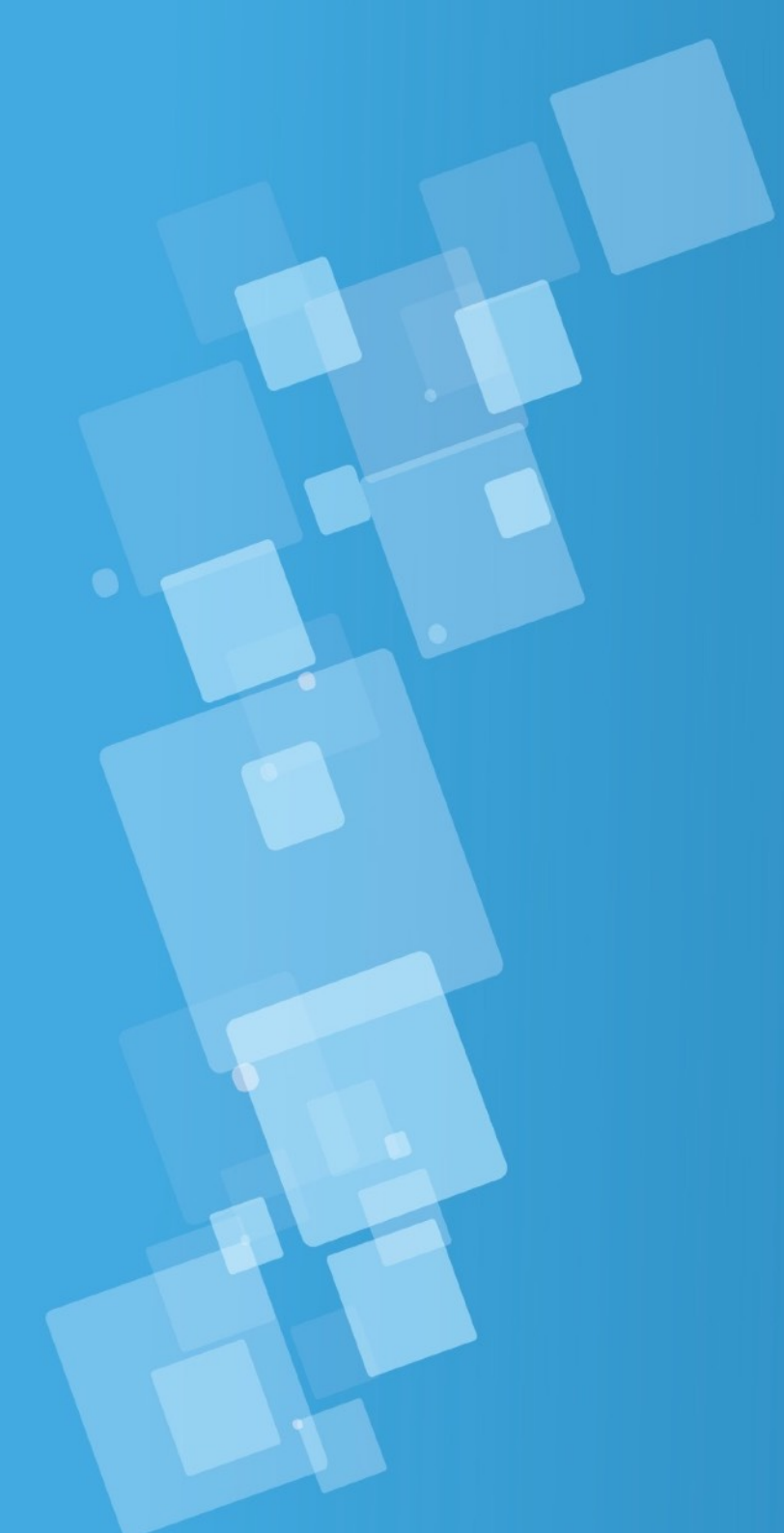


Vac, Vcycle, VMs status and plans

Andrew McNab
University of Manchester
LHCb

Overview

- Vac vs Vcycle
- VacMon, Pipes, Multiprocessor
- VMCondor
- Deployment status
- Docker containers in Vac
- Quality of Service and Tier-2/3 evolution
- Next steps

