Making it simpler to provide CPU at Tier-2

Andrew McNab
University of Manchester
LHCb

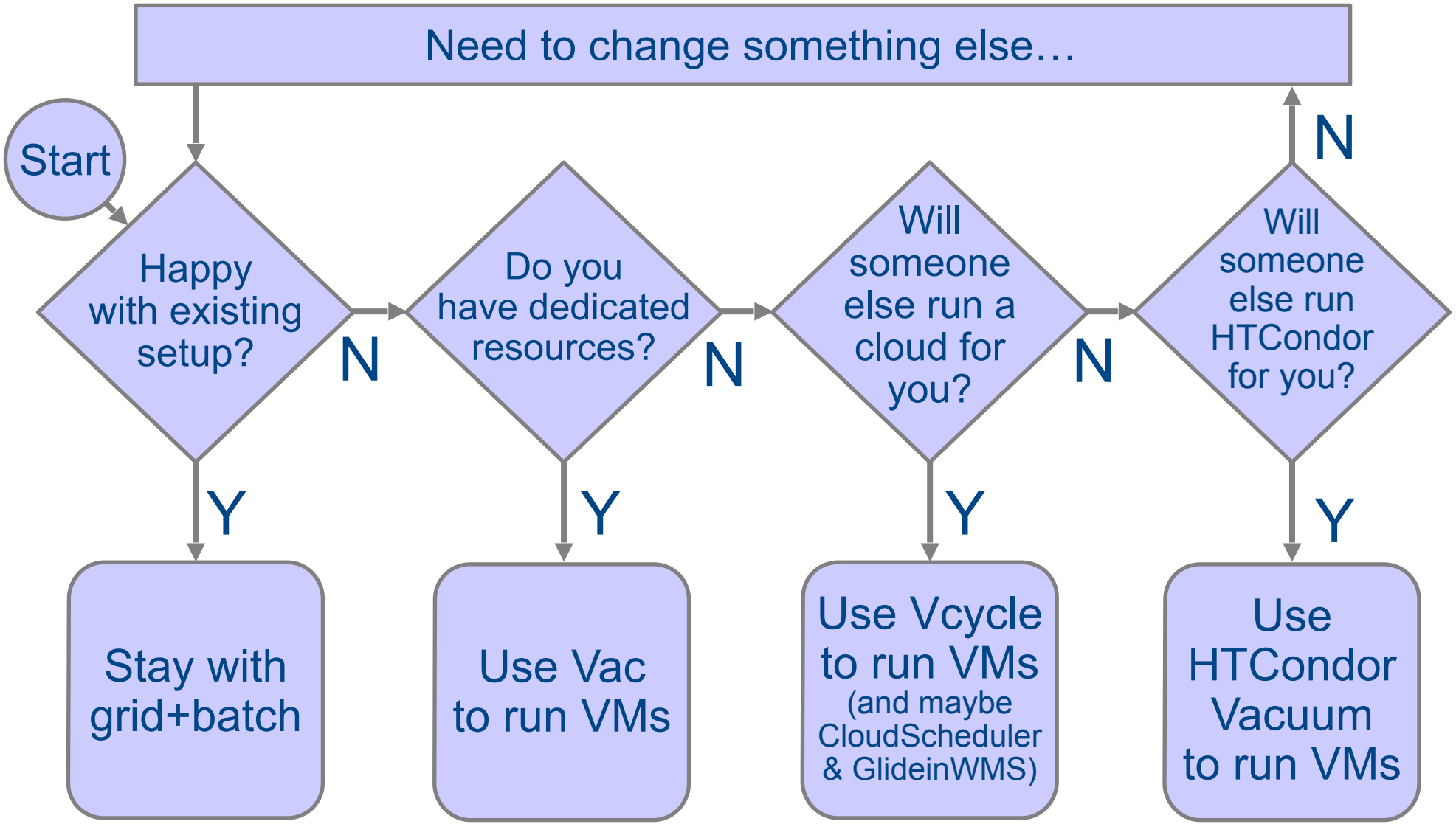
Overview

- Aimed at GridPP Tier-2 sites that want to run jobs for LHC experiments and smaller ones supported by GridPP
- Want to simplify not only what we run but what we actually depend on for our major experiments
- I'm going to give my personal recommendations
- I'm going to try to avoid hedging and equivocation
- But hopefully it's balanced and realistic

Virtualization

- The new development is the ability to virtualize the environments we run jobs in
- At the moment this is almost exclusively virtual machines based on CernVM
  - Containers now being actively worked by CernVM
  - So where I say “virtual machine” in these slides, I could say “logical machine” to be general
- VMs shift responsibility for the worker node environment to the experiments - immediate gain for sites
- VM management technologies can remove the need to run a traditional grid+batch site
- GridPP is very active in this in ATLAS, CMS, LHCb, and DIRAC so support is available
- For local users, can run HTCondor worker nodes in VMs

