologies (eg Cloud)
  - Opportunistic / volunteer resources
  - What implementations are going to be part of that landscape?

The Grid with Pilot Jobs

"Push became pull"

Getting rid of WMS was already a major simplification

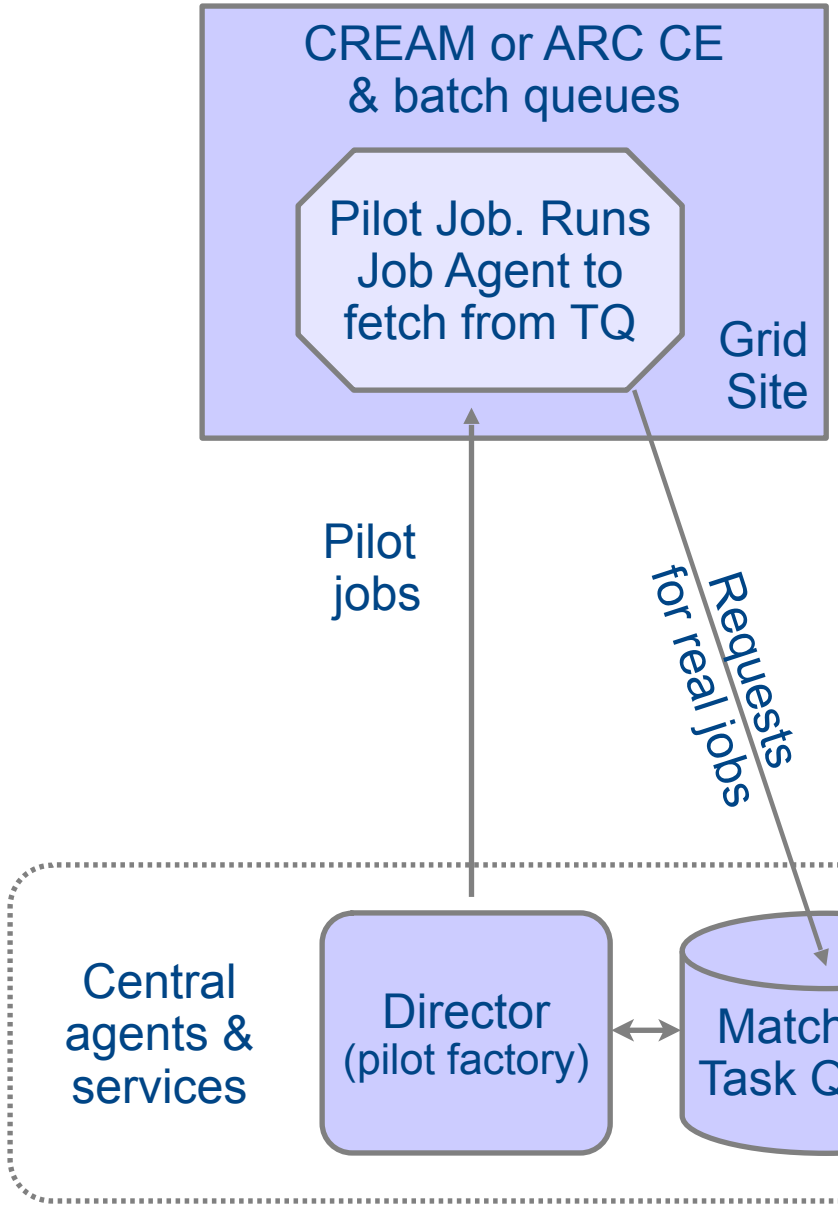


The Grid + pilot jobs is still the dominant model for running HEP jobs.

Well established, and gives access to resources around the world.

But it's very HEP-specific, and relies on a lot of "middleware" which we have to maintain ourselves.

The Grid with Pilot Jobs



Last year or so has seen a big move away from PBS+CREAM towards Condor+ARC

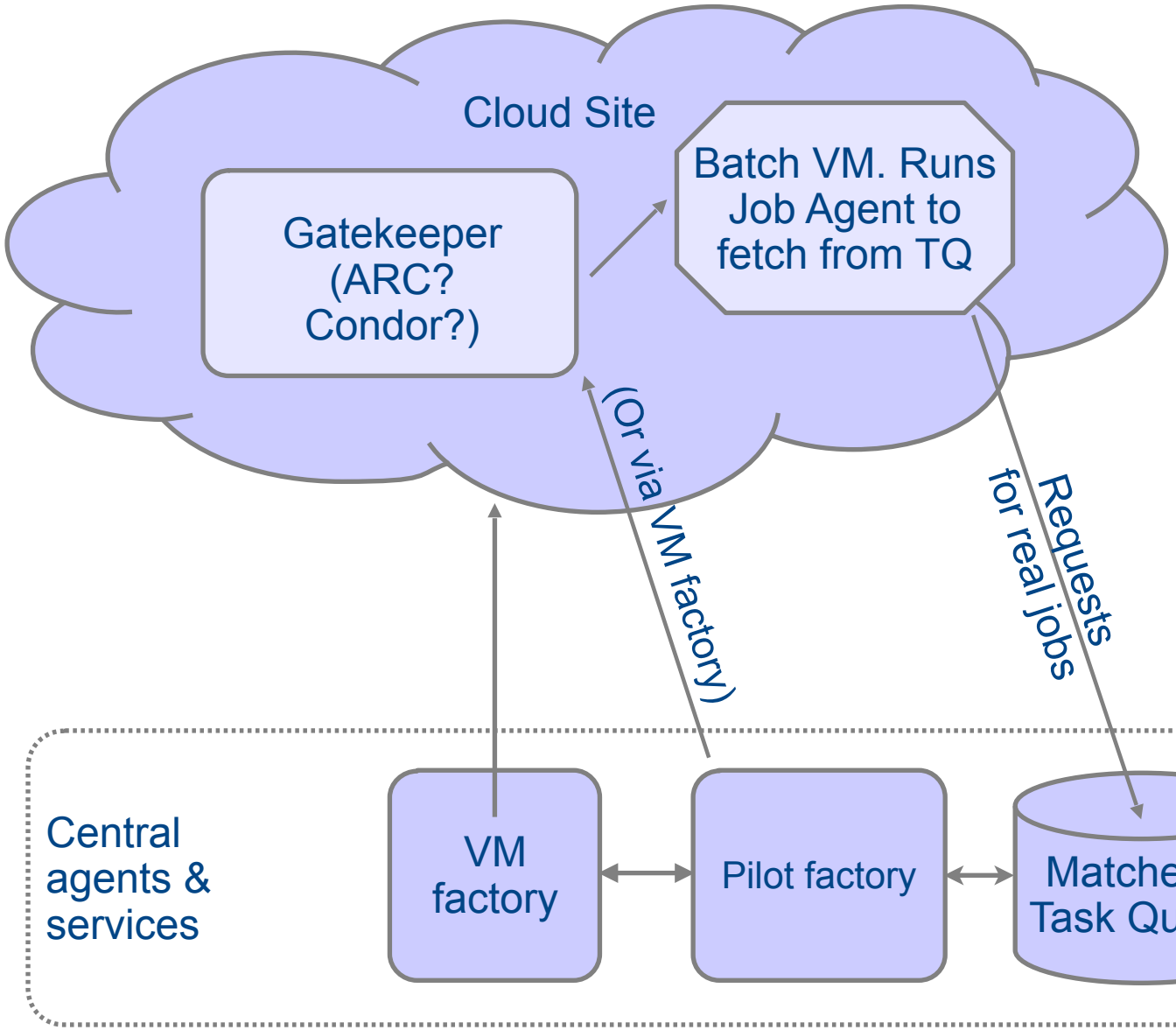
Motivated by similar goals but still the same pattern of operation