Vac vs Vcycle recap

- Two GridPP systems aimed at running VMs
- Vac - autonomous hypervisors
  - Each VM factory machine creates VMs in response to observed demand for each type of VM
  - Factory installation by Puppet etc or Vac-in-a-Box
- Vcycle - uses OpenStack, EC2, Google Cloud etc
  - VMs created via Cloud API in response to observed demand for each type of VM
  - Same VM definitions as Vac
- VMs are self-contained black boxes defined by experiments
  - Know how to pull in jobs to run from experiment HQ

New since GridPP37 Ambleside

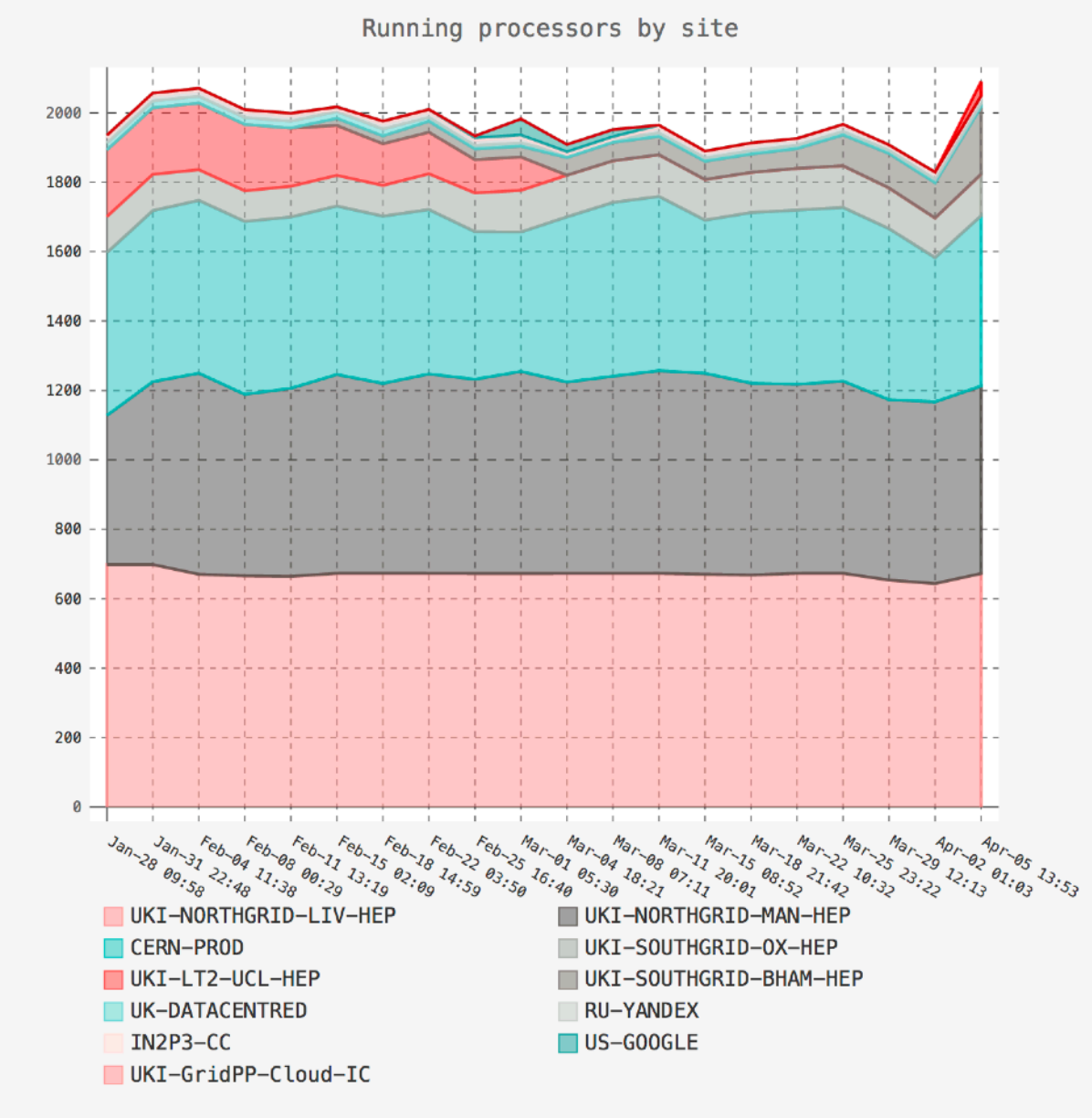
- VacMon - Ganglia-style monitoring at site, space, VM factory level
- Vac 2.0 deployed
  - Multiple VM sizes on the same VM factory: eg 8 and 1
  - Vacuum Pipes to reduce VO configuration to a URL
- ALICE VMs
  - Enabled Birmingham to start converting worker nodes to Vac
- VMCondor framework in production for ATLAS and ALICE, and available for generic VMs running HTCondor jobs
  - Should also work for CMS
- Google Compute Engine plugin for Vcycle

