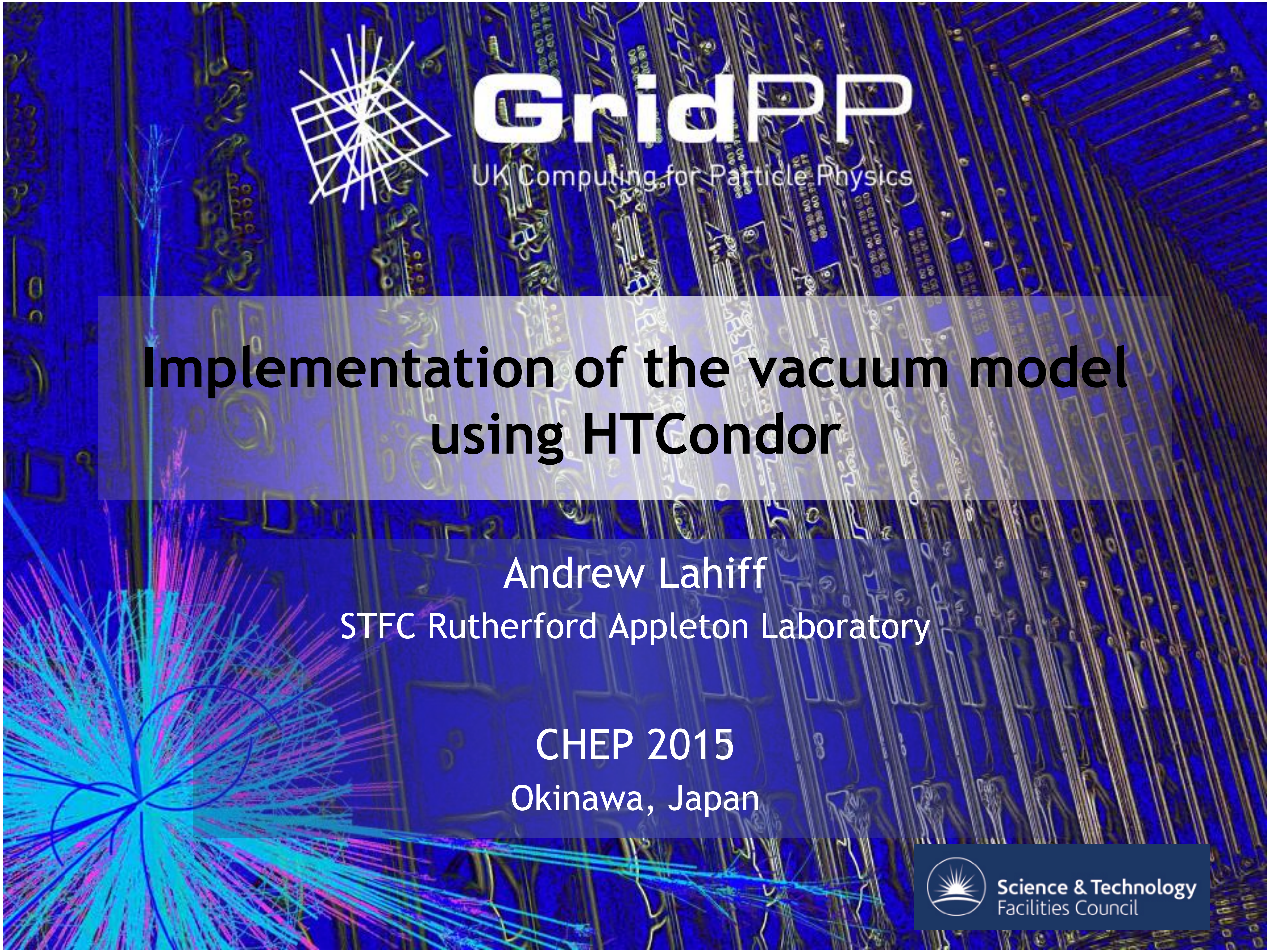


GridPP

UK Computing for Particle Physics

Implementation of the vacuum model using HTCondor

Andrew Lahiff

STFC Rutherford Appleton Laboratory

CHEP 2015

Okinawa, Japan

Science & Technology Facilities Council

Outline

- Introduction, aims & overview
- How it works
  - Creating VMs
  - Implementation of “back off”
  - VM lifecycle
  - Target shares & accounting
  - Traceability
- Some results
- Conclusion

