Managing virtual machines with Vac and Vcycle

Andrew McNab
University of Manchester

Peter Love
Lancaster University

Ewan MacMahon
University of Oxford

Overview

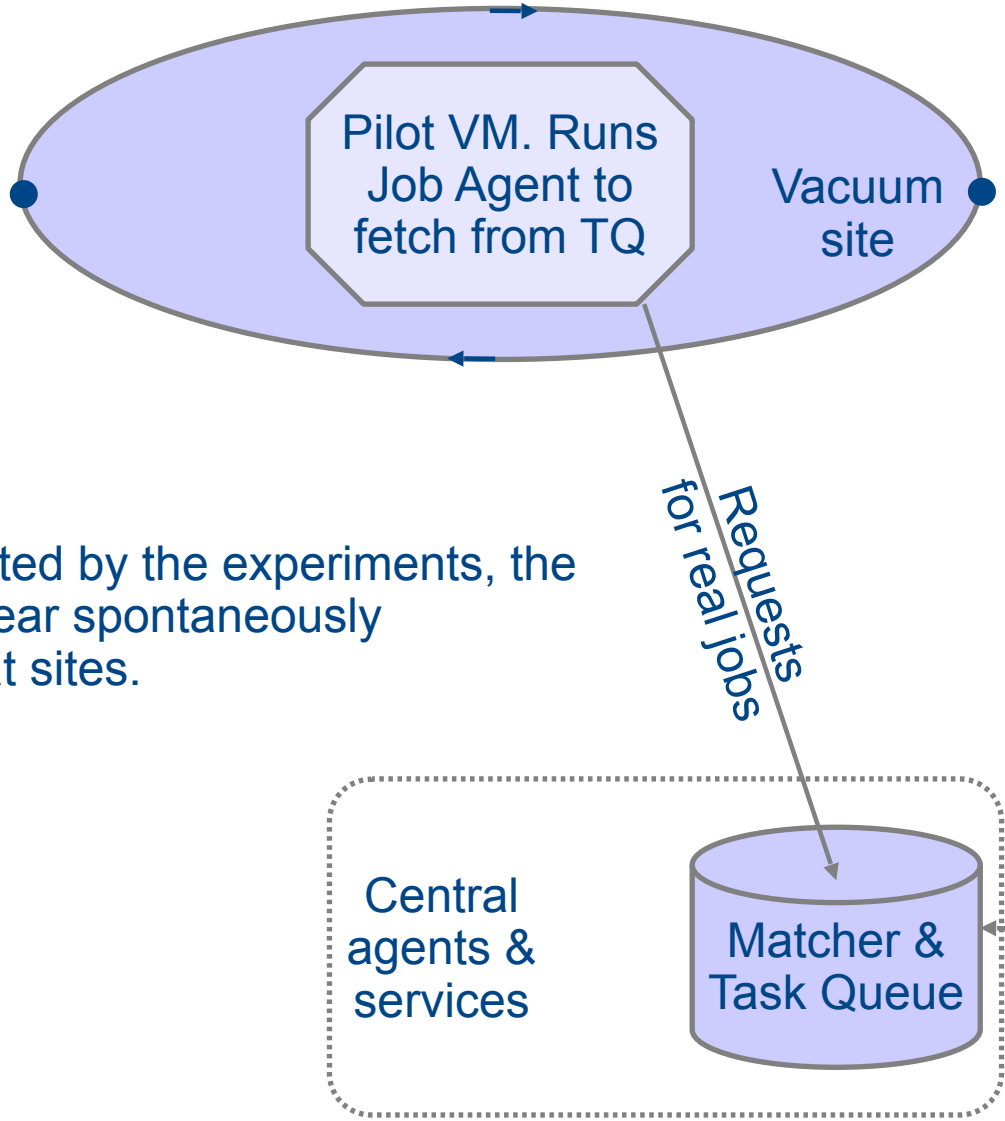
- Vac and Vcycle
- Status at sites
- user_data templates
- boot images
- Target shares
- Accounting
- Multiprocessor