Further optimisations proposed, e.g. HTCondor-CE T4:519:Tue (Condor everywhere)

Virtual grid sites

- Cloud systems like OpenStack allow virtualization of fabric
- Bringing up a machine can be delegated towards user communities
  - Rather than using PXE+Kickstart+Puppet on bare metal
- Zeroth order virtualization is just to provide a Grid with Pilot Jobs site, on VMs
  - Potential to use staff more efficiently: one big pool of hardware and hypervisors managed all together rather than separate clusters (eg at CERN, T7:80:Mon)
- However, can take this a step further and have the experiment managing the cloud resources as a virtual grid site
  - ATLAS using CloudScheduler (T7:131:Tue)
  - ALICE using elastiq (Poster 460, session B)

Virtual “Grid with Pilot Jobs” site



Experiment creates its own “conventional” grid site on the cloud resources.

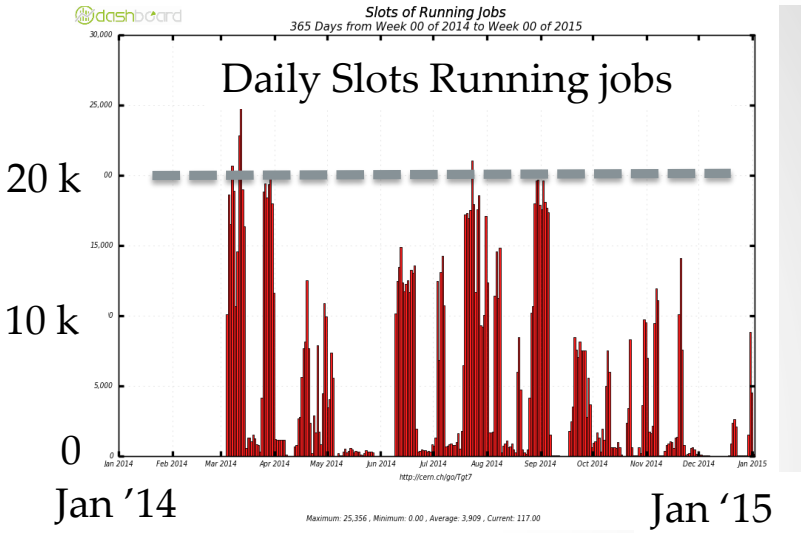
Transparent to existing central services, and user/production job submitters.

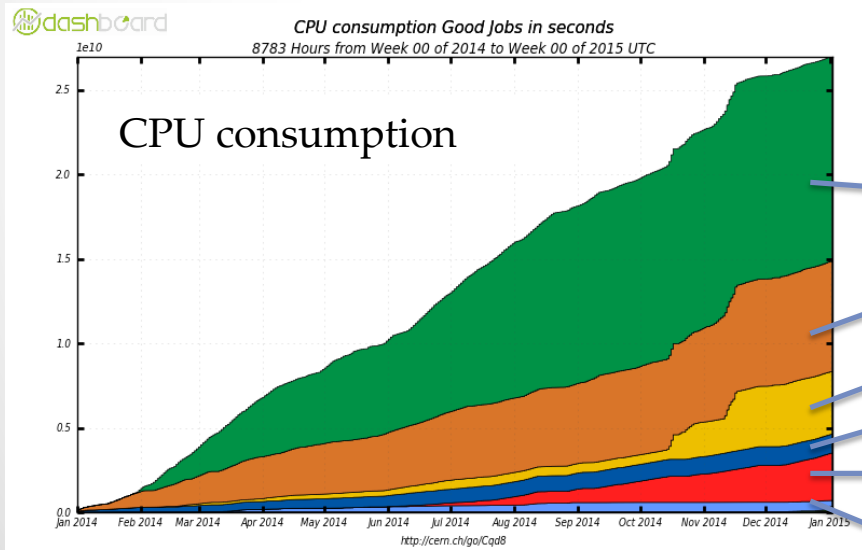
CloudScheduler (ATLAS) and elastiq (ALICE) are implementations of this model.