Introduction

- Traditional way for VOs to run work at grid sites
  - Experiments submit pilot jobs to CEs
  - CEs submit the pilot jobs to the local batch system
  - Pilot jobs run on the batch system, launch pilot framework
  - Pilot framework pulls down payload jobs
- Alternative is the vacuum model
  - Sites automatically create VMs
    - No CEs required, no BDII required, …
  - VMs contextualized for each required experiment
    - Contextualization provided by experiments
  - VMs launch the pilot framework
  - Pilot framework pulls down the payload jobs

Introduction

- Implementations of the vacuum model
  - Vac *[resources dedicated to the vacuum model]*
    - Machines setup as hypervisors running the Vac software
  - Vcycle *[existing cloud resources]*
    - Works with clouds, e.g. OpenStack
    - Service instantiates VMs for each experiment
- What about an existing batch system?
  - Can we use ideas of the vacuum model with an existing batch system?
  - Make use of existing batch resources for both:
    - Traditional grid jobs (running directly on the physical worker nodes)
    - Jobs run in VMs using the vacuum model
  - Avoids static partitioning, e.g. batch + Vac
- HTCondor has a “VM universe”
  - Jobs can be VMs

Aims

- Consistency with Vac/Vcycle
  - Should work with existing experiment user data created for Vac/Vcycle without modification
  - Should have similar features, e.g. “back off”, caching of images, …
- Use existing features of HTCondor as much as possible
  - Job hooks, job router daemon, file transfer plugins, condor_chirp, …
- No significant changes to worker nodes
  - But some changes unavoidable
    - Libvirt installed, libvirtd running
    - Some additional HTCondor configuration & scripts run as job hooks
      - Easy to deploy via Quattor, Puppet, etc
- “bare metal” batch jobs & VMs on the same machines
  - Resource usage of each can be limited by cgroups
    - E.g. CPU, memory

Overview

- Additional HTCondor configuration added to a machine running a schedd
  - Jobs (VMs) created here
- Single configuration file for the vacuum
  - Specifies configuration for each vmtype
    - Usually one vmtype per experiment
  - User data obtained from a URL provided by each experiment
  - Image can be a local file on the schedd or a URL
- VMs are created regularly for each vmtype
  - When there is no work or failures for VMs of a particular vmtype, not many VMs are created
  - When there is work, more VMs are created

Configuration

- Uses a config file almost identical to Vac

```
[vmtype atlas]
user_data_option_queue = RAL-LCG2_VAC
user_data_option_default_se = srm-atlas.gridpp.rl.ac.uk
user_data_option_cvmfs_proxy = http://squid04.gridpp.rl.ac.uk:3128
user_data_file_hostcert = /scratch/Vac/ATLAS/hostcert.pem
user_data_file_hostkey = /scratch/Vac/ATLAS/hostkey.pem
user_data = https://www.gridpp.ac.uk/vac/atlas/user_data
vm_model = cernvm3
root_image = https://www.gridpp.ac.uk/vac/atlas/cernvm3.iso
rootpublickey = /scratch/Vac/root.pub
heartbeat_file = vm-heartbeat
heartbeat_seconds = 600
max_wallclock_seconds = 172800
log_machineoutputs = True
accounting_fqan = /atlas/Role=NULL/Capability=NULL
htcondor_cpus = 1
htcondor_memory = 3200
htcondor_failure_rate_threshold = 0.001
htcondor_accounting_group = group_ATLAS.prodatls
```

Creating VMs

- HTCondor has a feature to allow worker nodes to pull work rather than to have work pushed to them
  - Fetch work hooks
- Limitations
  - Cannot be used with VM universe jobs
  - Since the negotiator isn’t involved in deciding what jobs to run, fairshares won’t be respected
- Alternative
  - Simple script which submits jobs using HTCondor Python API
  - Maintains job pressure, always *n* idle jobs, for each vmtype