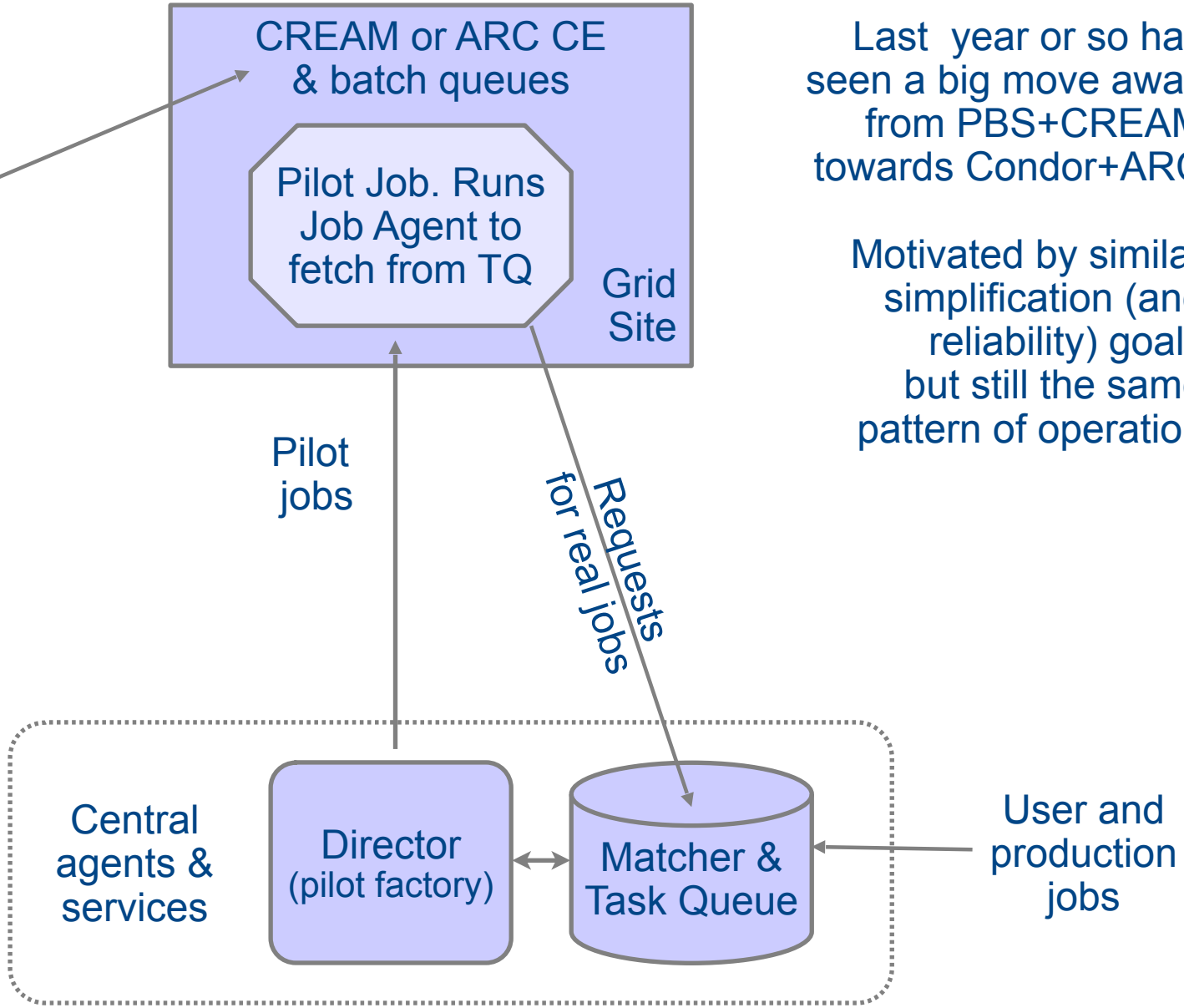
My personal simplification flowchart



Stay with grid+batch site?

This is complicated even if in this room we know how to do it.

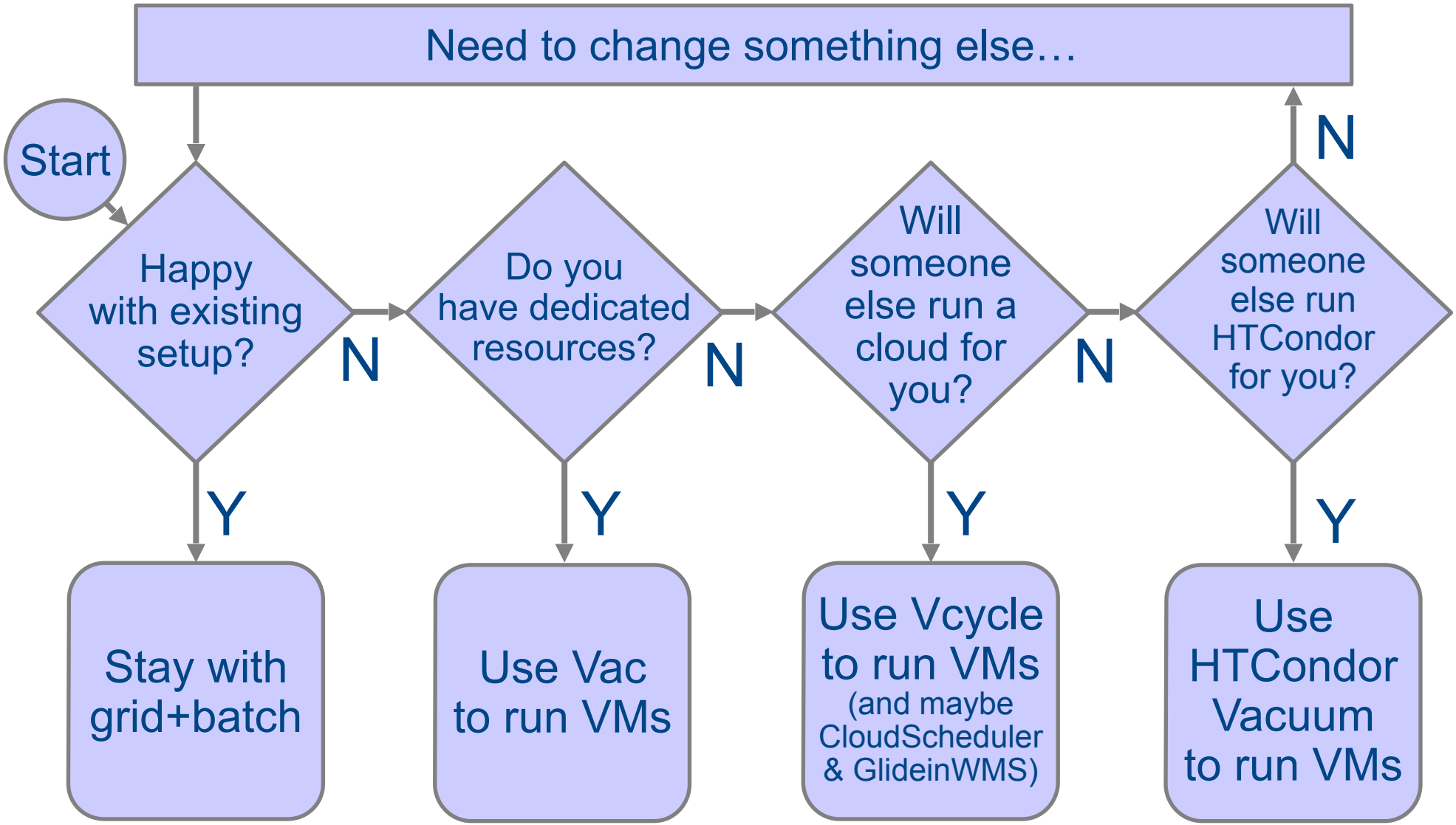
Difficult to get other people (eg non specialist sysadmins) to be able to cover for site experts.