(Cloud Scheduler jobs, from Doug Benjamin's talk, CernVM workshop, March 2015)

ATLAS Cloud 2014

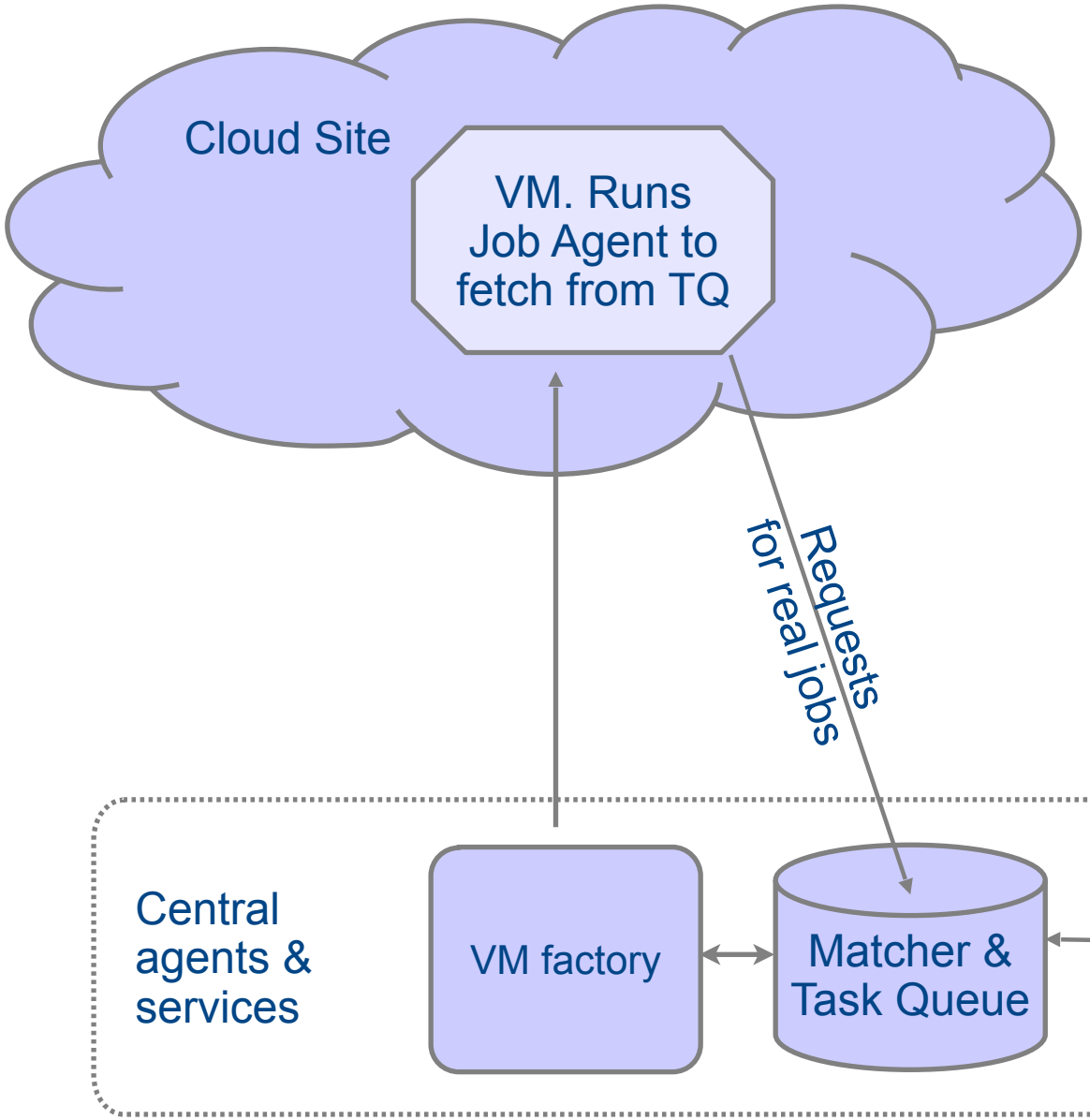
CERN – HLT farm Point 1 – Sim@P1
For Run 2 - run when LHC off for 24 hrs – under complete control of Online group
1.5 hrs to saturate the farm with VM instantiation
Running 10 GB cvmfs cache w/o issues





CERN PROD – 394 CPU yrs (CernVM)
IAAS – 202 CPU years (CernVM)
BNL/Amazon – 118 CPU years
NECTAR – 35 CPU years
ATLAS@HOME – 89 CPU yrs (CernVM)
GRIDPP – 17 CPU yrs (CernVM)

Experiment creates VMs directly?



Experiment creates VMs instead of pilot jobs.

Job Agent or pilot client runs as normal inside.

CMS glidein WMS works this way for cloud sites: looks at demand and creates VMs to join Condor pool (T7:230:Tue)