Implementation of “back off”

- Don’t want to create VMs constantly
  - Wastes resources
  - Could overload experiment central task queues
- “Back off”
  - If no work, jobs failing, or site misconfigured, wait before running more VMs
- Make use of the Job Router daemon
  - From the manual:

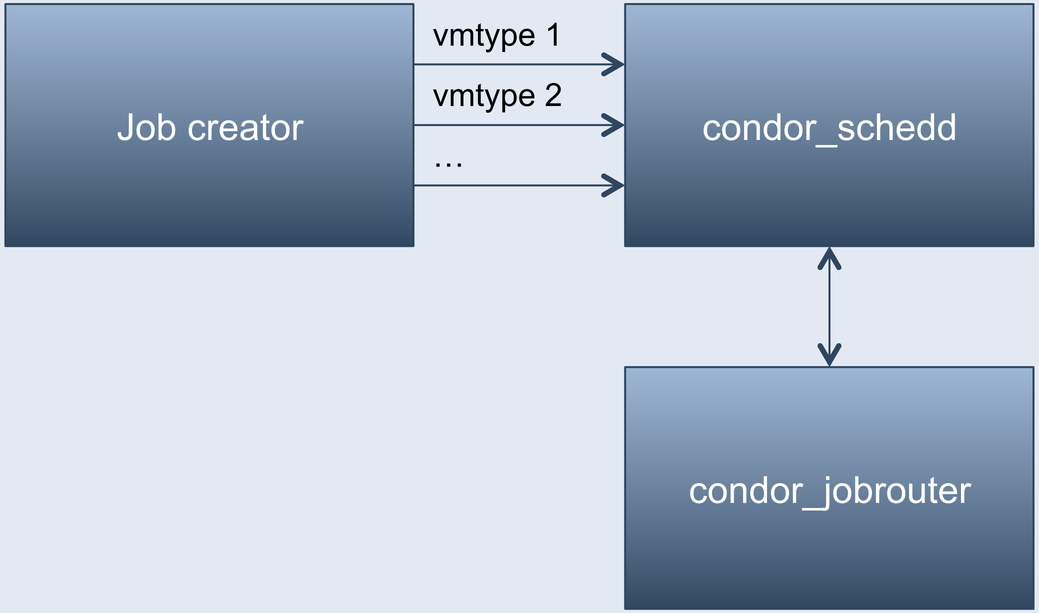
    “The HTCondor job router is an add-on to the condor_schedd that transforms jobs from one type into another according to a configurable policy”
  - Has a built-in throttle
    - Usually used to prevent sending grid jobs to bad sites
    - Definition of failure is configurable
  - Provided information about status of VMs (shutdown code) is put into job ClassAds, can use job router to implement “back off”

Implementation of “back off”

- How it works
  - The jobs created are configured so that they *can’t run*
    - Requirements = false
  - Job router
    - Sets Requirements such that jobs can run
    - Job router therefore is responsible for determining when VMs can run
  - Has a built-in throttle for failing jobs
    - Set expression used to determined whether a job failed to depend on the VM’s shutdown code
    - If VMs don’t have any work or fail, this is regarded as failure
    - FailureRateThreshold defines the maximum tolerated rate of job failures
  - 1 route per vmtype
    - Routes can be generated automatically from vacuum config file
    - Makes use of JobRouter ability to run an arbitrary script to dynamically generate routes

