Vac containers

Shipping containers in Pitlochry

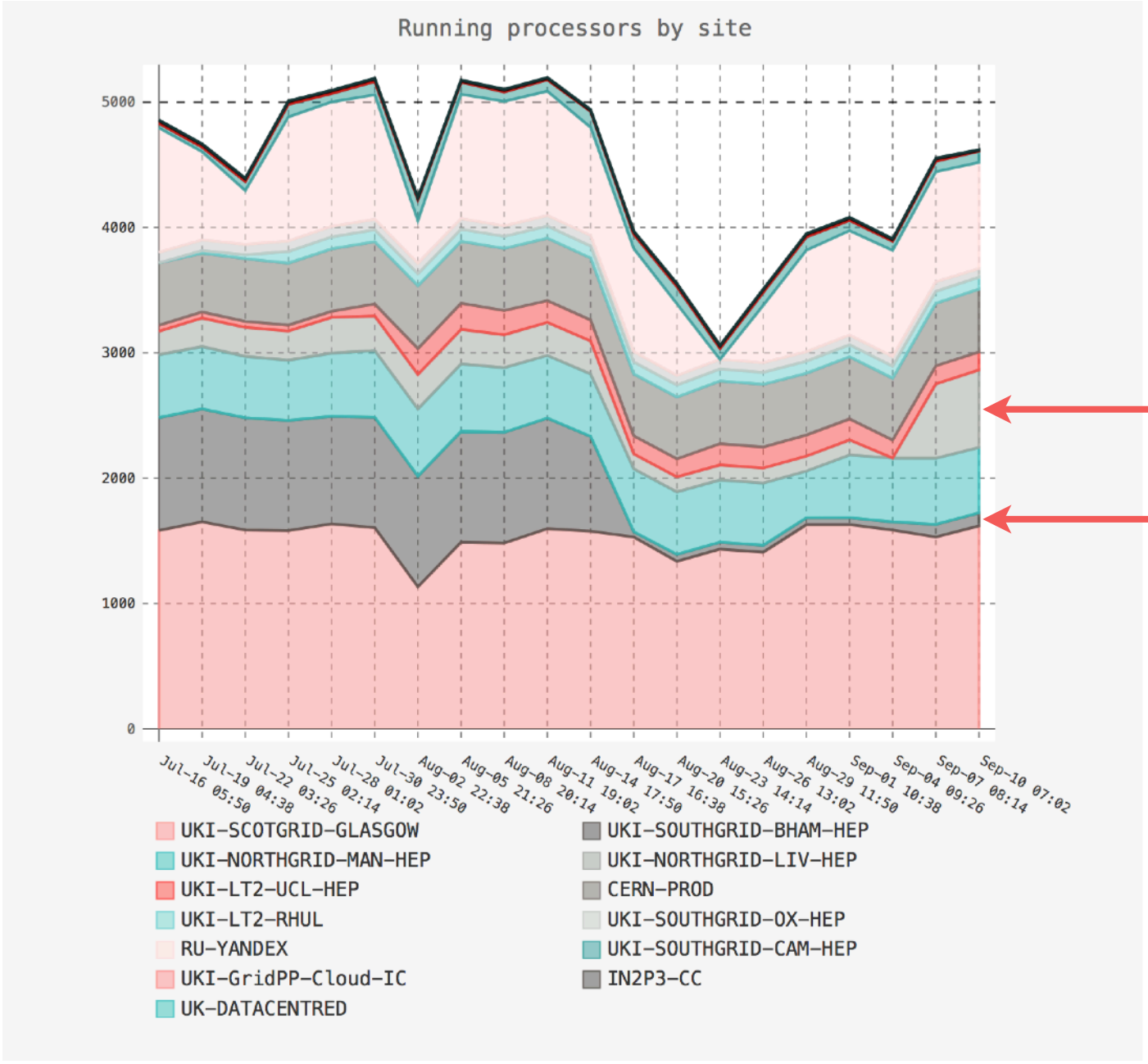
Andrew McNab
University of Manchester
GridPP and LHCb

Overview

- Current Vac status
- Singularity Containers
- Docker Containers
- Production VMs, SCs, DCs
- Implications
- Vac sites supporting VMs/SCs/DCs
- Next steps