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

Motivated by similar simplification (and reliability) goals but still the same pattern of operation

My personal simplification flowchart



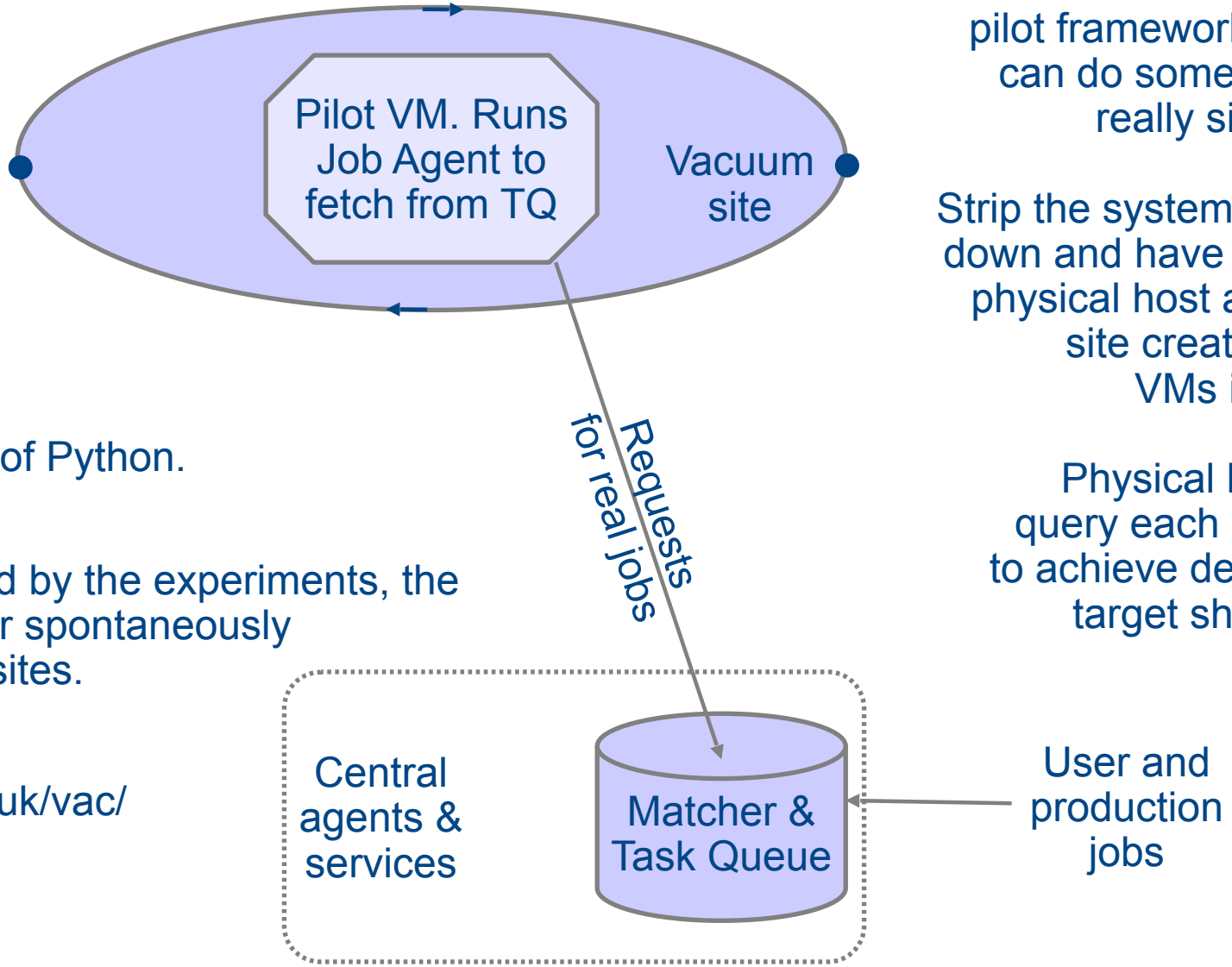
Vac

This is the simplest way to exploit the VMs offered by the experiments.

Codebase ~3000 lines of Python.

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

http://www.gridpp.ac.uk/vac/



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

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

Physical hosts query each other to achieve desired target shares.