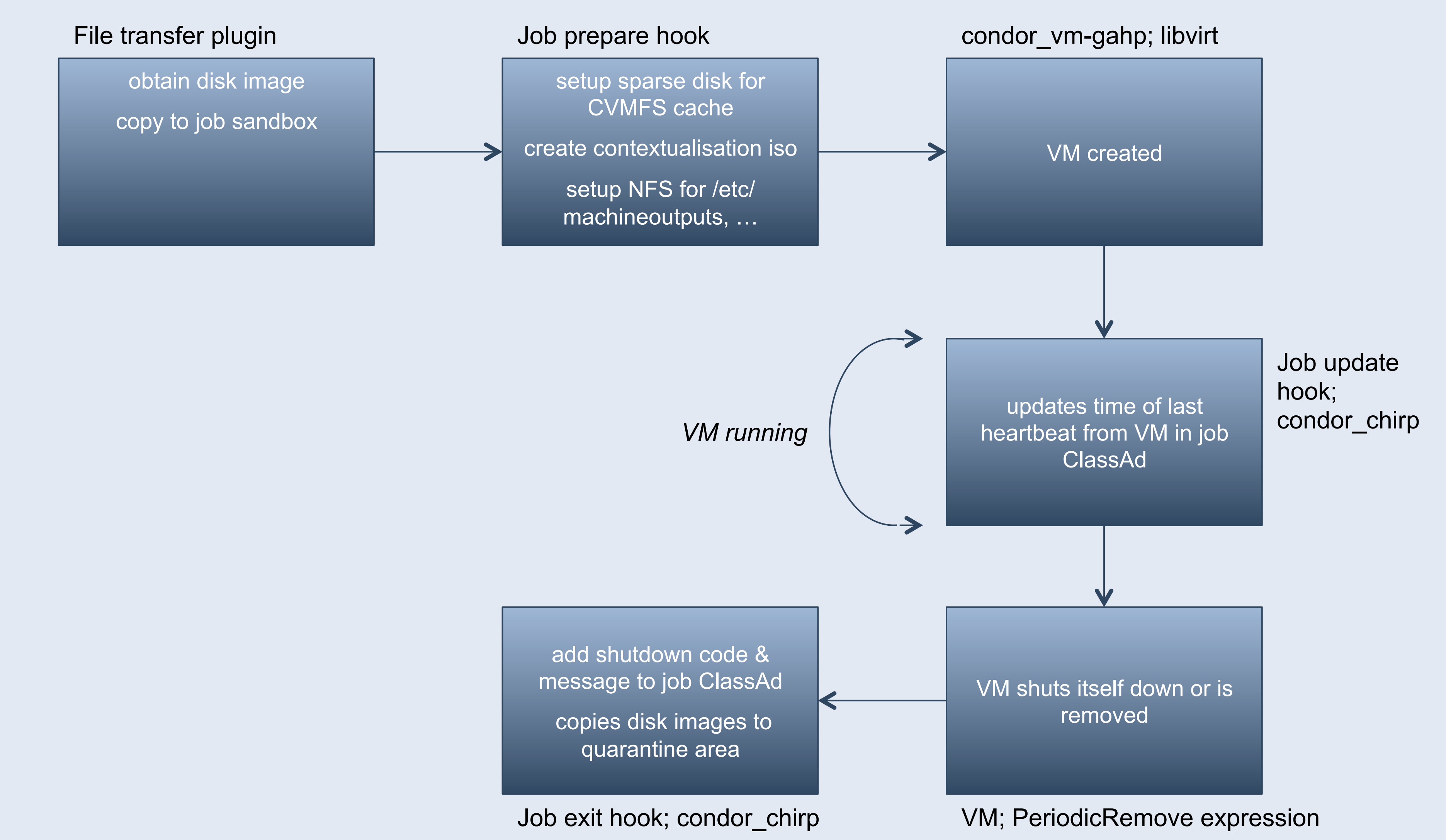
Creating VMs

Jobs created for each VO



Ensures VMs are not created if there are failures or there is no work

VM lifecycle



File transfer plugin

obtain disk image

copy to job sandbox

Job prepare hook

setup sparse disk for CVMFS cache

create contextualisation iso

setup NFS for /etc/machineoutputs, …

condor_vm-gahp; libvirt

VM created

VM running

updates time of last heartbeat from VM in job ClassAd

Job update hook; condor_chirp

VM shuts itself down or is removed

VM; PeriodicRemove expression

add shutdown code & message to job ClassAd

copies disk images to quarantine area

Job exit hook; condor_chirp

GridPP
UK Computing for Particle Physics

Target shares & accounting

- VMs are no different to any other jobs from HTCondor's point of view
  - Hierarchical group quotas configured on central manager node(s)
  - Accounting group for each vmtype is specified in the vacuum config file
    - Could have traditional grid jobs and vacuum VMs in the same accounting group
    - Or could have separate accounting groups for vacuum VMs
- Accounting data sent directly to APEL central service
  - APEL accounting records generated directly from information in the standard condor history files
  - Sent to APEL using ssmsend (like ARC CE, APEL publisher node)