Wider Vac status

- New sites: Cambridge, Glasgow, RHUL
  - Vac/Vcycle running at 50% of GridPP Tier-2 sites
- Most sites using 2.00 release, from the start of the year
- 2.01pre in production testing at Manchester
  - Now supports CentOS7 in addition to SL6
  - Also exposes all CPU features from hypervisor (done for SKA)
  - But other than that no changes wrt 02.00
- Vac-in-a-Box being updated to use CentOS7
- Vcycle continues managing OpenStack at Imperial, CERN, Yandex, and CC-IN2P3



Liverpool and Birmingham machine room upgrades

Deployment by site and experiment

		ATLAS	ALICE	LHCb	GridPP DIRAC
Vac * = new since GridPP38	Birmingham	✔	✔	✔	✔
	Cambridge*	✔		✔	✔
	Glasgow*	✔		✔	✔
	Liverpool	✔	✔	✔	✔
	Manchester	✔	✔	✔	✔
	Oxford	✔	✔	✔	✔
	RHUL*	✔		✔	✔
	UCL	✔		✔	✔
Vcycle	Imperial			✔	✔
	CERN (LHCb)			✔	
	CERN (Dev)	✔	✔	✔	✔
	CC-IN2P3			✔	
	Yandex			✔	
	DataCentred			✔	✔

Containers

- Vac started as a way of running Virtual Machines on autonomous hypervisors
  - Uses libvirt/kvm and (usually) CernVM model
  - Simulates the OpenStack/EC2 API presented to VMs
- Lots of interest in HEP in Docker and now Singularity containers
- For High Throughput Computing HEP jobs, these containers can be like lightweight VMs
  - “Logical Machines” rather than single services or applications each in their own container
- Vac already has the machinery to handle configuration, provision images, customize run script templates
- **Container support now implemented in Vac using existing model**
  - Provision containers in the Vac “slots” of CPU, memory, disk

Singularity Containers

- Core of Singularity is its command
  - Works very like the chroot command
  - The command is the API
- Sets up namespaces to maps areas of filesystem for isolation
  - But users are the same inside and outside the container
  - Vac uses a non-privileged account for this e.g. vacsngly
- Vac uses bind mounts to share run script, boot image hierarchy, Machine/Job Features directories into the container
- Vac monitors the singularity process it created to see if the container is still running

Singularity Containers (2)

- Vac contextualizes VMs by providing the experiment's user_data file using the same API as EC2 and OpenStack
- For containers, it binds the provided file at /user_data
  - After ##user_data_….## template substitutions
  - user_data might be a shell script or something else the container understands
- Can specify the script to run: /user_data by default
- Currently specify the image as a directory hierarchy
  - /cvmfs/cernvm-prod.cern.ch/cvm3 works to give SL6
- If any cvmfs repositories are requested, then /cvmfs is shared too (and the repos are kept mounted by Vac)

Log file excerpt (newlines for clarity)

Sep 11 19:32:19 [1584]: Creating SC with

/usr/bin/singularity exec --contain

--workdir /var/lib/vac/machines/1505154729_lhcb-prod-sc/mnt

--bind /cvmfs:/cvmfs

--bind /var/lib/vac/machines/1505154729_lhcb-prod-sc/machinefeatures:/tmp/machinefeatures

--bind /var/lib/vac/machines/1505154729_lhcb-prod-sc/jobfeatures:/tmp/jobfeatures

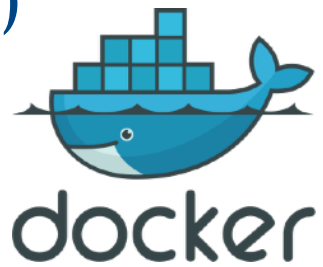
--bind /var/lib/vac/machines/1505154729_lhcb-prod-sc/joboutputs:/tmp/joboutputs