Deployment by site and experiment

		ATLAS	ALICE	LHCb	GridPP DIRAC
Vac	Birmingham	✔	✔	✔	
	Liverpool	✔	✔	✔	✔
	Manchester	✔	✔	✔	✔
	Oxford	✔	✔	✔	✔
	UCL	✔		✔	✔
Vcycle	Imperial			✔	✔
	CERN (LHCb)			✔	
	CERN (Dev)	✔	✔	✔	✔
	CC-IN2P3			✔	
	Yandex			✔	
	Datacentred			✔	✔
	INFN Naples		Belle II		

VacMon

- Ganglia-style monitoring at site, space, VM factory level
- Produces charts like this
- Uses Vac's internal JSON status message formats
- Sent over UDP to vacmon.gridpp.ac.uk
- Stored in ElasticSearch



Docker Containers

- Current Vac development is to add support for Docker Containers as another logical machine model alongside VMs
- Will be able to run arbitrary Docker images, or Vacuum Containers
  - Extending Vacuum Platform API to define how to provide CernVM-FS to unprivileged containers, init script as a volume etc
- So Vac factories will be able to run a mix of VMs and DCs alongside each other, using target share mechanism etc to decide what to start next
- Using LHCb container definition first, but will extend VMCondor framework too (so available to ATLAS + ALICE)

Tier-2/Tier-3 evolution ideas …

… in the context of falling effort during GridPP5

Quality of service

- QoS is a really important concept
  - It's behind Tier 0 vs 1 vs 2 distinction for instance
  - It's also why commercial clouds are still expensive
- Some services, some experiments are easier than others when it comes to providing a particular QoS
  - For example, you can "fix" a batch system with a misbehaving WN by turning the WN off.
  - "Fixing" a misbehaving RAID array is more complicated.
  - These kinds of QoS are associated with how long we can leave something broken, which is linked to staff availability
- We might choose to maximise QoS for major users of a site (eg WLCG experiments that are 90%+ of workload)

Vac and Quality of Service

- Part of the idea of Vac is to enable high QoS even when you don't have time to fix things quickly
  - Remember: autonomous VM (or DC) factories with no headnode or central point of failure
  - VM factories are also designed to degrade gracefully
- If 90%+ of your workload can be run on Vac, then you provide this high level of QoS with much less effort
- Then you could, for example, run the 10%- on a best effort basis, or declare a downtime and just run Vac VMs till you have time to fix things
- Implicitly, this kind of scenario calls for sharing of resources between Vac and batch