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
- 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
```

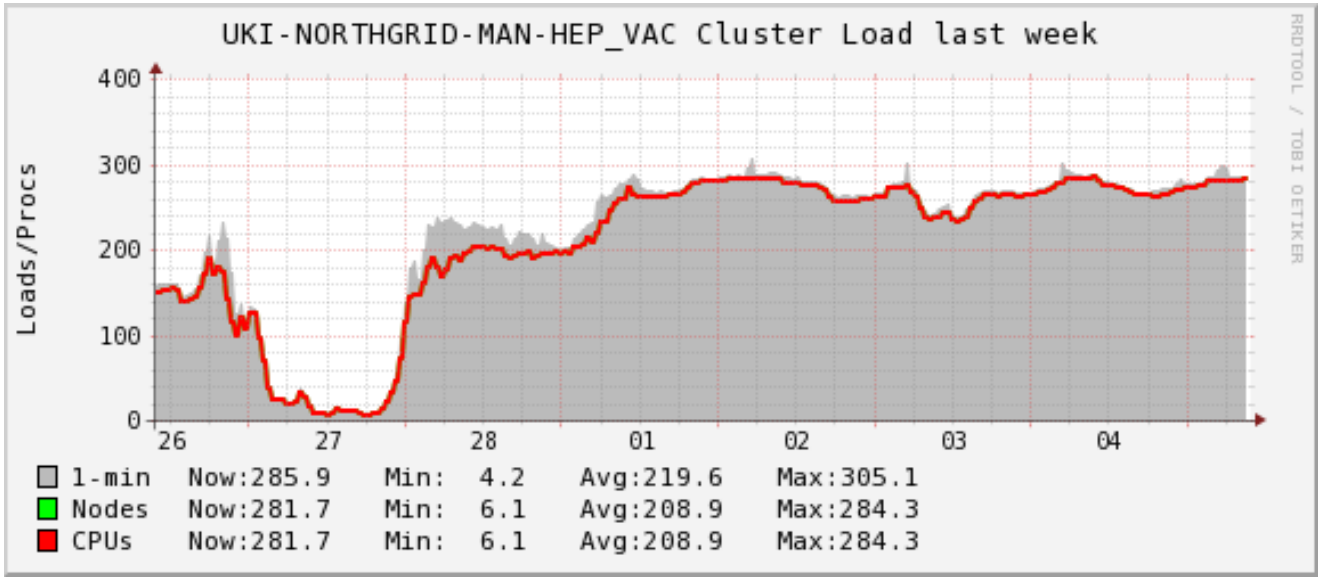
Target shares: ATLAS vs LHCb

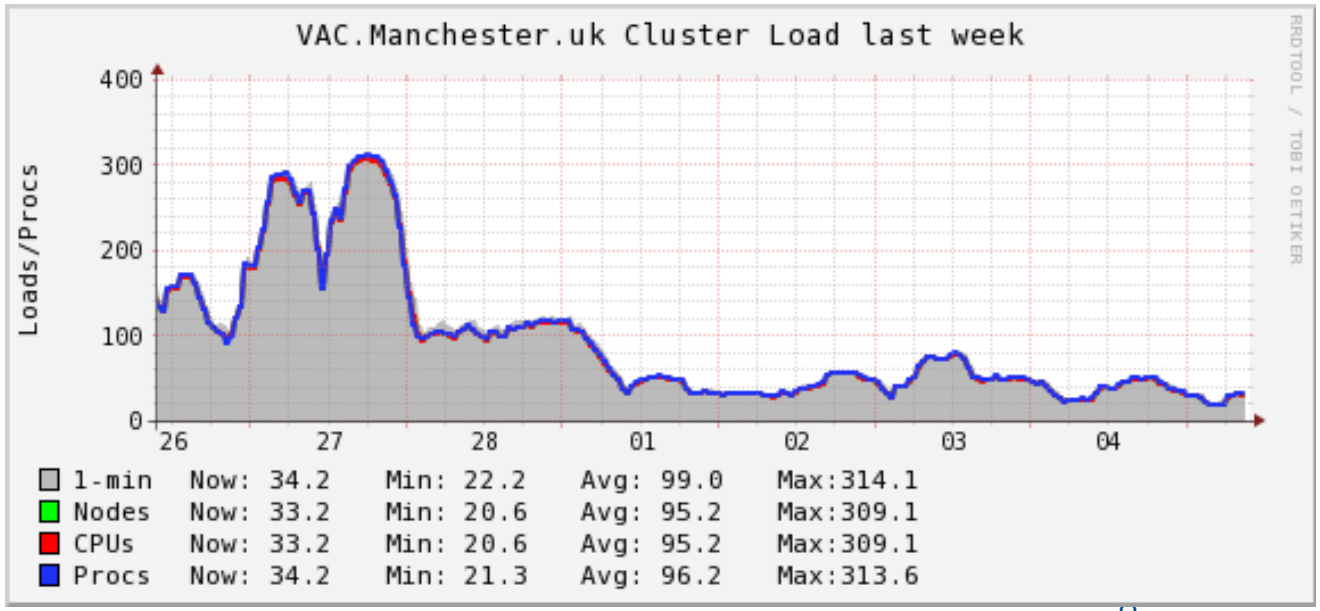
Each autonomous Vac machine uses VacQuery UDP protocol to discover what else is happening at the site.

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.

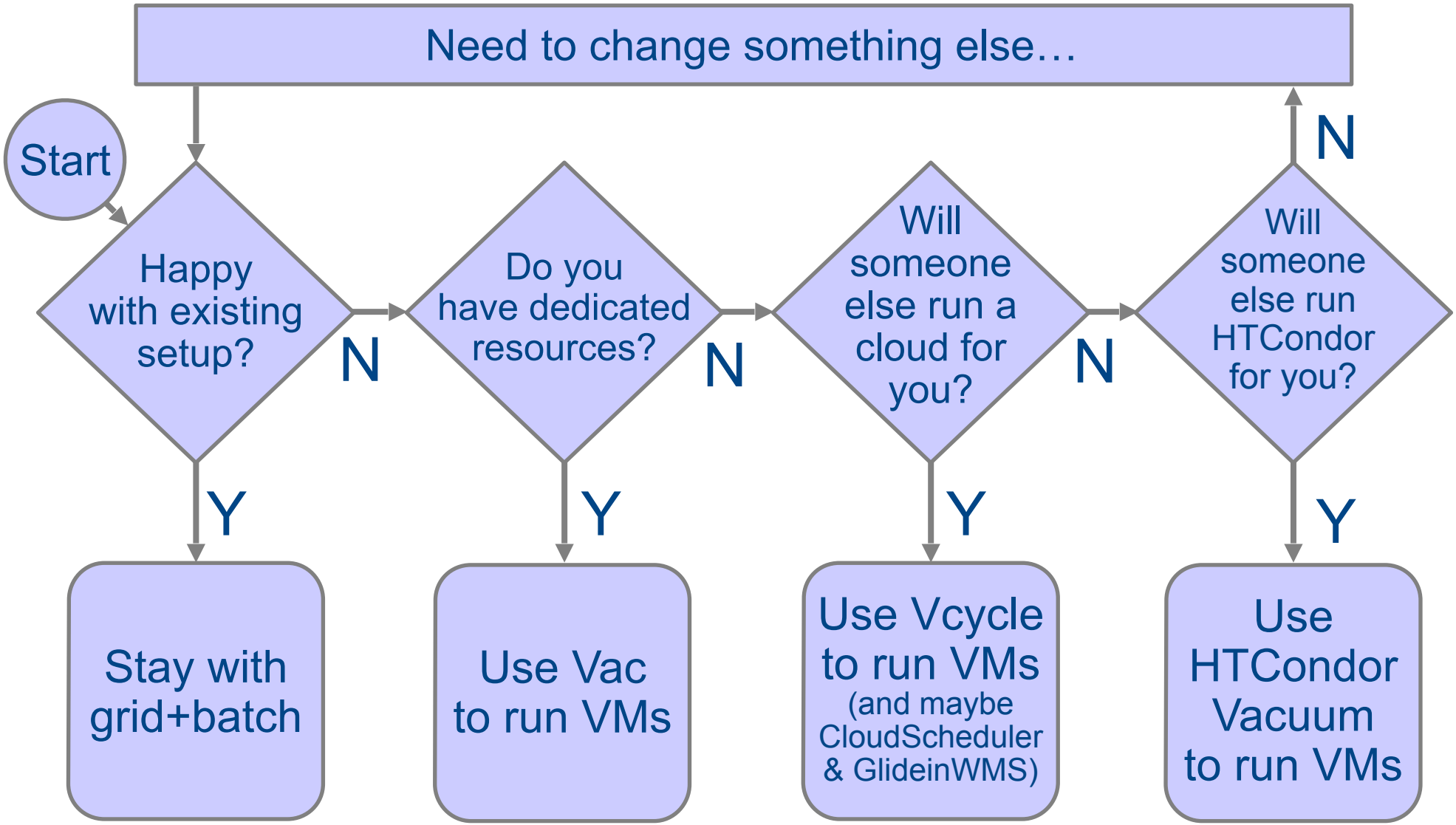




Vac-in-a-Box

- Vac comes with a Puppet module and was designed to be managed by conventional site admin systems
- Vac-in-a-Box (ViaB) is now being developed to make getting started even easier - prototype at UCL this week
- ViaB is a kickstart procedure for creating self-contained Vac machines
- Use a Squid cache on each Vac machine, communicating using Squid's ICP UDP protocol to discover local copies