Further simplification: Vacuum model

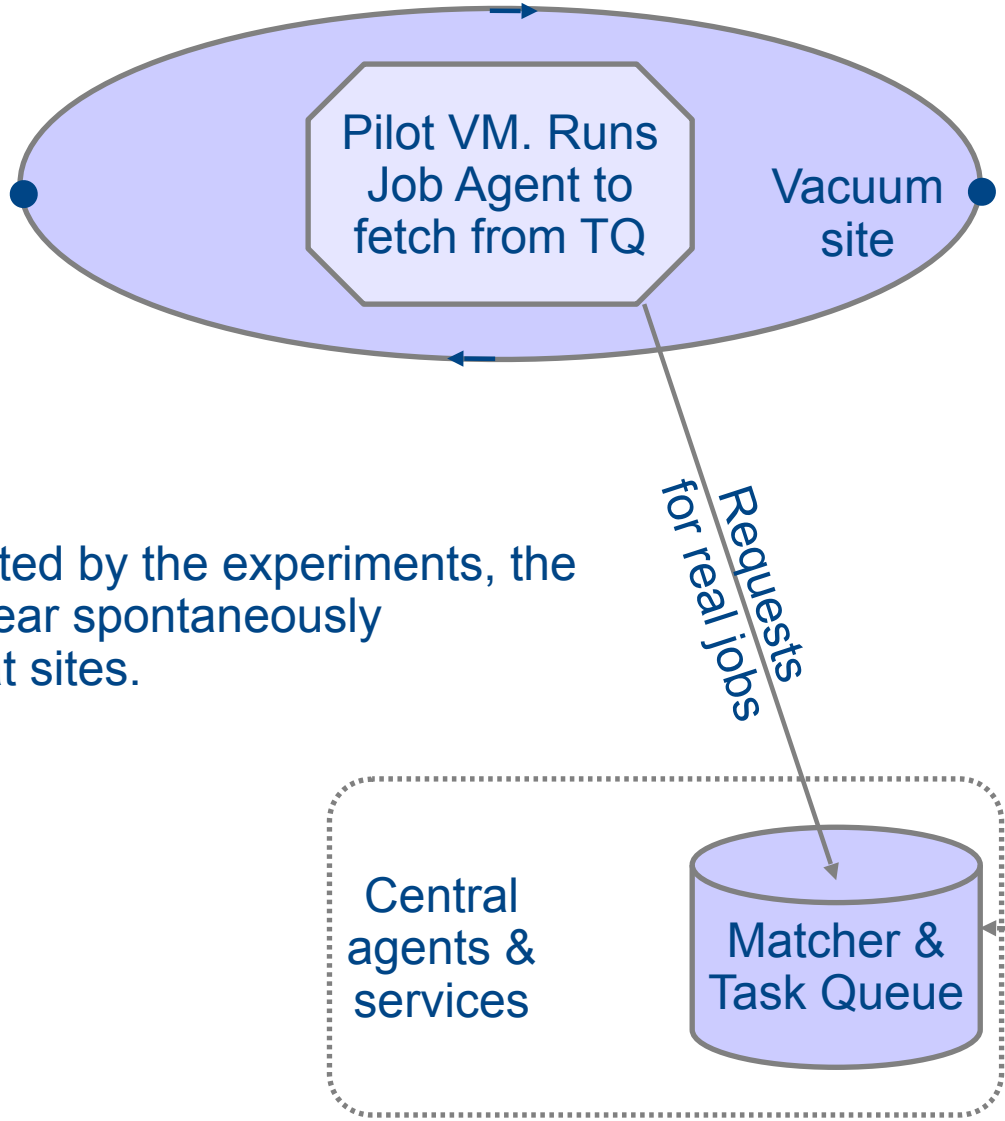
- Following the CHEP 2013 paper:
  - *"The Vacuum model can be defined as a scenario in which virtual machines are created and contextualized for experiments by the resource provider itself. The contextualization procedures are supplied in advance by the experiments and launch clients within the virtual machines to obtain work from the experiments' central queue of tasks."*
    ("Running jobs in the vacuum", A McNab et al 2014 J. Phys.: Conf. Ser. 513 032065)
  - a loosely coupled, late binding approach in the spirit of pilot frameworks
- For the experiments, VMs appear by "spontaneous production in the vacuum"
  - Like virtual particles in the physical vacuum: they appear, potentially interact, and then disappear
- CernVM-FS and pilot frameworks mean a small user_data file and a small CernVM image is all the site needs to create a VM
  - Experiments can provide a template to create the site-specific user_data

Vac, Vcycle, HTCondor Vacuum

- Three VM Lifecycle Managers that implement the Vacuum model
- Vac is a standalone daemon run on each worker node machine to create its VMs
  - At Manchester, Oxford, Lancaster, Birmingham
- Vcycle manages VMs on IaaS Clouds like OpenStack
  - Run by the site, by the experiment, or by regional groups like GridPP
  - Resources at CERN (LHCb), Imperial (ATLAS, CMS, LHCb), IN2P3(LHCb)
  - Vcycle instances running at CERN, Manchester, Lancaster
  - Vac/Vcycle talk T7:271:Mon
- HTCondor Vacuum manages VMs on HTCondor batch systems
  - Injects jobs which create VMs; VM jobs can coexist with normal jobs
  - Running at STFC RAL. See T7:450:Mon
- All make very similar assumptions about how the VMs behave
  - The same ATLAS, CMS, LHCb, GridPP DIRAC VMs working in production with all three managers

Vac - the first Vacuum system

Infrastructure-as-a-Client (IaaC)



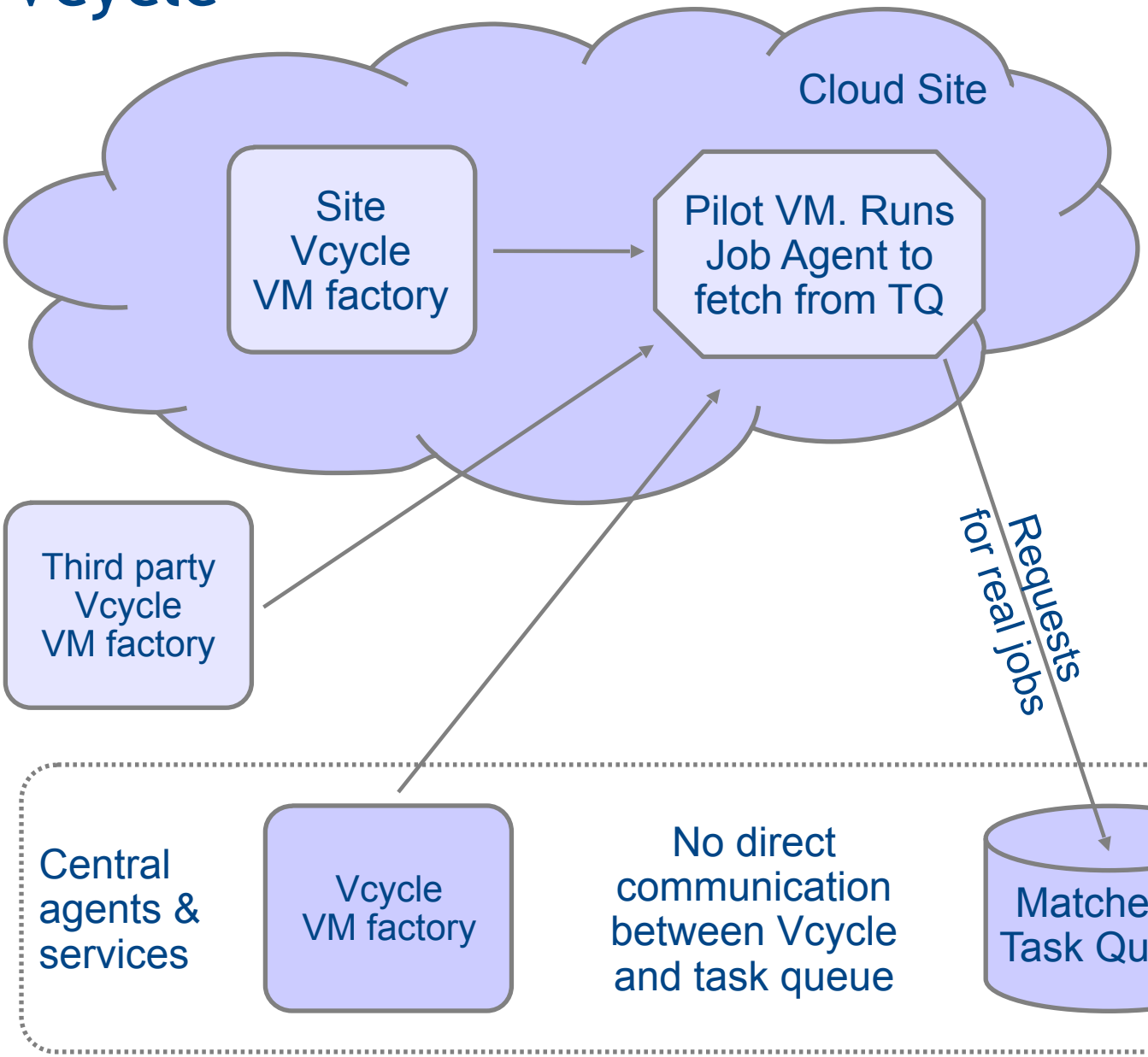
Since we have the pilot framework, we could do something really simple