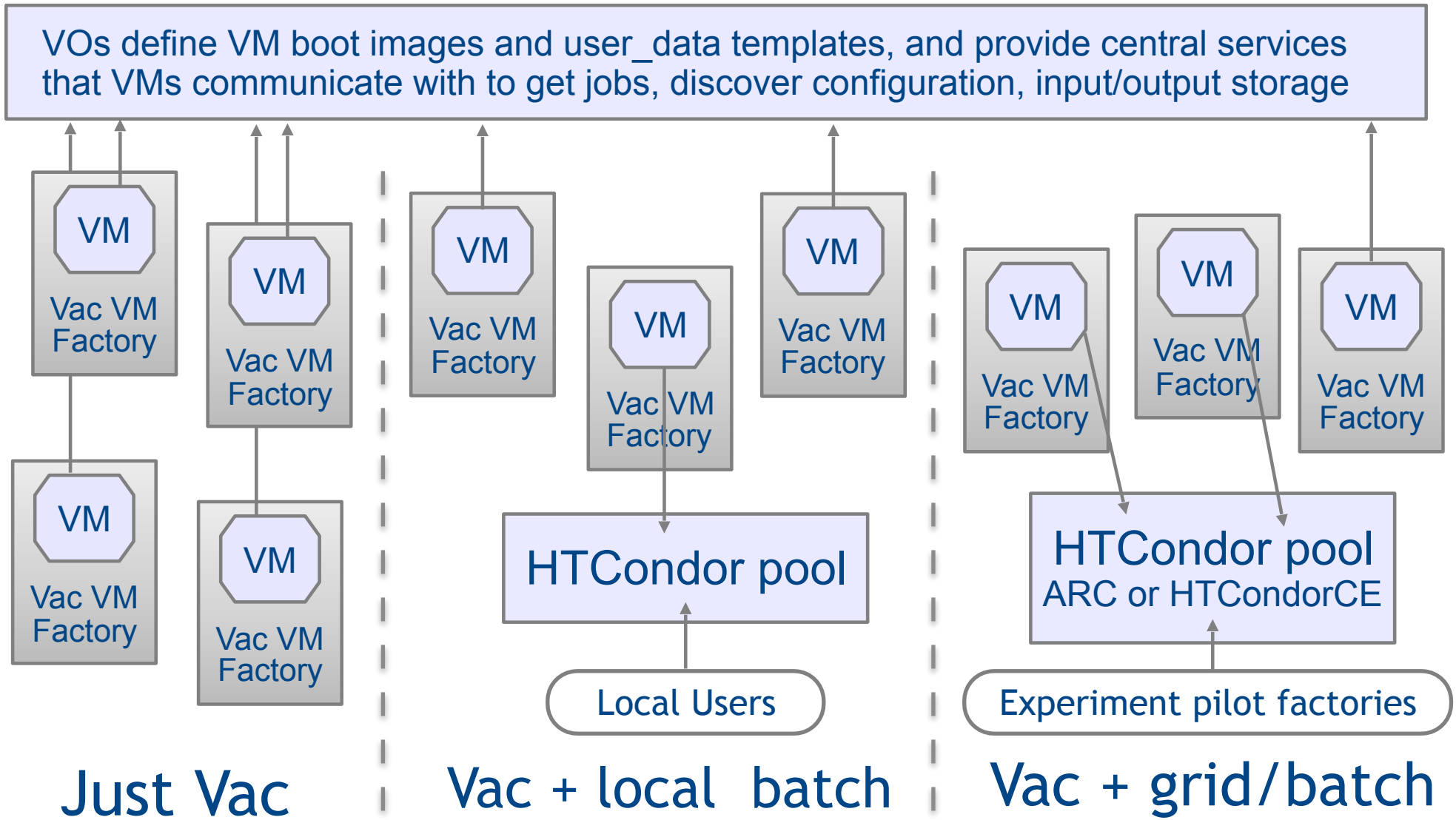
Scenario 1: local or “Tier 3” batch

- VMCondor definition can be used to attach VMs to a local HTCondor pool
- So we can have batch workers in VMs managed by Vac
  - Compare cloud extensions of CERN batch system
- When local users have lots of jobs, VMCondor VMs get run; when no local jobs, VMs for ATLAS, LHCb, … are run
  - HTCondor makes things a lot easier, as it was designed for machines that repeatedly join and leave pools
- You can get the Vac resilience to failure for free too
- Either set up an HTCondor pool specifically; or allow VMs to join your existing pool that is used by dedicated local/Tier-3 WNs
- Long term target shares would reflect grid vs local funding

Scenario 2: ARC/HTCondorCE WNs as VMs

- Could apply same idea to grid+batch provision
- Still need to run the set of middleware services to provide the grid API for experiments that need it
  - Feed into HTCondor pool which VMCondor VMs can access
  - Same set of Vac VM factories supports native Vac VMs and VMCondor - HTCondor - ARC/HTCondorCE
- This means if you run into problems with middleware or an update takes a long time, you can put those services into downtime
  - But because 90%+ of your workload does run in dedicated VMs, you just run more of those VMs for now
  - Also, you don't need to maintain WN definition, apply security patches etc (it's all done centrally by CernVM team)
- So high QoS for major users; lower QoS for minor users