Vac and Vcycle

- Both VM Lifecycle Managers for VMs running jobs
- Both implement Vacuum Model, defined in 2013
  - *("Running jobs in the vacuum", A McNab et al 2014 J. Phys.: Conf. Ser. 513 032065)*
- Vac is a standalone daemon on each worker node machine to create VMs
- Vcycle manages VMs on IaaS Clouds like OpenStack
  - Can be run at the site, by the experiment, or by regional groups like GridPP
- Both developed at Manchester as part of GridPP Clouds/VMs effort
  - With help from Lancaster, Oxford, IC, CERN, Birmingham, LHCb and ATLAS
- Both make very similar assumptions about how the VMs behave
  - The same LHCb, ATLAS, CMS VMs working in production on Vac and Vcycle
- See tomorrow's LHCb VMs talk for "Pilot VM" details
- Vac/Vcycle not tied to CernVM but uses it everywhere in practice
- Vacuum model also implemented by HTCondor Vacuum - see Andrew Lahiff's talk later today

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.

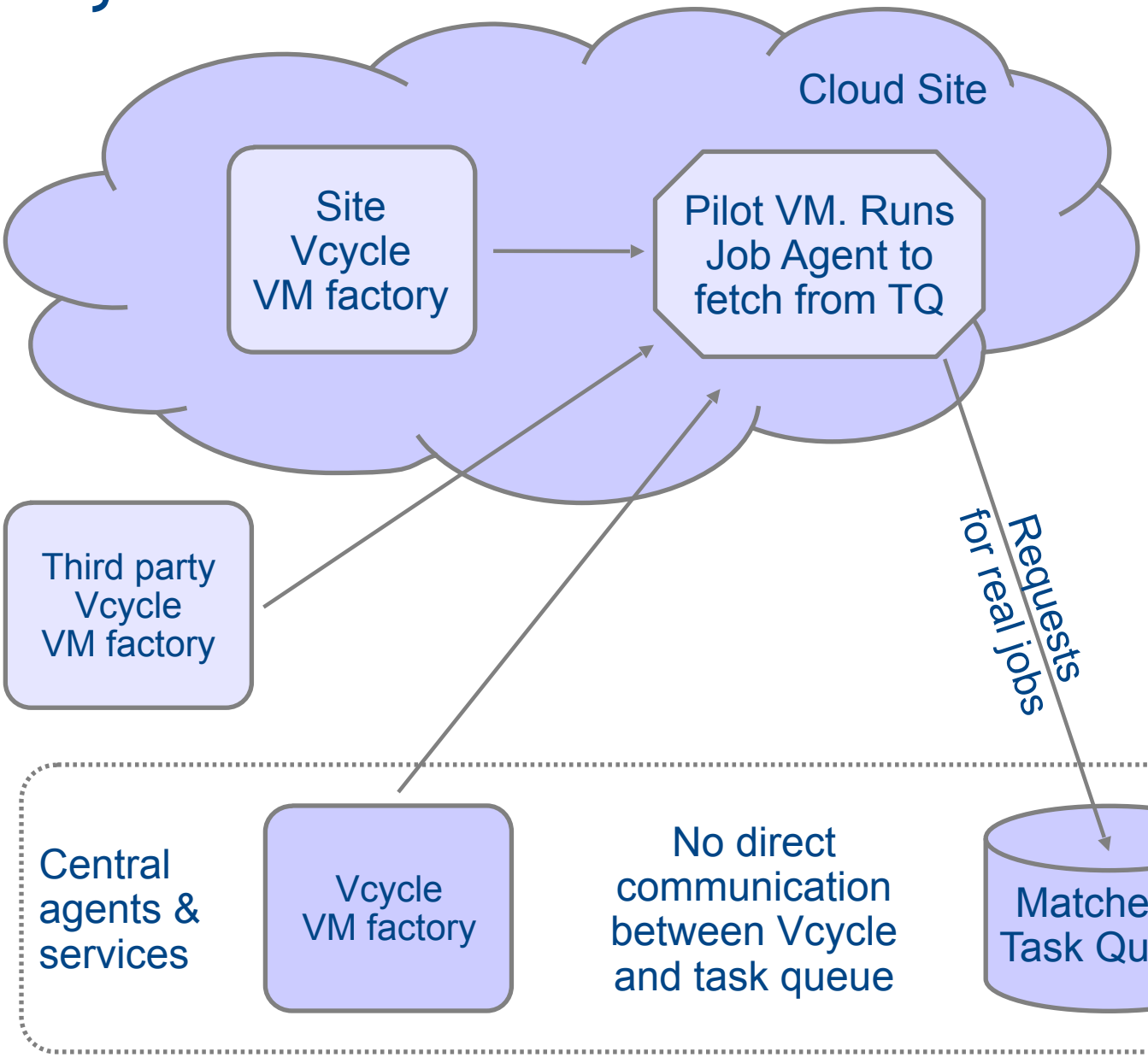
Instead of being created by the experiments, the virtual machines appear spontaneously "out of the vacuum" at sites.