Strip the system right down and have each physical host at the site create the VMs itself.

Instead of being created by the experiments, the virtual machines appear spontaneously "out of the vacuum" at sites.

Use same VMs as with IaaS clouds

Vcycle



Apply Vac ideas to IaaS.

Vcycle implements an external VM factory that manages VMs.

Can be run centrally by experiment or by site itself or by a third party

VMs started and monitored by Vcycle, but not managed in detail ("black boxes")

Pilot VMs

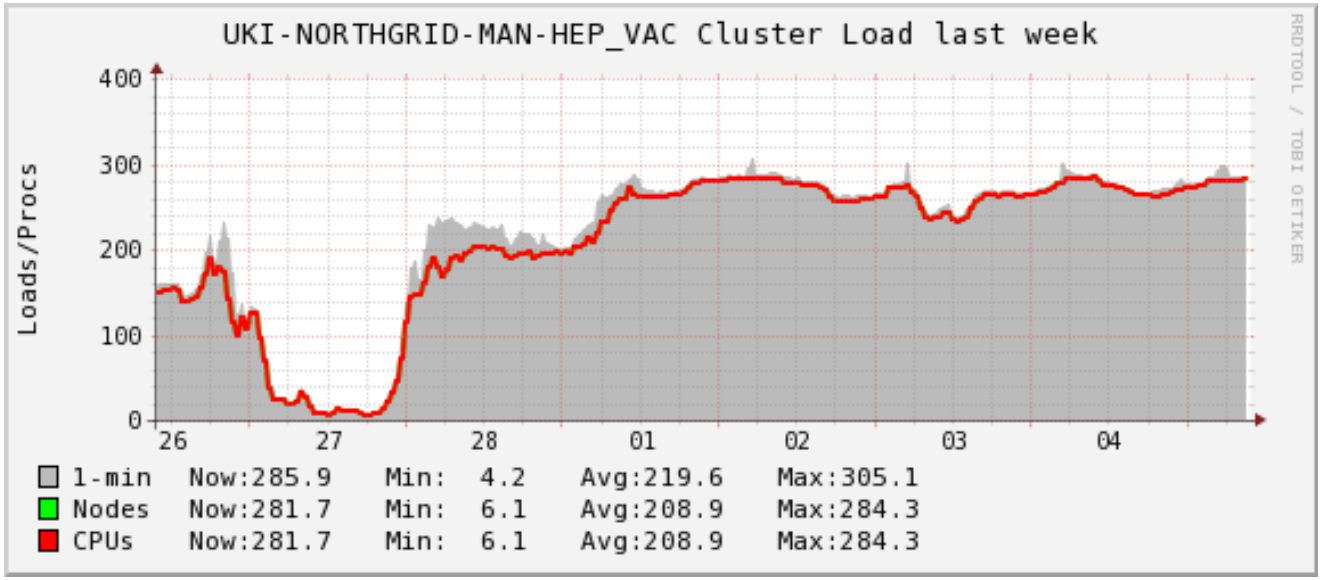
- Vac, Vcycle, HTCondor Vacuum assume the VMs have a defined lifecycle
- Need a boot image and user_data file with contextualisation
  - Provided by experiment centrally from an HTTPS web server
- Virtual disks and boot media defined and VM started
- machinefeatures and jobfeatures directories may be used by the VM to get wall time limits, number of CPUs etc
- The VM runs and its state is monitored
- VM executes shutdown -h when finished or if no more work available
  - Maybe also update a heartbeat file and so stalled or overruning VMs are killed
- Log files to /etc/machineoutputs which are saved (somehow)
  - shutdown_message file can be used to say why the VM shut down
- Experiments' VMs are a lot simpler for the site to handle than WNs
- ATLAS, CMS, GridPP DIRAC very similar to original LHCb VMs (T7:269:Tue)