  - Want to remove need for site Squid
- ViaB website to create kickstart files and configuration RPMs for sites using web page forms
- Customised files will also be generated to enable migration to managing config RPMs yourself or to using Puppet

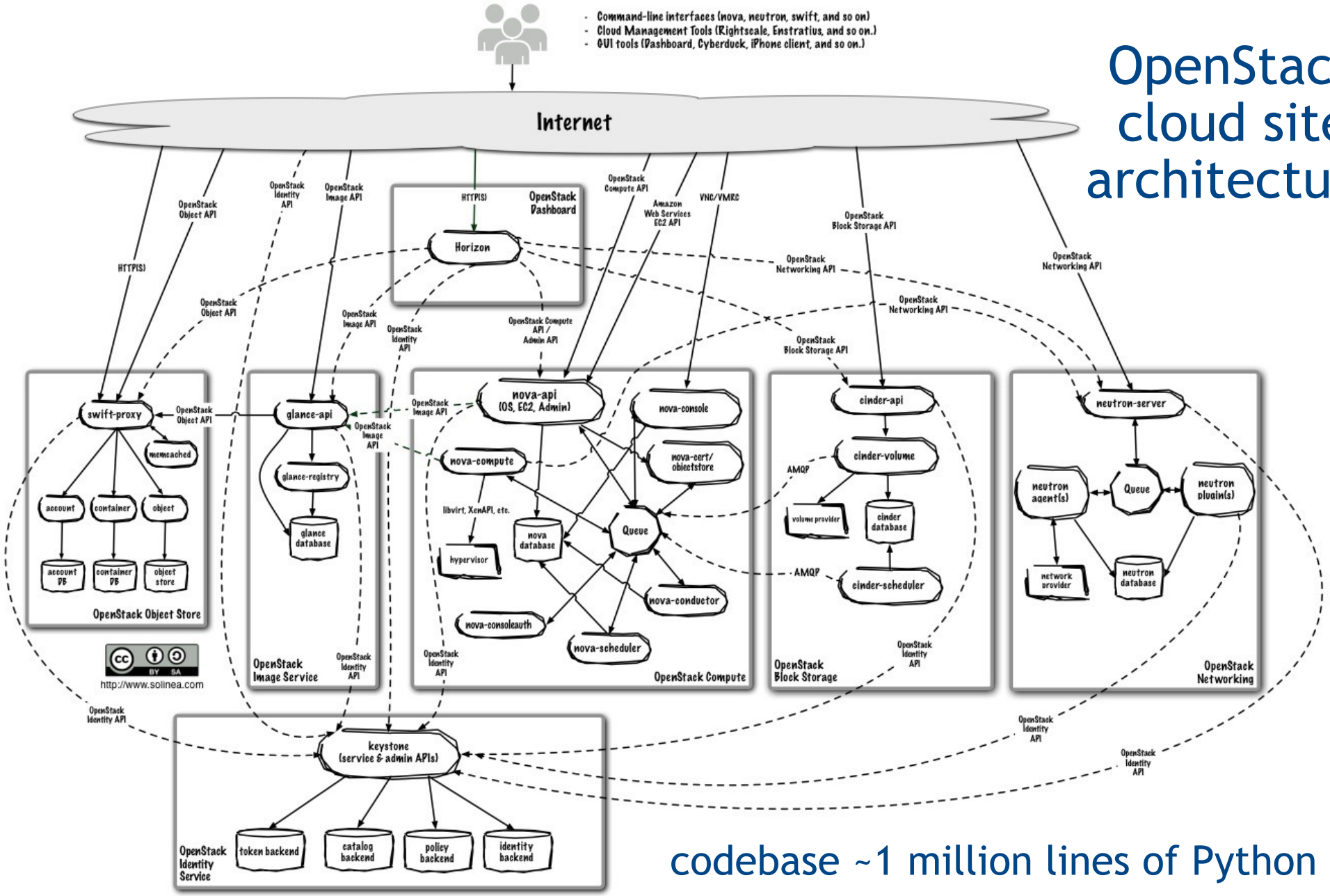
My personal simplification flowchart



Cloud systems: OpenStack etc

- Infrastructure-as-a-Service (IaaS) clouds popularised by Amazon with EC2
- All these systems provide APIs for creating VMs on a pool of hardware they manage
- Typically as complicated to run as a grid+batch site
  - So turning your hardware into a cloud isn't that a big simplification
  - (Although you can at least but books on all this)
- "Market leader" open source implementation is OpenStack (started and used by RackSpace)