Use same VMs as with IaaS clouds

Vac implementation

- On each physical node, Vac VM factory daemon runs to create and supply contextualization user_data to transient VMs
- Multiple VM flavours ("VM types") are supported, ~1 per experiment
- Each site or Vac "space" is composed of autonomous factory nodes
  - All using same /etc/vac.d/*.conf files; managed by Puppet, Chef, Cfengine, ...
- Factories communicate load info with each other via VacQuery UDP
- Backs off creating types of VM that are failing to find work
- Provides a logical partition to the VM to use as fast workspace
- VMs on a NAT network, with the factory node at 169.254.169.254
- Vac assumes the VM will shut itself down if it has no work
  - Vac can also check a heartbeat file and destroy stalled/idle VMs
  - (Could also kill stalled/idle VMs based on CPU usage)

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")

Vcycle implementation

- Apply Vac ideas to OpenStack etc IaaS resources
- Experiment-neutral, and can be run by experiment or site or 3rd party
- Vcycle daemon creates VMs using user_data template
- Watches what they do
- Backs off if they are failing to stay running
  - No work? Fatal errors?
  - Can also use shutdown messages to make better decisions
- Provides machine/job features via HTTPS
  - Also used to collect log files, shutdown messages, and heartbeat updates
- Currently uses OpenStack REST API directly
  - Luis Villazon Esteban has finalised an OCCI plugin using OCCI REST API
  - We're starting on an EC2 plugin

Deployment by site and experiment

		ATLAS	CMS	LHCb	GridPP DIRAC
Vac	Manchester	✔	✔	✔	✔
	Oxford	✔	✔	✔	✔
	Lancaster	✔		✔	✔
	Birmingham				✔
Vcycle	CERN (LHCb)			✔	
	CERN (Dev)	✔	✔	✔	✔
	Imperial	✔	✔	✔	✔
	CC-IN2P3			✔	
HTCondor Vacuum at RAL (STFC)		✔	✔	✔	✔