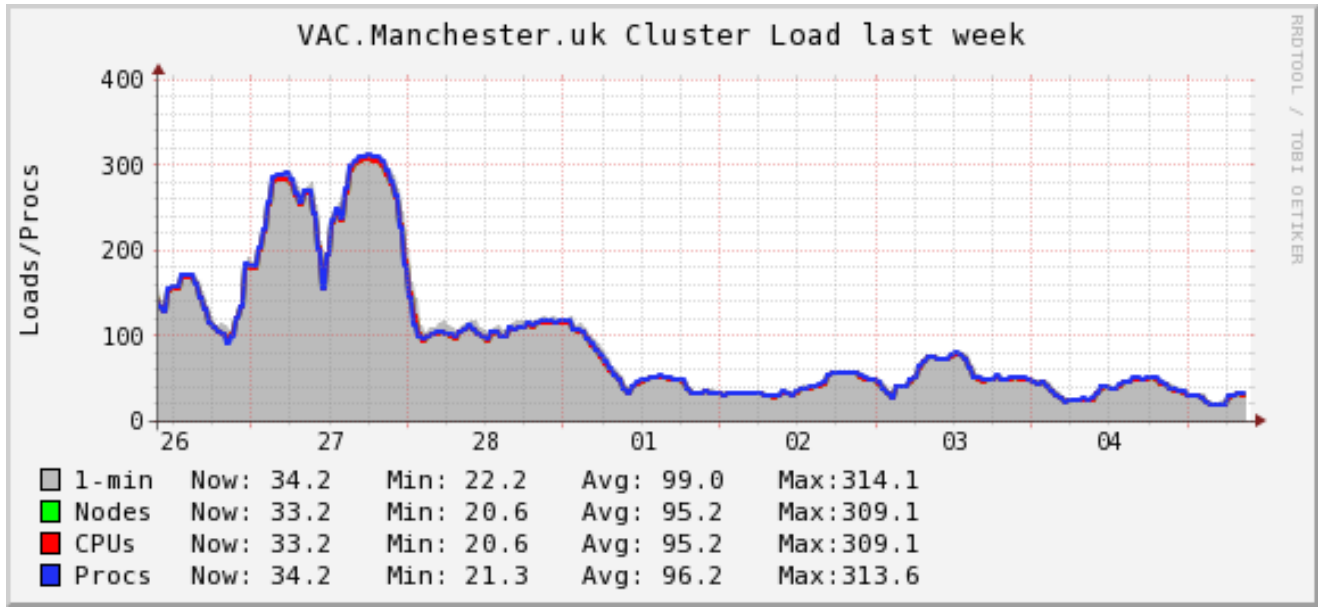
Example of Vac configuration

- Section of vac.conf used to enable LHCb VMs at Manchester
- They just need this and to create a hostcert/key.pem
- (vcycle.conf configuration is very similar)
- Compare what YAIM has to do to add a VO to a CE/Batch site

```
[vmtype lhcbprod]
vm_model = cernvm3
root_image = https://lhcbproject.web.cern.ch/lhcbproject/Operations/VM/cernvm3.iso
rootpublickey = /root/.ssh/id_rsa.pub
backoff_seconds = 600
fizzle_seconds = 600
max_wallclock_seconds = 172800
log_machineoutputs = True
accounting_fqan=/lhcb/Role=NULL/Capability=NULL
heartbeat_file = vm-heartbeat
heartbeat_seconds = 600
user_data = https://lhcbproject.web.cern.ch/lhcbproject/Operations/VM/user_data
user_data_option_dirac_site = VAC.Manchester.uk
user_data_option_cvmfs_proxy = http://squid-cache.tier2.hep.manchester.ac.uk:3128
user_data_file_hostcert = hostcert.pem
user_data_file_hostkey = hostkey.pem
```

Target shares: ATLAS vs LHCb





Each autonomous Vac machine uses VacQuery UDP protocol to discover what else is happening at the site.

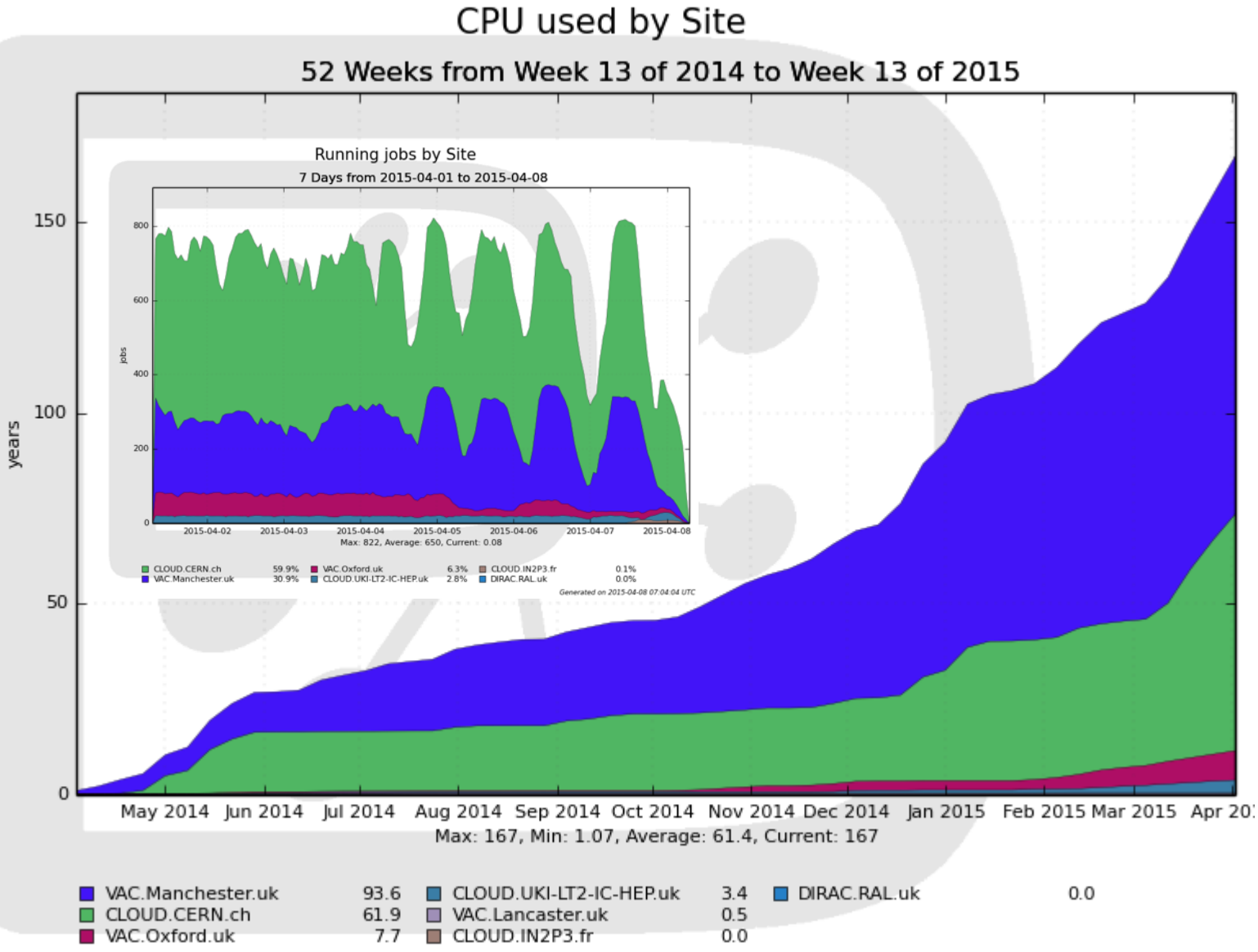
Compares this against target share for each type of VM (~1 per experiment).