  - CERN have now moved their interactive, batch, and services to OpenStack-managed VMs
- OpenNebula and CloudStack (Apache) also exist

OpenStack cloud site architecture

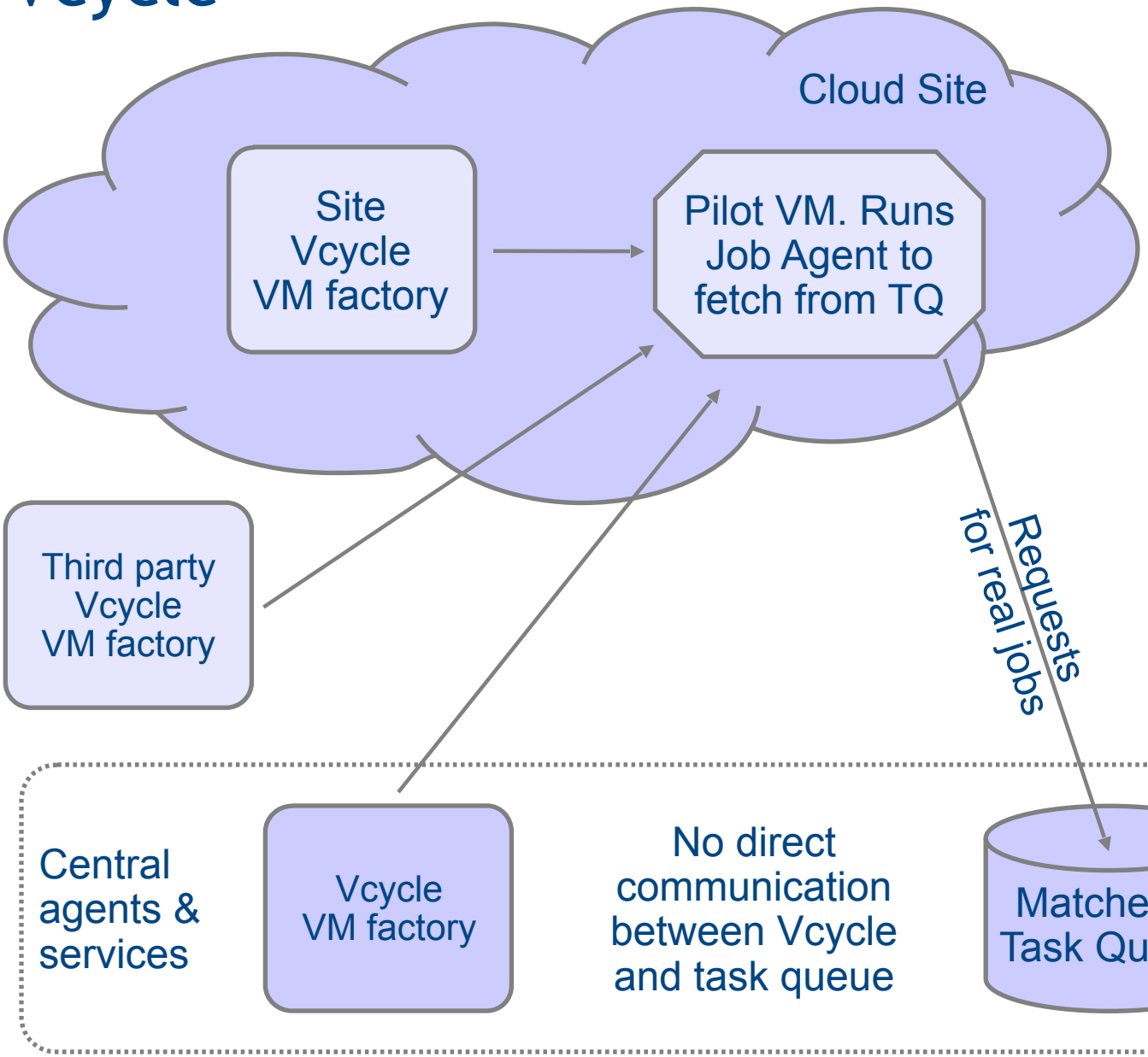


codebase ~1 million lines of Python

Cloud systems: OpenStack etc

- So assuming you have **someone else** who is offering to run all that for you, how to use it?
- You need a virtual machine lifecycle manager to decide which VMs to run
- I'd recommend Vcycle
  - Works with ATLAS, CMS, LHCb, GridPP DIRAC VMs
  - Can put experiments in the same tenancy for dynamic load balancing
- Also possible to use experiment-specific systems and dedicated tenancies
  - ATLAS: CloudScheduler
  - CMS: GlideinWMS
  - LHCb, GridPP DIRAC: Vcycle

