Containers support in Vac

Shipping containers in Pitlochry

Andrew McNab
University of Manchester
GridPP and LHCb

Containers

- Vac started by GridPP in the UK 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 also 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

Docker Containers with Vac

- Older, more mature, more complex than Singularity
  - Many scientific users creating Docker containers
- Not just filesystem namespace (network, users, etc)
  - So easier to make workernode-like environments
  - Can create users, run sudo or singularity etc
- Vac can run arbitrary images from Docker repositories
- Docker uses cgroups to limit CPU, memory etc
  - Vac uses this for accounting, monitoring
- (Vac also supports Singularity containers, but we concentrate on Docker here)

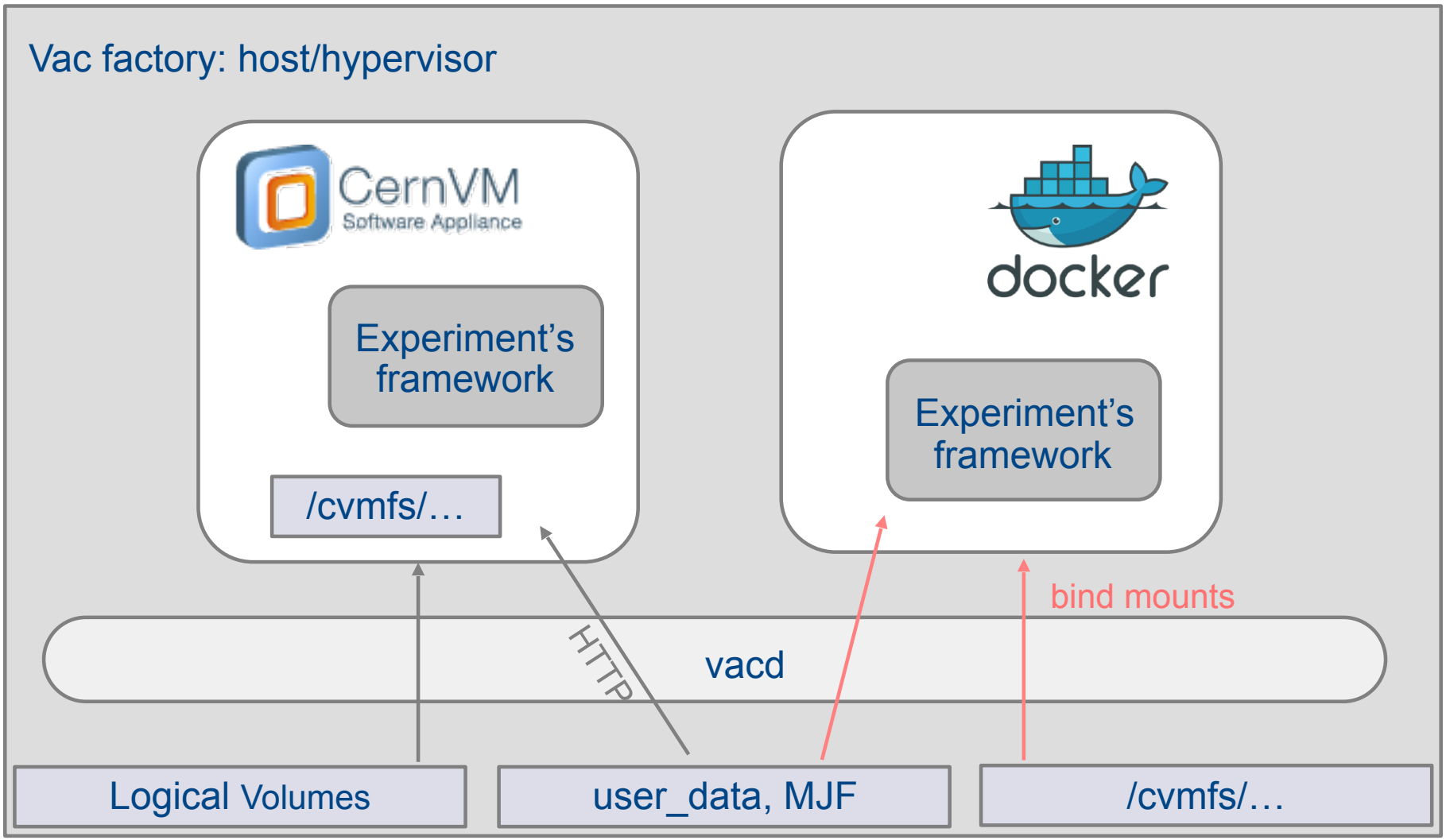
“Vacuum Containers”

- Extend Vacuum Platform API to work with Containers
  - (For Docker and demonstrated with Singularity Containers)
- Containers defined in “Vacuum Pipe” JSON file published by the experiments, alongside existing VM definitions
  - Image URL, user_data URL, cvmfs repositories, maximum lifetime, number of processors needed, …
- Host shares Machine/Job Features directories, /user_data, requested cvmfs repositories as bind mounts
- Container then runs as normal, and when it finishes writes relevant log files and shutdown_message to /var/spool/joboutputs (also a bind mount), and then stops
- Vac uses the shutdown_message to decide whether to create more containers like that
  - Ran ok? Found no work? Failed with an error?