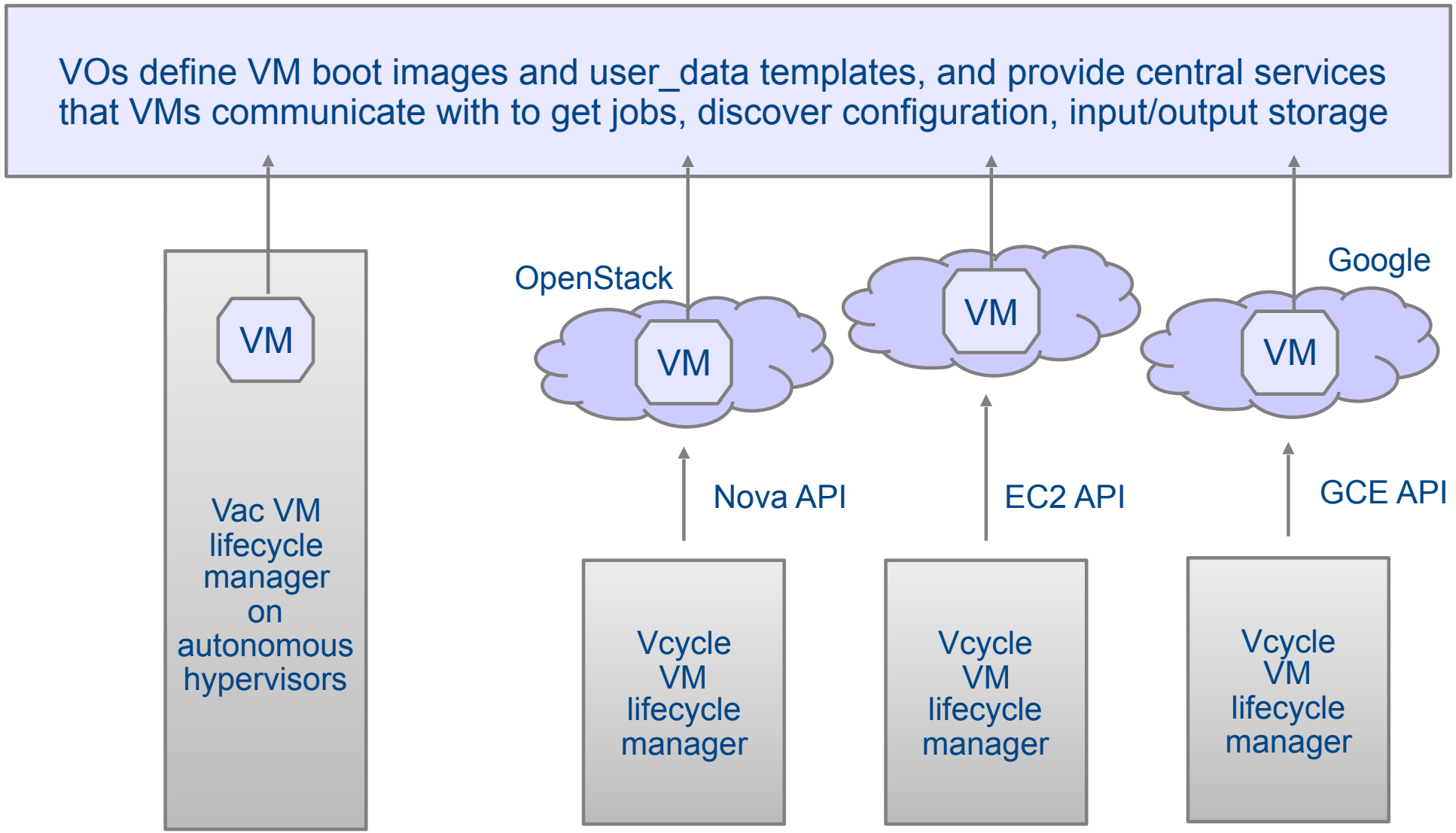
Scenario 0, 1, 2



Summary and next steps

- Vac 2.0 deployed
  - Better multiprocessor support and Pipes
- VMCondor framework
- VacMon monitoring website
- Two scenarios for using Vac with HTCondor locally
- Docker containers being added for Vac 2.1
- Major missing piece is (re)creation of CMS VM definition for Vac/Vcycle