Creates new VMs for experiments currently under their share.

But backs-off creating types of VM which are failing to find any work to do.

Vcycle uses similar target shares approach.

LHCb jobs in VMs



Routine production since May last year.

Increased capacity this year.

Periodic structure due to varying demand from other experiments and in availability of LHCb jobs

Some implications of virtualization

- Decouples operating system versioning requirements of experiments and sites
  - Jobs see an identical environment across sites
- Simplifies many aspects of security model (escalation much harder)
- Many opportunities for other simplifications (eg Vacuum)
- For the sites, VMs make WLCG systems seem a lot more “normal”
  - Especially if our “unusual” middleware is only inside the VMs
- When the Grid started, several sites had problems (or in fact failed) trying to run shared clusters with local IT services
  - eg shared PBS cluster for HEP and other sciences, with same WN configuration for everyone
- Virtualization gives us a second chance to try this where appropriate

BOINC and @home projects

- BOINC grew out of SETI@home, and now includes virtualization
  - A vacuum implementation before the term was invented
- Volunteers install BOINC on their private machines and run work for projects they want to support
- LHC experiments have prototype or production @home projects
- Potentially a **huge** source of *unreliable* CPU
  - HEP has a high profile in the media, and some fraction of that can be turned into contributions via BOINC
  - A new type of environment for us
  - Users have complete control over their machines so a risk of malicious and/or naive interference in the code
- Can also contribute to outreach
- See Laurence's talk yesterday and T7:82:Tue, T7:170:Tue, T7:104:Tue

Containers

- Linux containers can provide an alternative (often a drop-in alternative) to most of the places I've talked about VMs
- Can use containers to provide a protected virtual environment of libraries, files etc to payload jobs
  - "Container as VM"
- Other more lightweight models are possible, more focussed on the environment seen by particular processes or sets of processes
  - Compare Docker apps vs RPM installs
- See T7:373:Thu for CernVM roadmap and containers integration
- See Containers vs VMs talk T7:356:Mon

High Performance Computing resources

- Traditionally not something we've needed due to our parallel data
  - Some applications in end user analysis (eg big iterative fits)
- As they typically allocate many CPUs across multiple hosts to jobs, they can be vulnerable to having unused CPUs in between
  - Our single CPU and 8-CPU jobs are ideal for packing these gaps with useful work
- LHC experiments are using this kind of resource now
  - Posters from ATLAS (153, 92)
- Need to work with the often minimal environments they provide
  - eg use Parrot to provide cvmfs resources via user level libraries