Vcycle



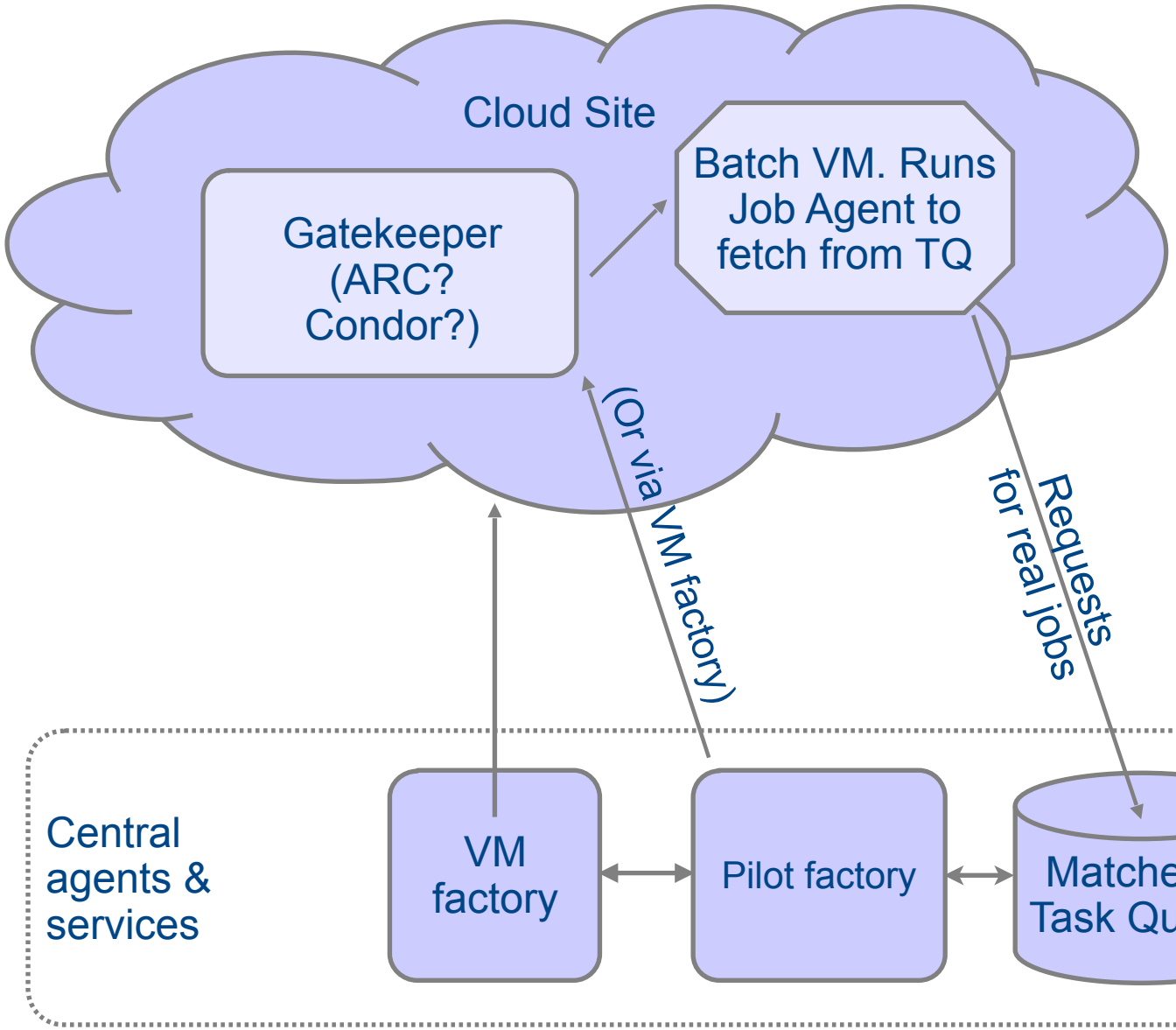
Apply Vac ideas to OpenStack etc

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")

Virtual “Grid with Pilot Jobs” site

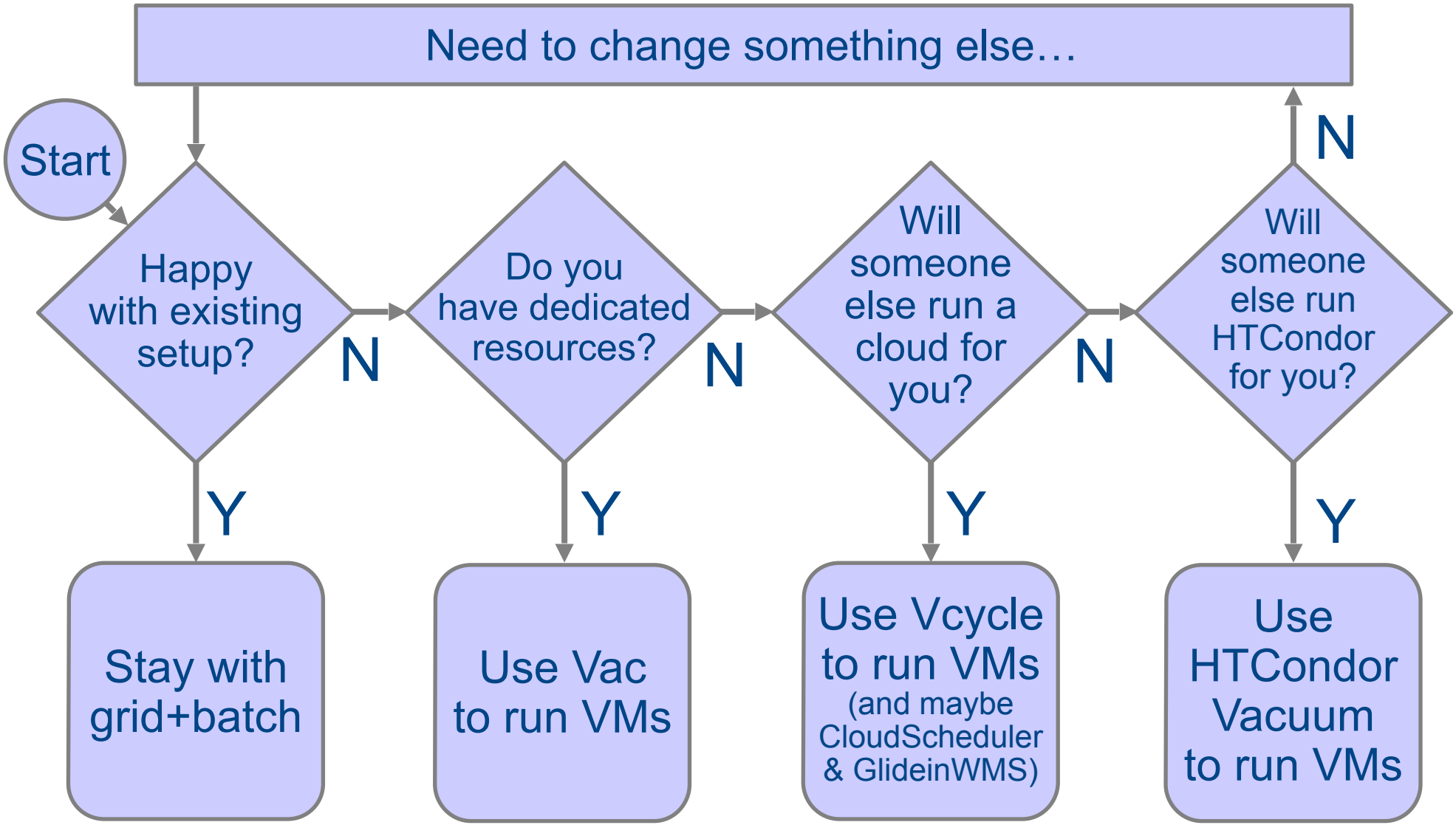


Experiment creates its own “conventional” grid site on the cloud resources.

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

CloudScheduler (ATLAS) is an implementation of this model.

My personal simplification flowchart



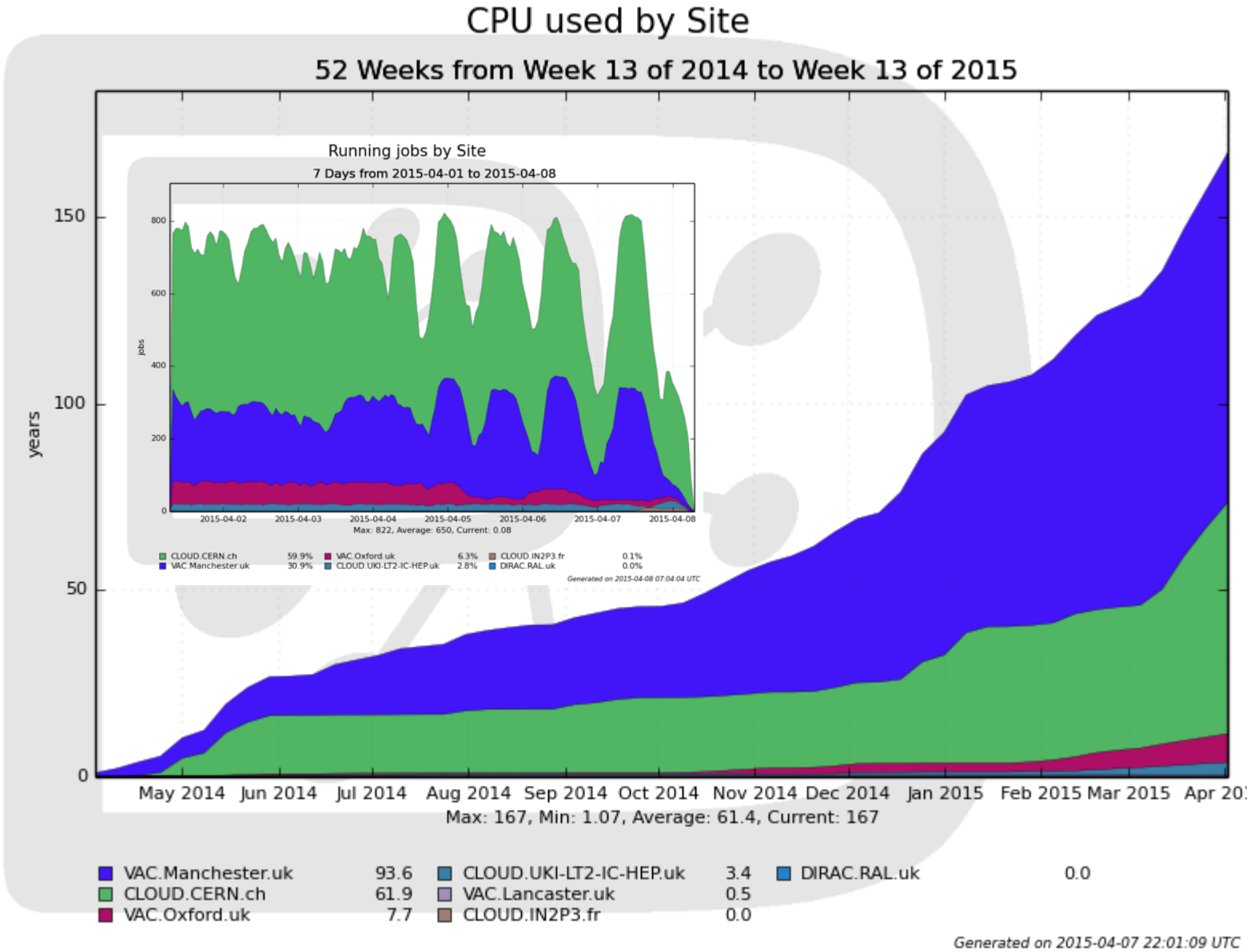
HTCondor Vacuum

- HTCondor can be made to create VMs
- HTCondor Vacuum is a lifecycle manager for deciding when to inject jobs to create VMs
- Uses same framework and VMs are Vac and Vcycle
  - Developed by Andrew Lahiff at RAL
- Not a huge simplification if you have to run HTCondor yourself