Vacuum platform



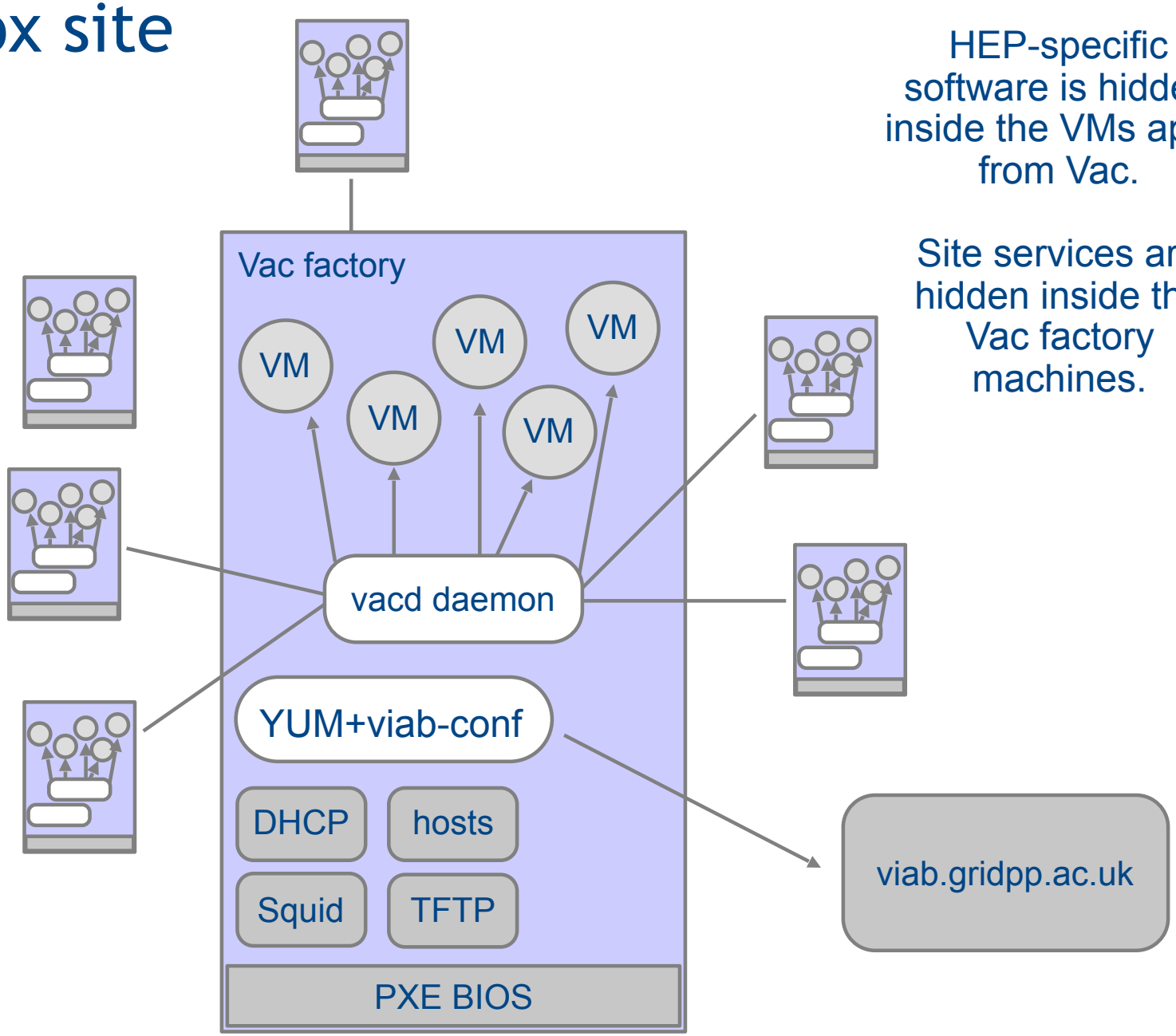
Vac-in-a-Box site

Simpler than installing via Puppet, Ansible etc.

Per-site dashboard at viab.gridpp.ac.uk

Kickstart from the website.

viab-conf RPM with configuration, via autoupdates from YUM repo.



HEP-specific software is hidden inside the VMs apart from Vac.

Site services are hidden inside the Vac factory machines.

Vac-in-a-Box dashboard