user_data templates and images

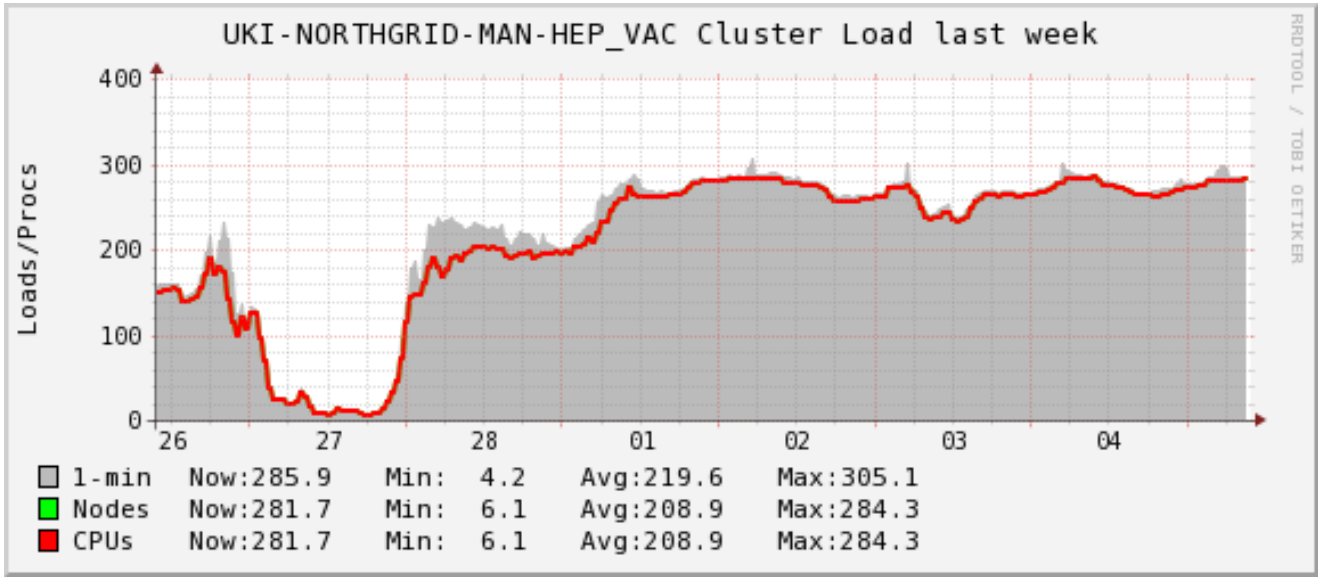
- Resource provider can create user_data file, or let Vac/Vcycle do it from a template.
- Templates can be given as URLs, on experiment's web server
  - Refetched each time since very small
- Experiment can also supply desired uCernVM 3 boot images
  - Since so small (20MB) again can be specified as URLs
  - Cached and rechecked each time with If-Modified-Since
  - Vcycle uploads new VM images to OpenStack automatically
- Experiments can update user_data and images everywhere without having to contact site admins
- Easy to add new experiment to a Vac or Vcycle installation