Traceability

- Central logging
  - rsyslog.conf in the VMs is updated to contain information about site's central loggers
    - Done as part of contextualization, independent of VO
  - Central logging starts before any of the VO scripts are run
- Quarantining of disk images
  - Want to keep disk images for a specified time period
  - Enables short-lived VMs to be investigated later if necessary
  - After a VM is shutdown, disk images are copied to a quarantine area on the worker node
    - Handled by job exit hook

Monitoring

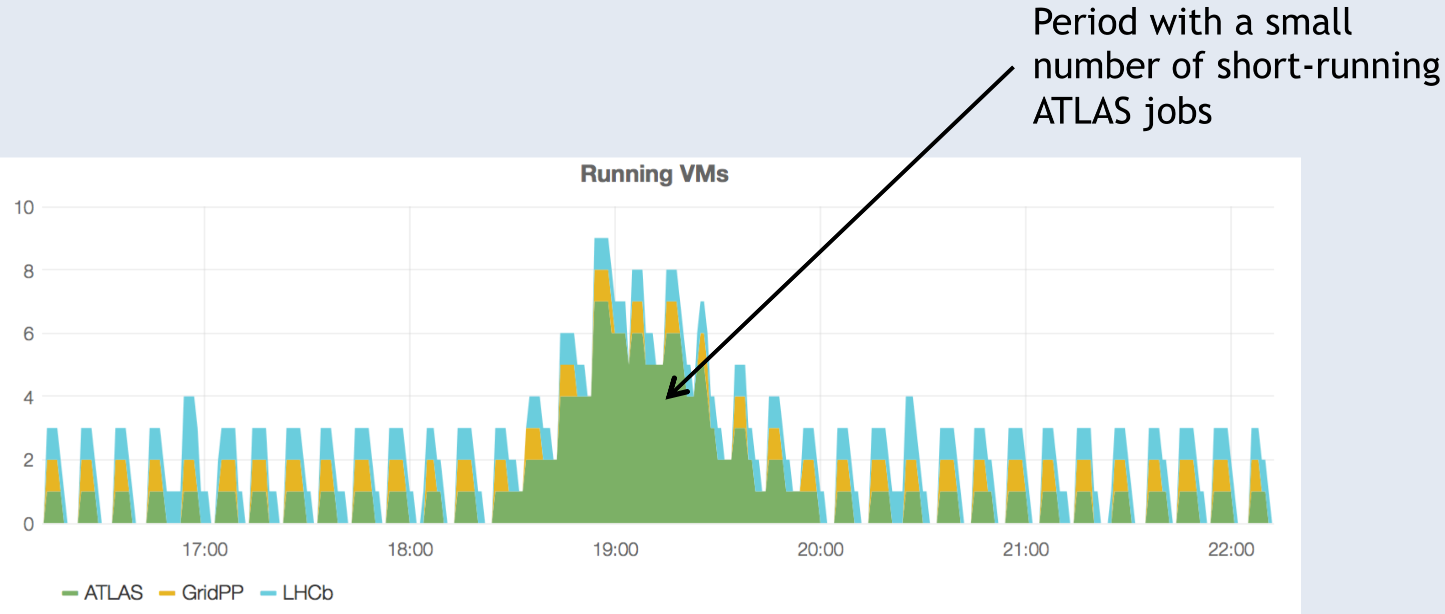
- condor_q with a custom print format to show status of VMs

```
-bash-4.1$ condor_q -pr vacuum.cpf

id          vmtype      Start date    Run time     Status   ShutdownCode/message
83092.0     atlas       3/25 04:53    0+04:24:26   R
83094.0     atlas       3/25 04:54    0+04:23:36   R
83097.0     atlas       3/25 04:56    0+04:21:57   R
83189.0     cms         3/25 08:09    0+00:13:10   C        200 Success
83190.0     atlas       3/25 08:09    0+00:39:06   C        200 Success
83191.0     atlas       3/25 08:11    0+00:25:27   C        200 Success
83201.0     cms         3/25 08:19    0+00:10:45   C        300 Nothing to do
83202.0     atlas       3/25 08:20    0+00:19:08   C        200 Success
83203.0     cms         3/25 08:20    0+00:09:46   C        300 Nothing to do
83241.0     gridpp      3/25 08:53    0+00:03:23   C        300 Nothing to do
```