--bind /var/lib/vac/machines/1505154729_lhcb-prod-sc/user_data:/user_data

/cvmfs/cernvm-prod.cern.ch/cvm3 /user_data

Docker Containers

- Older, more mature, more complex than Singularity
  - Many scientific users creating Docker containers
- Not just filesystem namespace (network, users, etc)
- Vac handles Docker the same way as Singularity
- Host shares MJF, /user_data, cvmfs if requested
- Vac can run arbitrary images from Docker repository
- vacproject/vcbusybox image enables the CernVM root filesystem as with Singularity
  - its /init script sets things up and then runs /user_data
- Docker uses cgroups to limit CPU, memory etc
  - Vac uses this for usage for accounting, monitoring



Log file excerpt (newlines for clarity)

Sep 11 19:53:02 [4759]: Creating DC with

/usr/bin/docker run --detach

-v /var/lib/vac/machines/1505155980_lhcb-prod-dc/joboutputs:/var/spool/joboutputs

-v /var/lib/vac/machines/1505155980_lhcb-prod-dc/user_data:/user_data:ro

-v /cvmfs:/cvmfs:ro

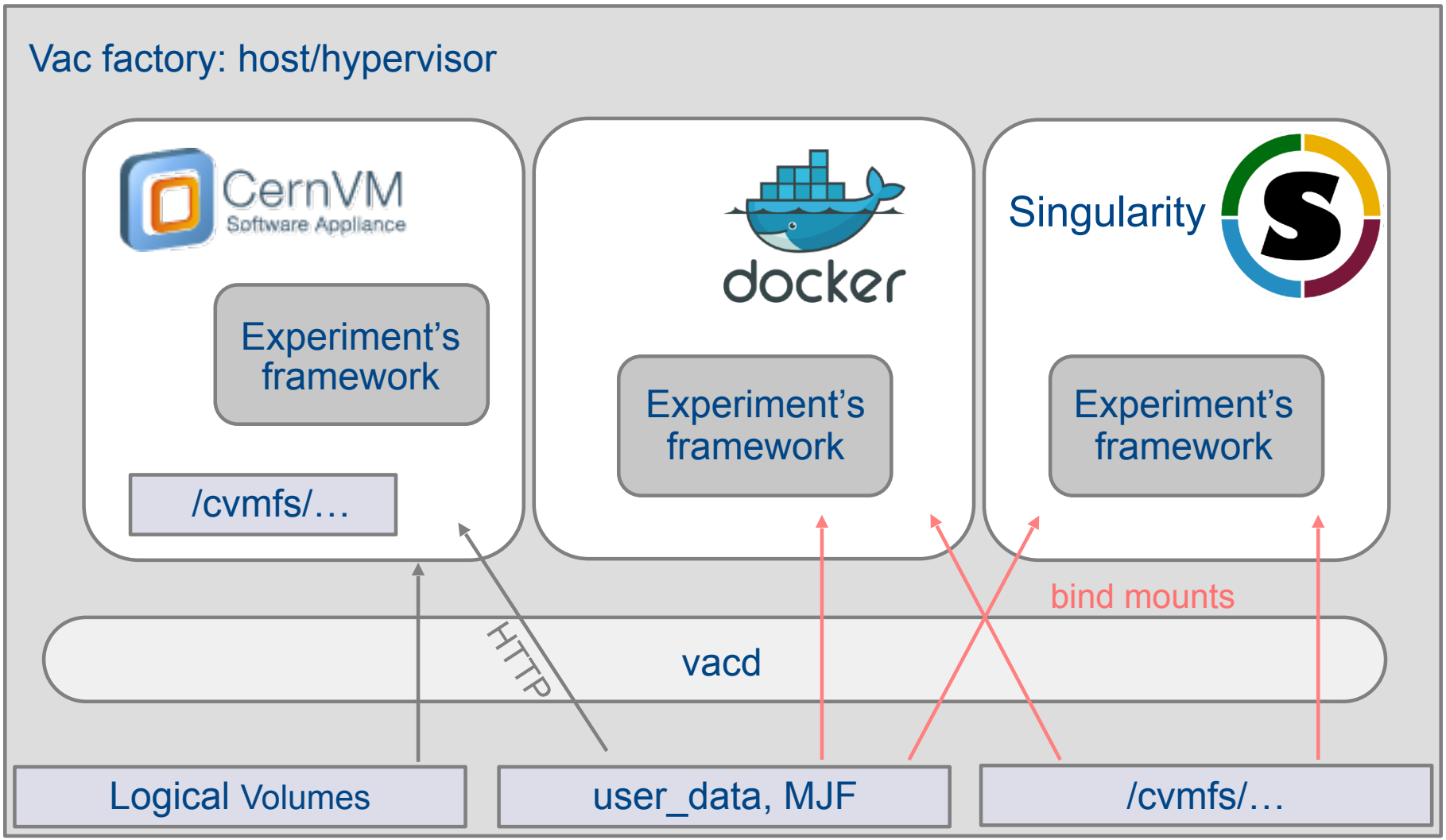
-v /var/lib/vac/machines/1505155980_lhcb-prod-dc/machinefeatures:/etc/machinefeatures:ro