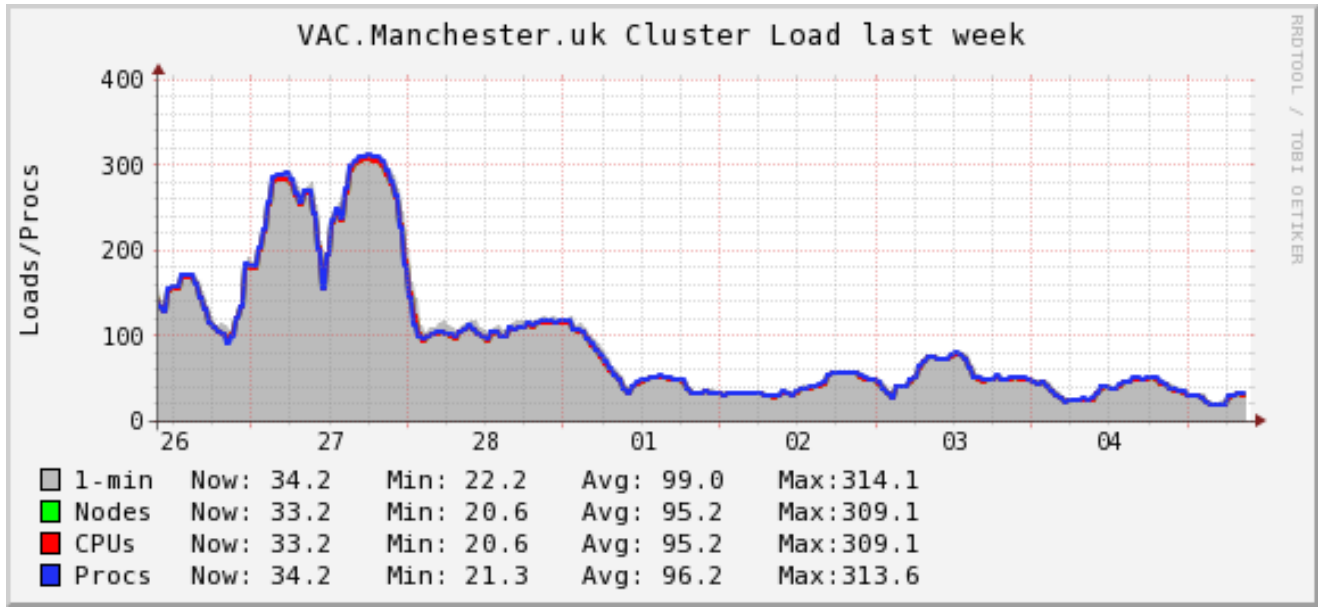
Example of configuration

- 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)
- root_image and user_data template fetched from experiment

```
[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.

Accounting and multiprocessor

- Vac and Vcycle write out APEL SSM records when each VM finishes
  - Can be sent to the central APEL service by the standard APEL ssmsend tool (as ARC does)
  - For Vcycle, this provides an alternative to installing accounting reporting tools for OpenStack etc at the site
  - For Vac, it makes the accounting at the site completely self-contained at the level of the individual physical machine
- Both Vac and Vcycle are ready for multiprocessor VMs
  - VM instantiation supports it natively
  - Vac/Vcycle do accounting and target shares in terms of HS06 so can handle a mix of VMs with different performance properly
  - machine/job features support allows sophisticated “elastic” strategies by VM architects to do internal payload job masonry based on VM lifetimes