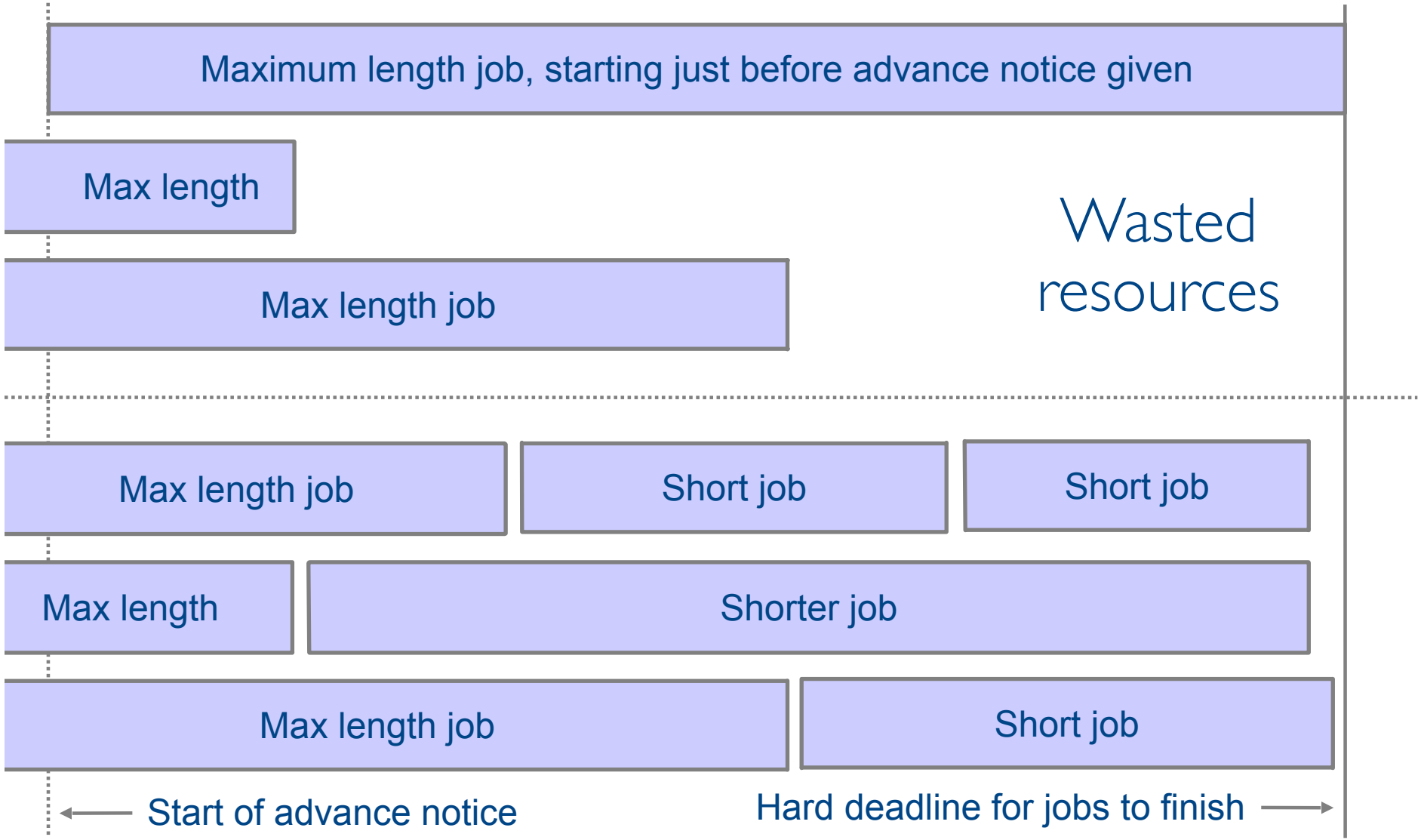
Summary

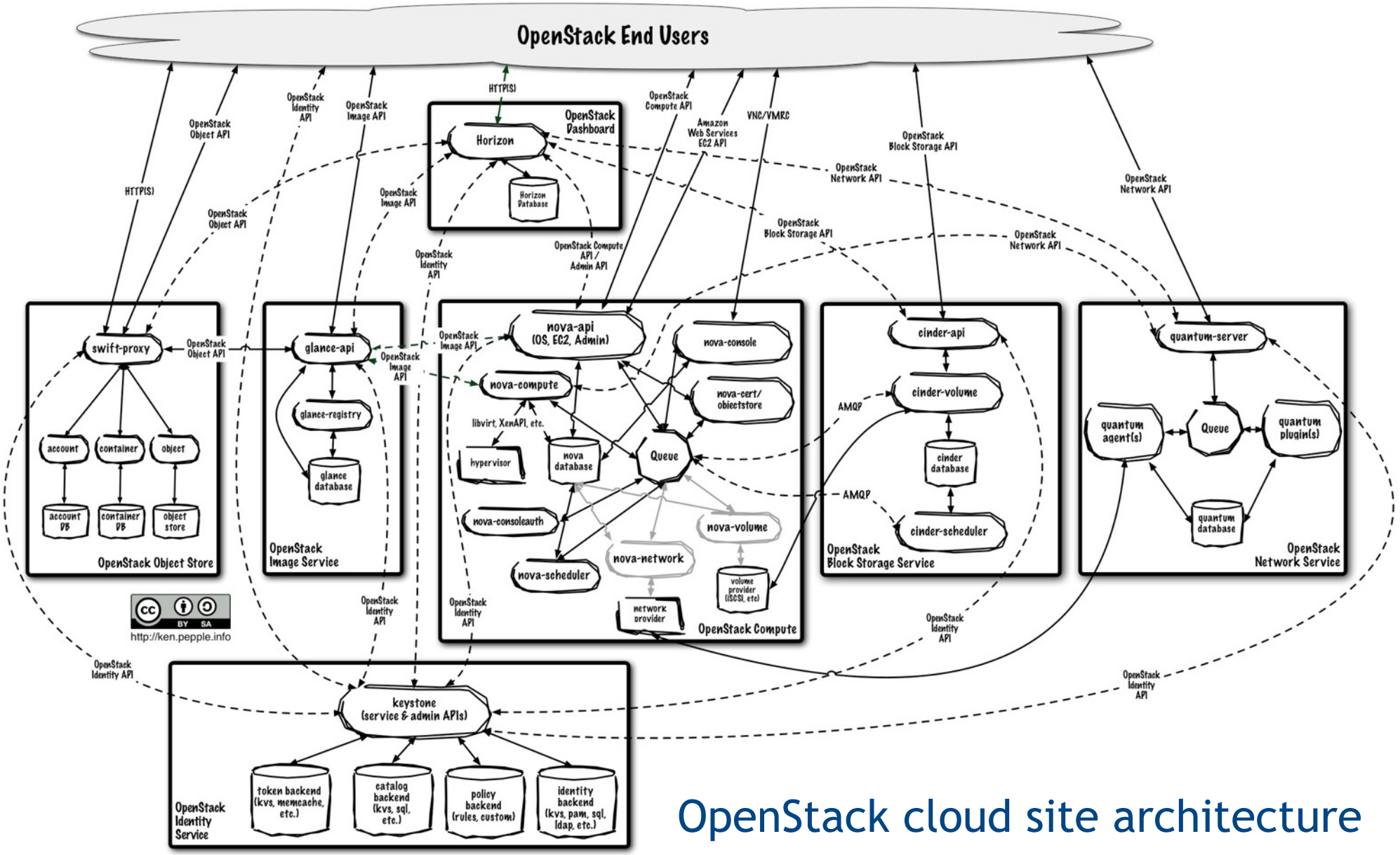
- “How can we do more with less?”
  - Simplify what we have
  - Use virtualized environments (including containers)
- Levels of how far to go down the virtual road
- Virtual grid site and Vacuum models are already running production jobs
  - CloudScheduler
  - elastiq
  - GlideinWMS
  - Vac
  - Vcycle
  - HTCondor Vacuum
- BOINC with VMs has a lot of potential to supply CPU
- Experiments already exploiting spare HPC resources opportunistically
- Many talks about these themes during CHEP!

Extra slides

The Masonry Problem...

The Masonry Problem





OpenStack cloud site architecture