In use

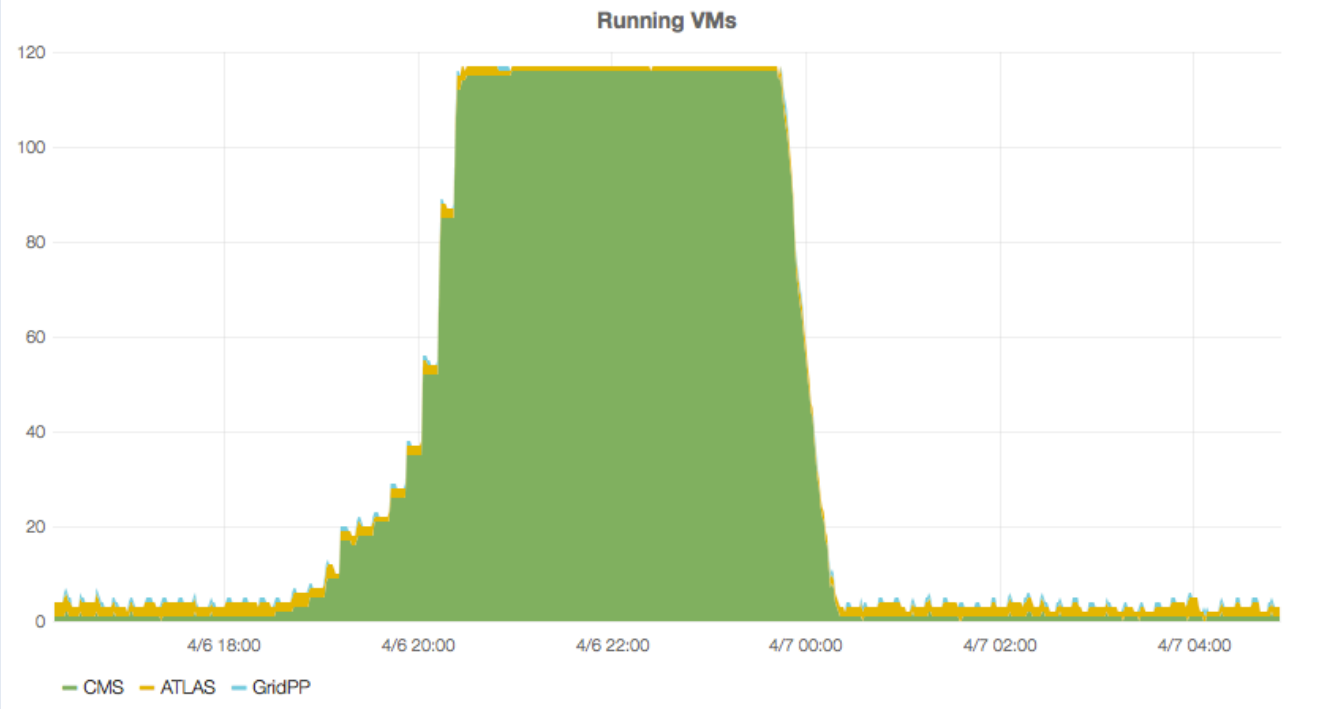
- When there is no work, VMs of each type are created regularly