Summary

- Vac provides a simple way for sites to run VMs
- Vcycle provides a simple way to manage VMs on OpenStack
- “Simplicity is a feature”

- Both demonstrated with ATLAS, CMS, LHCb production jobs
  - Hundreds of thousands of jobs; hundreds of CPU years
- Aim has been to demonstrate production quality rather than roll out to a large number of sites so far
- For Vac http://www.gridpp.ac.uk/vac/ for RPMs, Yum repo, links to GitHub, docs, man pages etc
- For Vcycle http://www.gridpp.ac.uk/vcycle/ has links to source, man pages etc

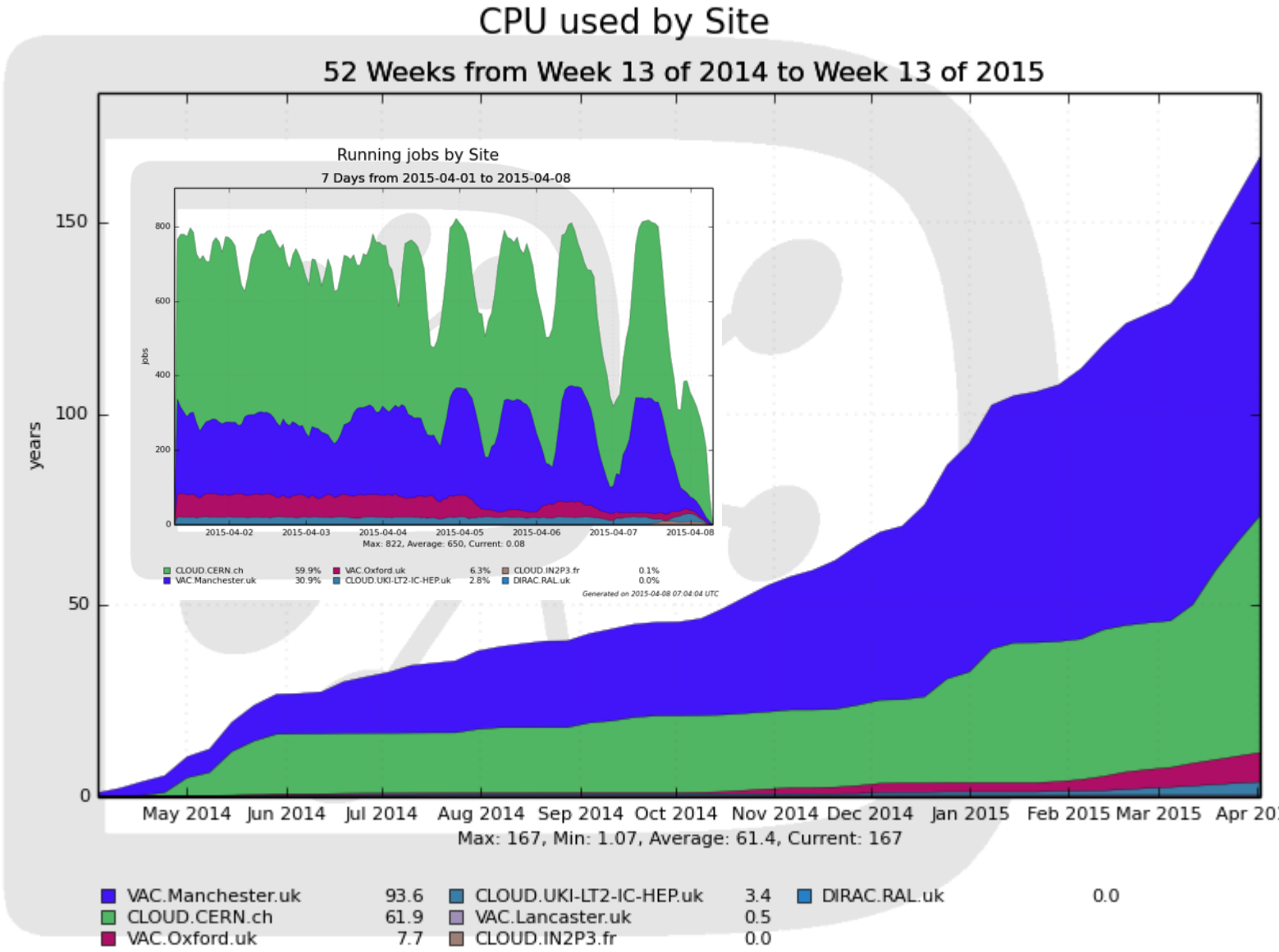
- Tomorrow’s talk on LHCb VMs in explains the Pilot VM internals

Extra slides

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

LHCb jobs in VMs



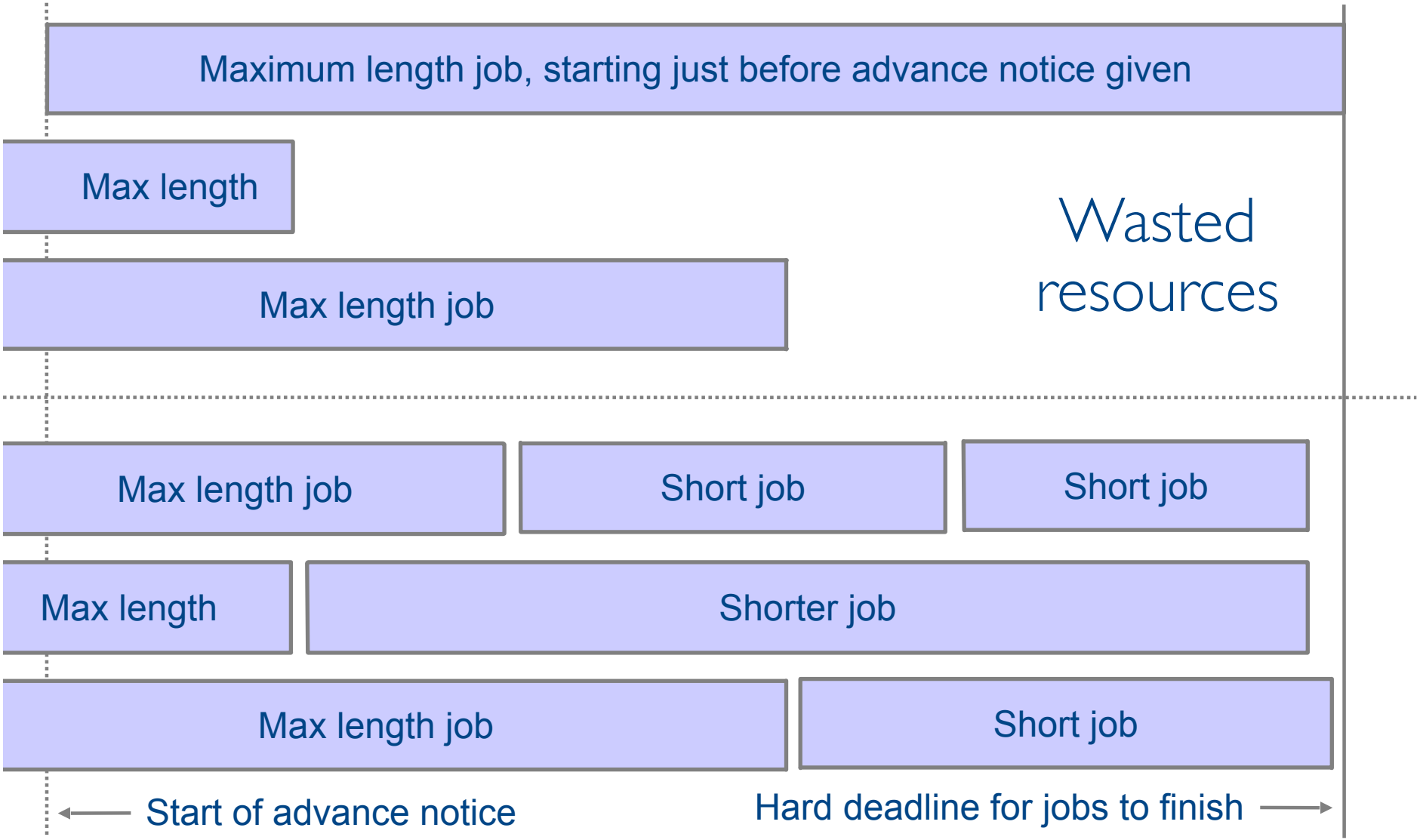
Routine production since May last year.

Increased capacity this year.

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

The Masonry Problem...

The Masonry Problem



Vac and Vcycle at sites

- Manchester
  - Vac running ATLAS, CMS, LHCb VMs
    - In last year: 300,000 jobs, 196 CPU years
  - Vcycle instance managing Imperial (LHCb/ATLAS) and CERN development VMs
- Lancaster
  - Long running Vac site
  - Currently moving to an increased number of machines using nodes retired from batch
- Oxford
  - Vac running ATLAS, CMS, LHCb VMs
- Imperial
  - GridPP OpenStack tenancy has ATLAS, CMS, LHCb VMs
- Birmingham
  - Developing procedure to run Vac on interactive cluster machines overnight
- CERN
  - LHCb tenancy managed by Vcycle on LHCb vobox at CERN
  - Development tenancy with ATLAS, LHCb, CMS managed by Vcycle at Manchester
- Vac and Vcycle total resources each comparable to a typical Tier-2

“Pilot VM” lifecycle

- Vac and Vcycle assume the VMs have a defined lifecycle
- Need a boot image and user_data file with contextualisation
  - Experiment provides procedure to make a site-wide user_data file
- 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
  - Largely due to internal reliance on CernVM-FS