-v /var/lib/vac/machines/1505155980_lhcb-prod-dc/jobfeatures:/etc/jobfeatures:ro

--name vac-85-03.hep.manchester.ac.uk

--hostname vac-85-03.hep.manchester.ac.uk vacproject/vcbusybox /init

Vac factory with VMs and Containers



Some running Logical Machines

- Using the normal “vac machines” command
  - Shows the logical machines that are running
- Logical Machine hostname, machine type, state, number of logical processors, model, hours running, and CPU loads
- Tests with production LHCb MC in LMs
- As you can see, a mix of CernVM VMs, Singularity Containers and Docker Containers on the same hypervisor
  - Vac is deciding which type to run based on what is currently finding work to do

```
root@vac-85: vac machines
vac-85-00       lhcb-prod-sc     Running     1 SC 15.30 hrs
vac-85-01       lhcb-prod-vm     Running     1 VM 15.28 hrs 99.4%      100.0%
vac-85-02       lhcb-prod-dc     Running     1 DC 15.26 hrs 99.4%       98.3%
vac-85-03       lhcb-prod-dc     Running     1 DC 15.22 hrs 99.5%      100.0%
```

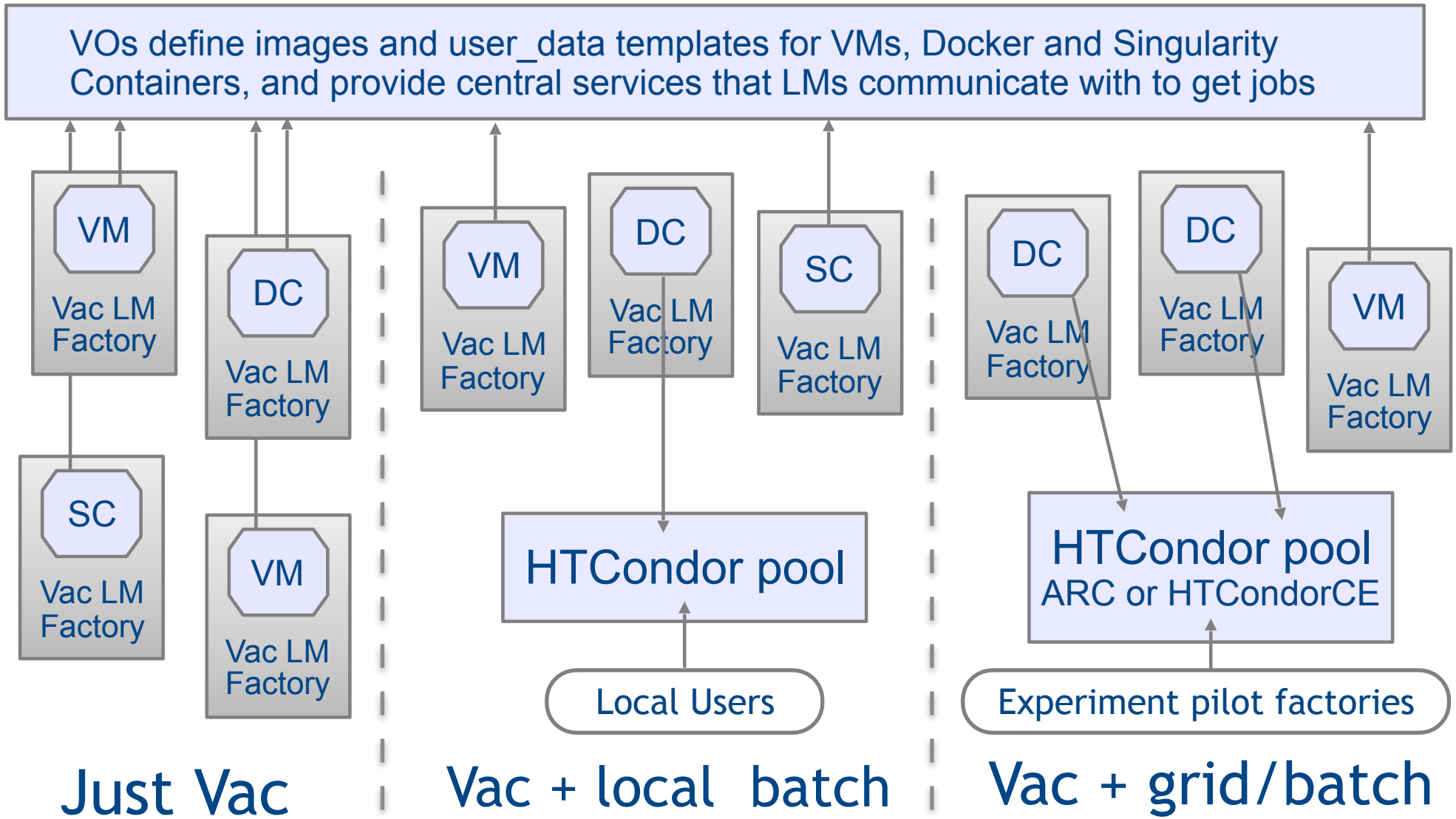
Logical Machines, with native commands

```
root@vac-85: virsh list
 Id    Name                           State
----------------------------------------------------
 567   vac-85-01.hep.manchester.ac.uk running


root@vac-85: docker ps
CONTAINER ID        IMAGE                   COMMAND    CREATED       STATUS
PORTS               NAMES
5bb83858b510        vacproject/vcbusybox   "/init"    15 hours ago  Up 15 hours
vac-85-03.hep.manchester.ac.uk
9eb5f3410765        vacproject/vcbusybox   "/init"    15 hours ago  Up 15 hours
vac-85-02.hep.manchester.ac.uk

root@vac-85: ps -u vacsngly -o user,pid,ppid,comm
USER       PID  PPID COMMAND
vacsngly 14706     1 sexec-suid
vacsngly 14714 14706 user_data
vacsngly 14715 14714 user_data
vacsngly 14734 14715 python
vacsngly 14819 14734 python
vacsngly 14820 14819 python
vacsngly 14978 14820 Job178170562
vacsngly 14979 14978 python2.7
vacsngly 15142 14979 python
vacsngly 15146 15142 python
vacsngly 15262 15146 python
```