  - You still depend on HTCondor for everything
- However, like with clouds, if someone else will run HTCondor for you, you are free of that responsibility

Vac, Vcycle, HTCondor Vacuum deployment

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

LHCb jobs in VMs

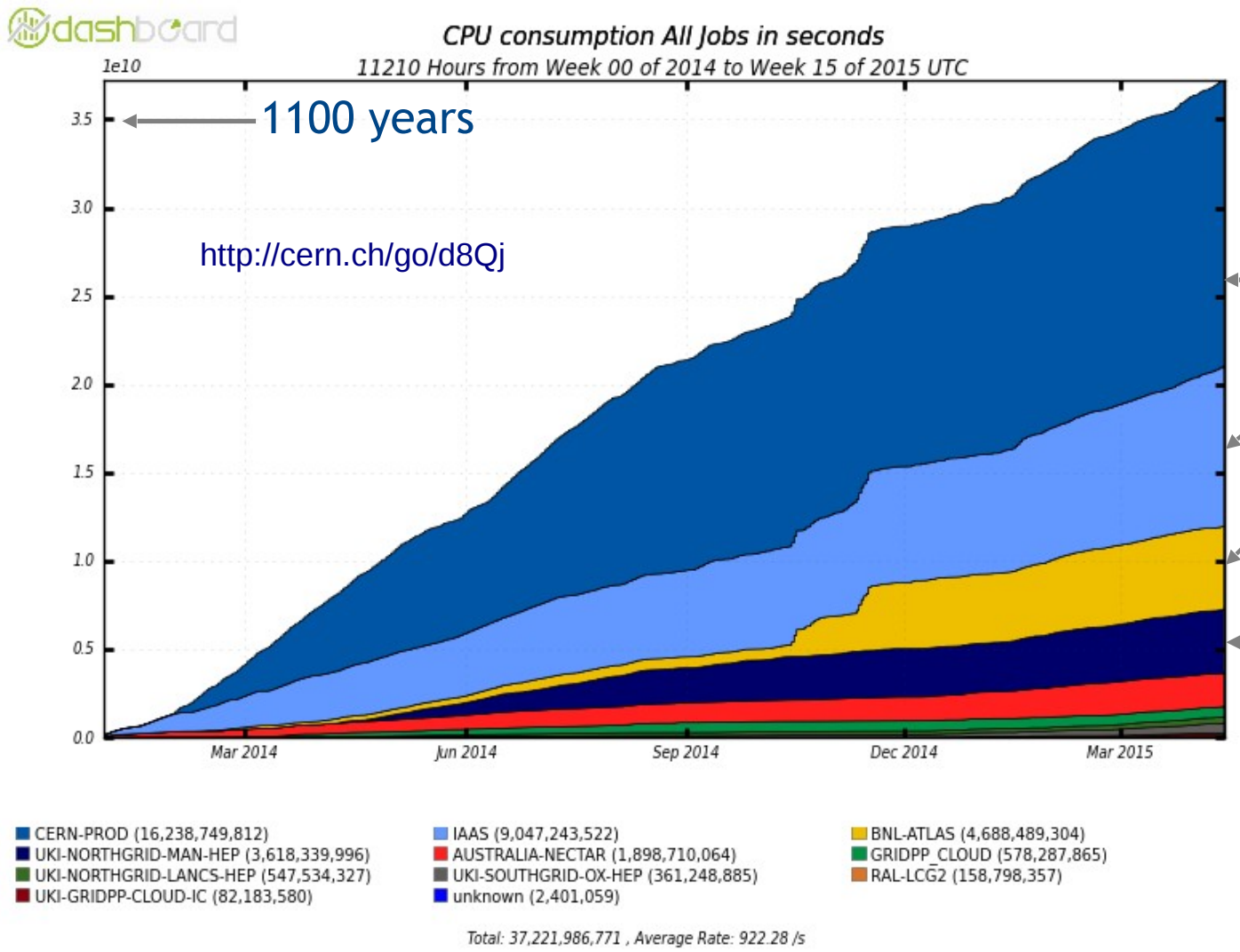


Routine production since May last year.

Increased capacity this year.

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

ATLAS jobs in VMs



CloudScheduler at CERN, Victoria, BNL

Vac at Manchester

(Slide from Ryan Taylor's ATLAS talk at CHEP)

Summary

- If you want to simplify things, virtualization offers some good options
- Working for ordinary production jobs from ATLAS, CMS, LHCb, and GridPP DIRAC service
- See my CHEP Track7 summary for an overview of the whole area