VMs vs Docker Containers

- 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
- If any cvmfs repositories are requested, then /cvmfs is shared too
  - The repos are kept mounted by Vac in case the host uses auto-(un)mounting

Vac factory with VMs and Containers



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

Some running Logical Machines

vac command vs native VM and Docker commands

```
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%
```

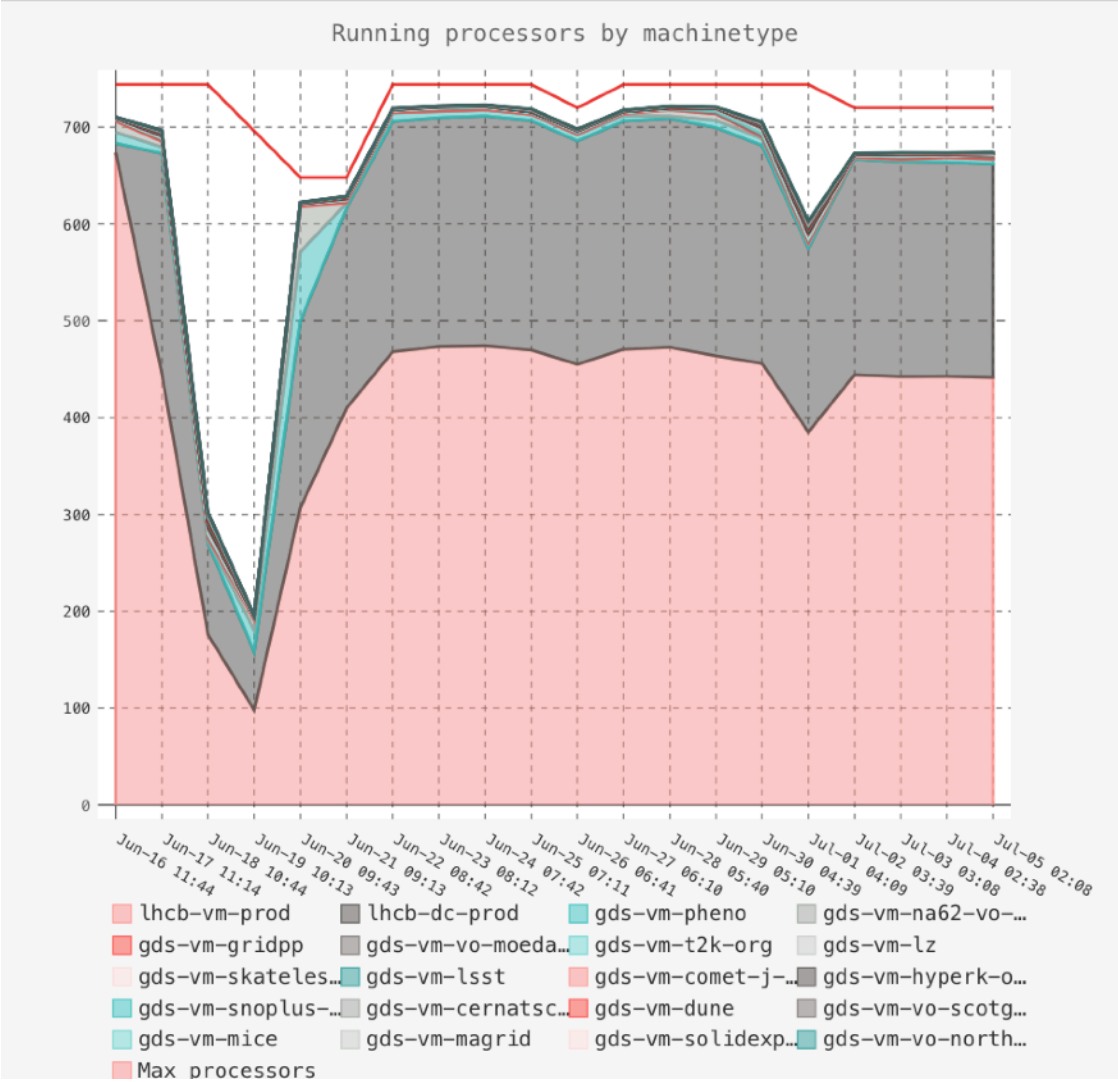
```
root@vac-85: virsh list
 Id    Name                           State
----------------------------------------------------
 567   vac-85-01.hep.manchester.ac.uk running
```

```
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
```

Containers and VMs at Manchester

- LHCb Docker containers in grey
- LHCb VMs in pink
- Before and after a software update
- Running alongside some other Vac workloads

- See the LHCb Containers talk later this session for the internals of these containers …



Implications and next steps

- We can now run both VMs and Docker containers provided by experiments, on the same pool of hypervisors
- 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
  - One of the target platforms for IRIS infrastructure in the UK (PP, Astro, Nuclear)
- Working on generic VMs to allow Docker containers to run at VM-only OpenStack sites managed by Vcycle
  - Using same API to discover the user_data, provide cvmfs etc

Summary

- Vac 3.0 added Docker Container support
  - First implementation of “Vacuum Container” API
  - Successfully running LHCb VMs and DCs on the same hypervisor
  - This allows us to support a mix of VMs and Containers at sites
- (And Vac is still only 3900 lines of Python …)