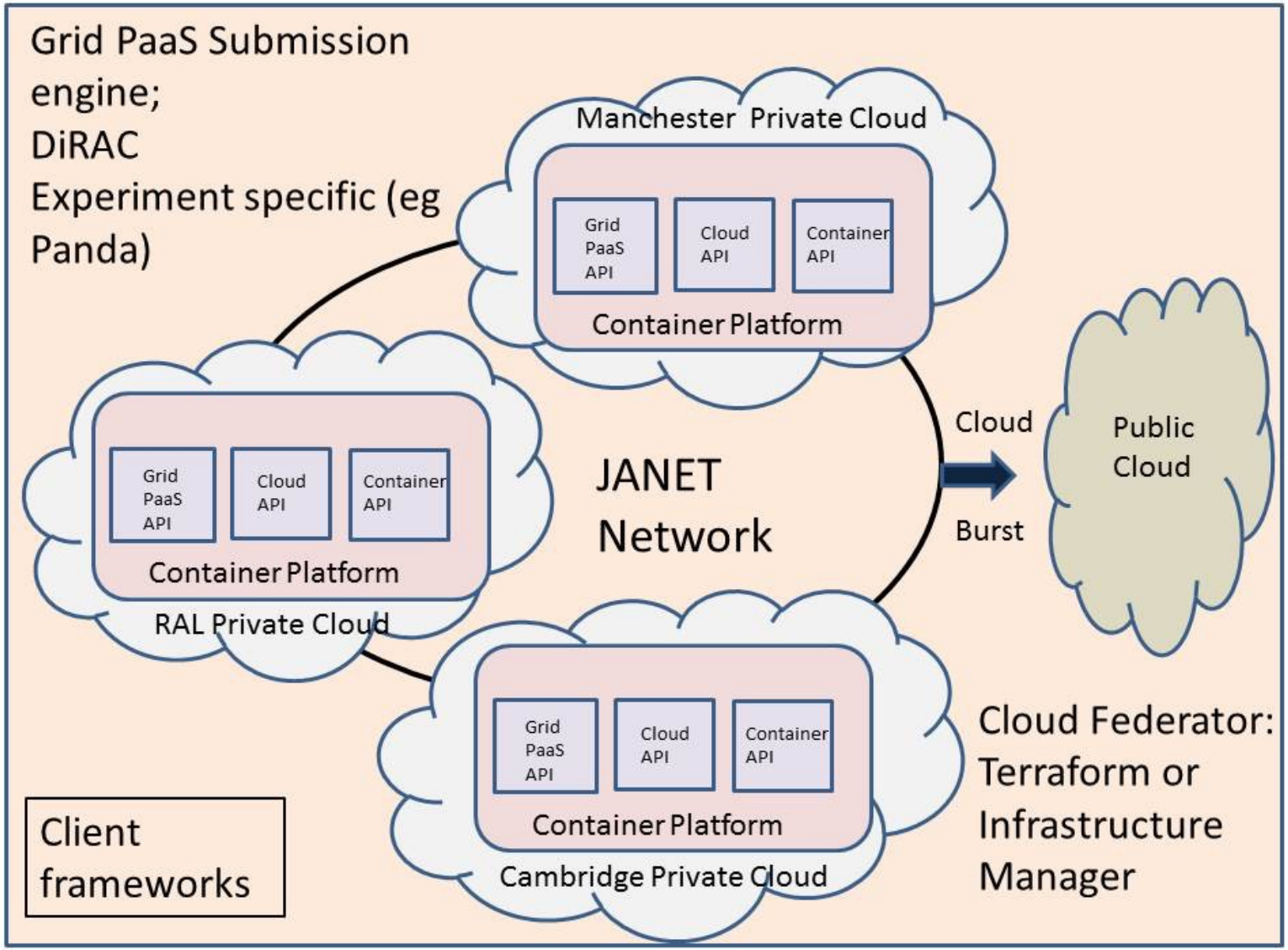
Implications

- We can now run VMs, Docker and Singularity containers provided by experiments
- Using containers rather than VMs allows more flexibility about memory, disk, CPU etc limits
  - cgroups are more forgiving than VMs when there isn't contention
- Lots of other user communities have targeted Docker as a way of making their application portable
- As outlined at GridPP38, we can set up mixed sites
  - VMs/Containers running experiment frameworks directly
  - Providing worker nodes for conventional Batch/CEs
  - Hadoop, Spark etc which allow nodes to join dynamically

