Vac and Vcycle

Andrew McNab

University of Manchester
LHCb and GridPP

Overview

- Vac and Vcycle
- Remote user_data and images
- Vcycle refactoring
- Target share convergence
- APEL accounting
- Syslog logging from VMs
- Multiprocessor VMs
- SAM tests
- Next steps

Vac and Vcycle

- Both VM Lifecycle Managers
- Vac is a standalone daemon you run on each worker node machine ("VM factory") to create its VMs
- Vcycle manages VMs on IaaS Clouds like OpenStack
  - Can be run at the site, by the experiment, or by regional groups like GridPP
- Both developed at Manchester as part of GridPP Clouds/VMs effort
  - With help from Lancaster, Oxford, IC, CERN, Birmingham, LHCb, ATLAS, CMS
- Both make very similar assumptions about how the VMs behave
  - Pilot VMs: use cvmfs and configure CernVM as WLCG WN, then run pilot client
  - The same LHCb, ATLAS, CMS VMs working in production on Vac and Vcycle
  - Compatible GridPP DIRAC VMs available too
- Talks about Vac, Vcycle, Pilot VMs and WLCG at CernVM Workshop last week
  - So I'll just concentrate on updates in this talk

Vacuum model

- For the experiments, VMs appear by “spontaneous production in the vacuum”
  - Like virtual particles in the physical vacuum: they appear, potentially interact, and then disappear
- Following the CHEP 2013 paper:
  - *“The Vacuum model can be defined as a scenario in which virtual machines are created and contextualized for experiments by the resource provider itself. The contextualization procedures are supplied in advance by the experiments and launch clients within the virtual machines to obtain work from the experiments' central queue of tasks.”*
- At many sites, 90% of the work is done by 2 or 3 experiments
  - So a simple, reliable way of running their “baseload” of jobs is worthwhile
- CernVM-FS and pilots mean a small user_data file is all the site needs

Remote user_data templates and images

- Resource provider can create user_data file, or let Vac/Vcycle do it from a template.
- Templates can now be URLs on experiment's web server
  - Refetched each time since very small
- Experiment can also supply desired uCernVM 3 boot images
  - Since so small (20MB) again can be specified as URLs
  - Cached and rechecked each time with If-Modified-Since
  - Vcycle uploads new VM images to OpenStack automatically
- Experiments can update user_data and images everywhere without having to contact site admins
- Easy to add new experiment to a Vac or Vcycle installation