Period with a small number of short-running ATLAS jobs

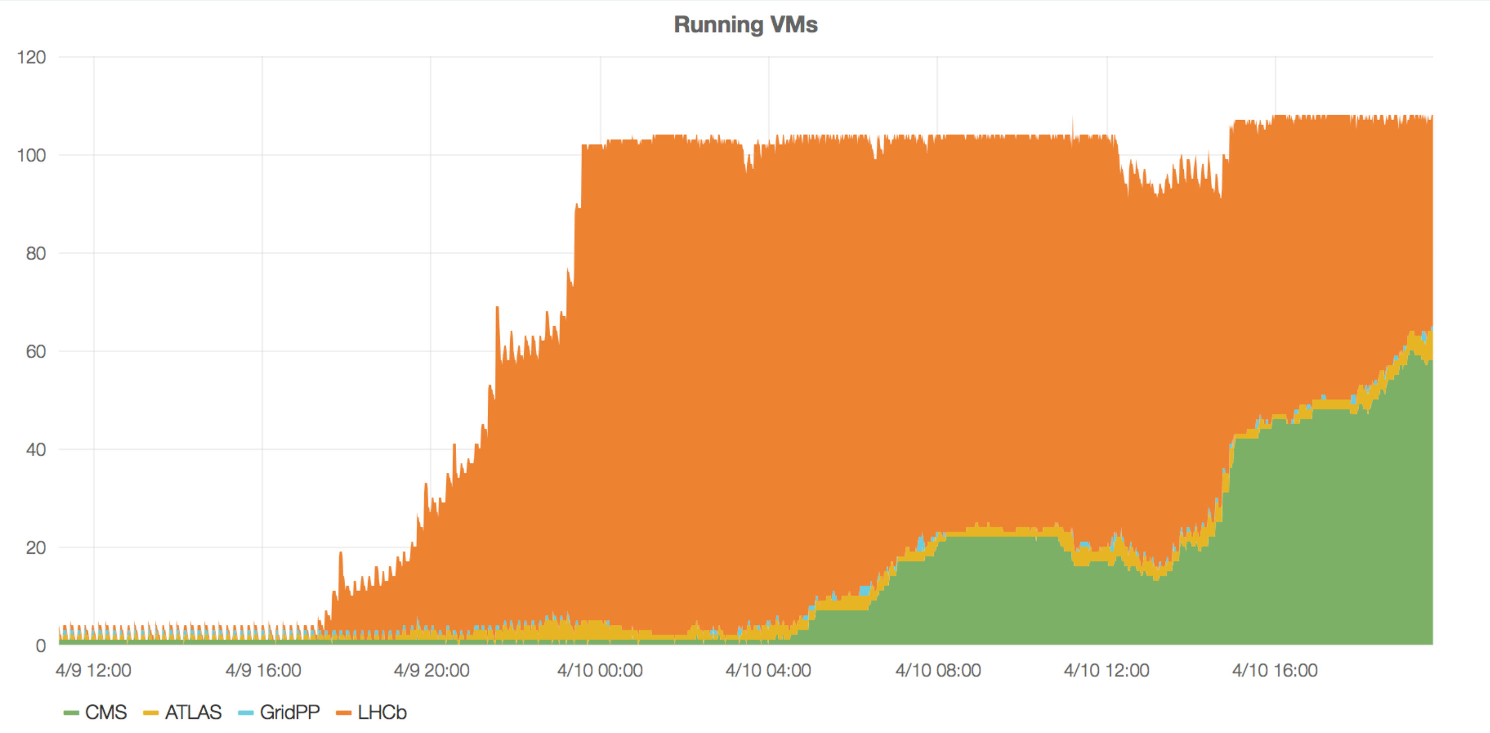
In use

- When there is work for a VO, the number of running VMs increases
- As the available work is completed, the number of running VMs decreases



In use

- Multiple VOs running work
  - Fairshares are handled in the usual way
    - Negotiator decides what jobs (VMs) to run



Conclusion

- Have demonstrated an implementation of the vacuum model using HTCondor
  - Almost all functionality derived from standard HTCondor features
- Future outlook
  - Today VMs are a common way for experiments to run jobs at different sites in a standard environment
    - Sites don’t need to install lots of software
  - But in a batch system, can already have standard grid worker nodes
    - Could have a vacuum model implementation without virtualization
  - Also, there is growing interest in containers, in particular Docker
    - Benefits include
      - No virtualization overheads
      - Faster startup times
  - Soon HTCondor will have a “Docker universe”
    - HTCondor vacuum model could easily be extended to use containers instead of (or as well as) VMs

Questions?