Scenarios for mixed VM/DC/SC sites with Vac



CPU in the e-Infrastructure bid



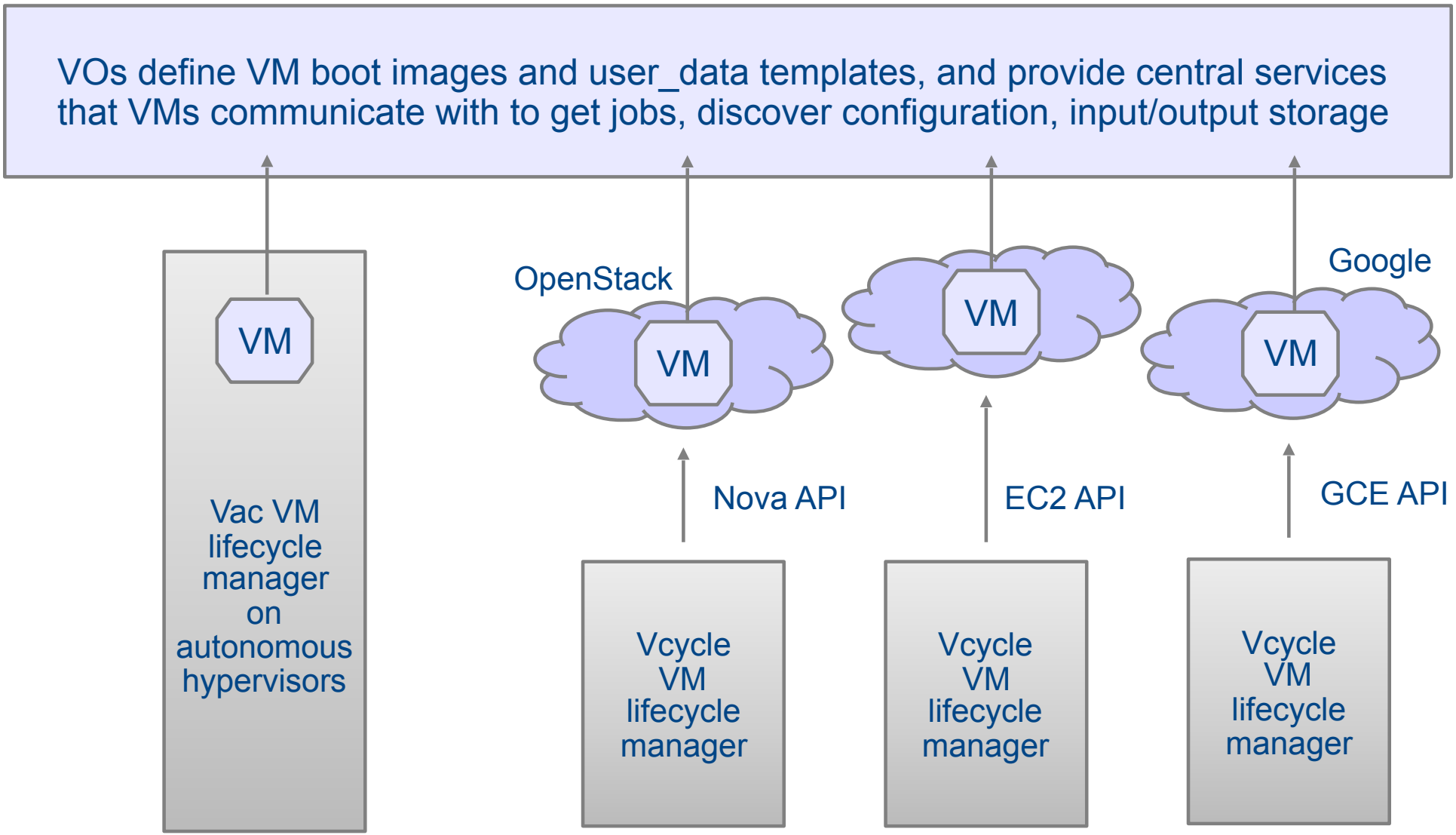
Next steps

- For 3.00 release:
  - Wrap Singularity containers inside cgroups managed by Vac to control usage and stop processes “escaping”
  - Use Docker’s cgroups to enforce CPU/memory usage
  - Implement/test all image sources for both models
    - e.g. images on random HTTPS server nominated by experiment
- Create Docker container definitions for ATLAS, ALICE, GridPP DIRAC based on existing VM definitions
- Provide a way of running containers in VMs managed by Vcycle?

Summary

- Vac 2.01pre delivers CentOS 7 support
  - Vac-in-a-Box being ported to CentOS 7
- Vac 3.00pre adds Singularity and Docker Container support
  - Successfully running LHCb VMs, SCs and DCs on the same hypervisor
  - A few details still to be implemented
    e.g. full resource control via cgroups
- This allows us to support a mix of VMs and Containers at sites
- (And Vac is still only 3900 lines of Python …)

Vacuum platform



Vacuum Pipes

- “Pipelines supplying VM components to VM factories”
- To define a VM (LM) in Vac and Vcycle requires a few lines of configuration
  - URL of user_data contextualization file
  - URL of boot image
  - Times: lifetime, heartbeat timings, “fizzle time”
- A Vacuum Pipe is a single URL with all this in a JSON file
- This means that adding a new VO to a site involves adding one URL to config
- Still need X.509 cert/key for authentication to VO
  - But for GridPP DIRAC, all VOs use the same cert/key

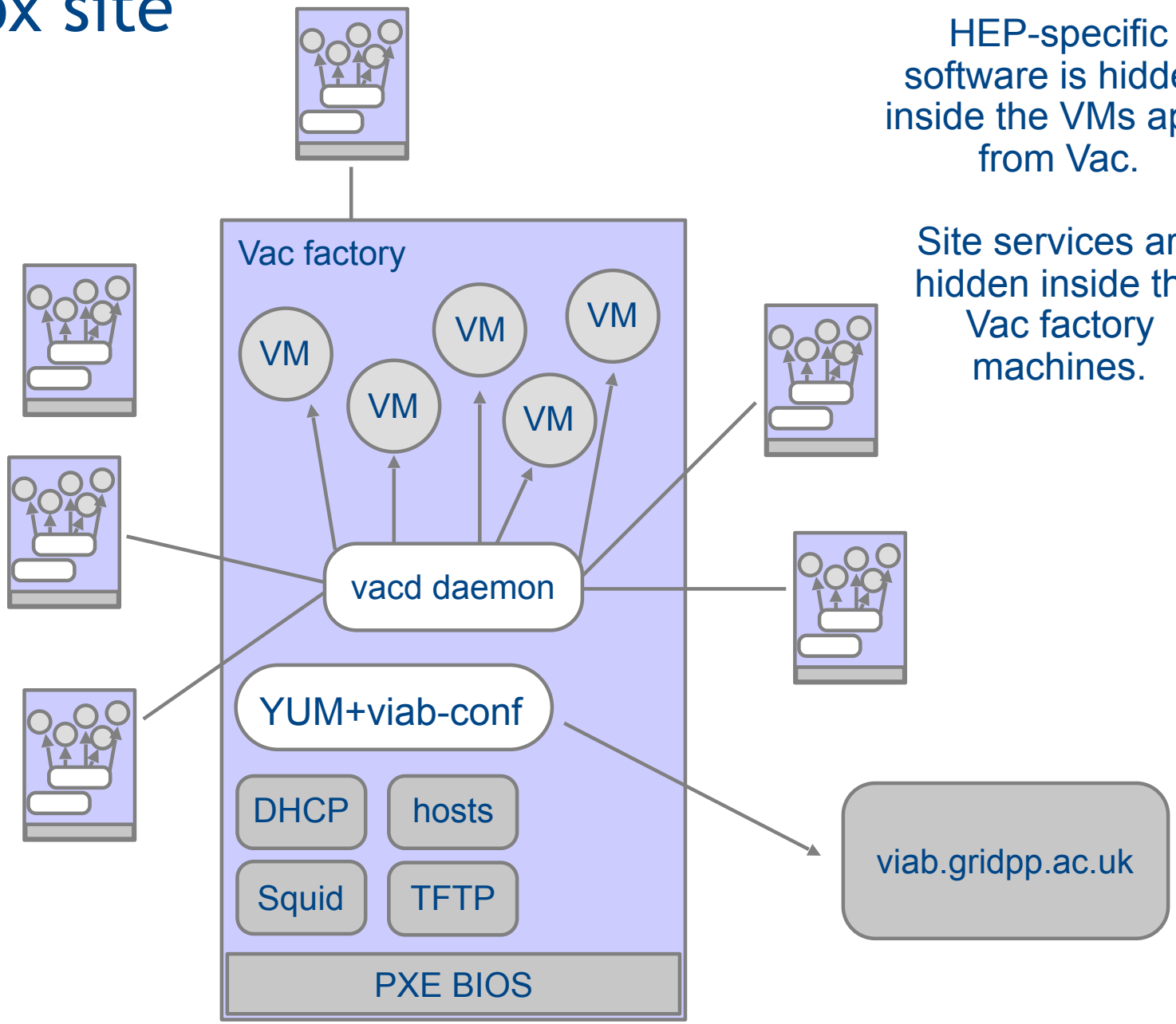
Vac-in-a-Box site

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