Example of configuration

- Section of vac.conf used to enable LHCb VMs at Manchester
- They just need this and to create a hostcert/key.pem
- (vcycle.conf configuration is very similar)
- 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
```

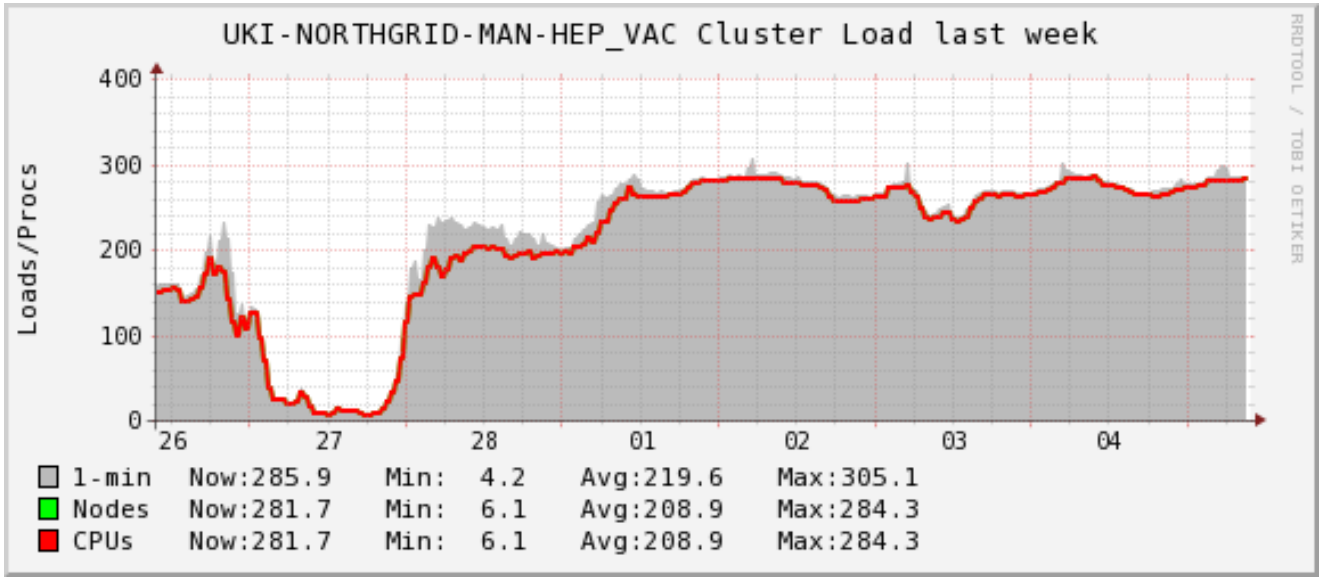
Vcycle refactoring

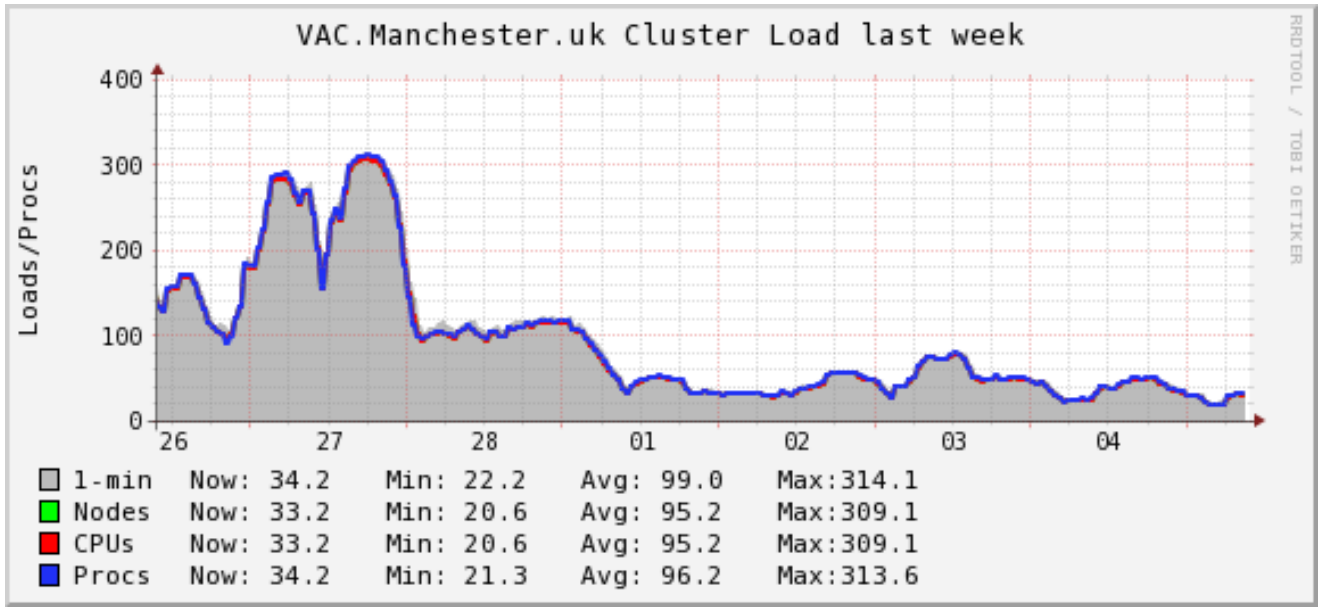
- In December/January, Vcycle was largely rewritten
  - Code redone as objects rather than lists of dictionaries
  - Aim was to support backend plugins for OCCI, EC2, …
  - Default OpenStack plugin written using REST API rather than nova library (both as easy to use)
- Took opportunity to converge option names, algorithms etc with Vac
- Shared vacutils.py for common Vac/Vcycle code
  - Remote user_data and images first fruit of this
  - Also ability to generate and insert legacy or RFC GSI proxies into VMs instead of hostcert/keys
- ssh keypairs also managed by Vcycle now
- man pages done; Admin Guide started; but Vcycle code still quite fluid
  - Usable by other sites, but would need our help still

Existing target shares with Vac

- When a VM slot becomes available, the node decides how to fill it.
- The other Vac nodes are queried via UDP to discover what they are running, in units of HS06
- The node bases decision on its list of target shares for each VM type
  - The VM type with the least slots, weighted by target share, gets the free slot
  - Uses a site-wide back-off procedure to veto VM types that don't have any work
- This approach is very simple, means factory nodes are autonomous
  - Avoids a central management daemon which would be a single point of failure
- The target shares are instantaneous
  - They are fair, in that if all experiments submit lots of jobs, the site shares out the capacity according to the stated shares
  - But quiet periods aren't credited and carried forward, so may need to adjust targetshares to achieve annual shares, say (as many batch sites do...)

Target shares: ATLAS vs LHCb (vs CMS)





Target shares now the same in Vcycle

- When a VM slot becomes available, Vcycle decides how to fill it
- Go through each VM type seeing if it is suitable to be created
- Limits and shares for each VM type are taken into account
  - Uses a site-wide back-off procedure to veto VM types that don't have any work
  - Can mix VMs for different experiments in the same Project/Tenancy
  - Target shares work as with Vac. Counts in units of HS06/VM.
- Target shares/limits are at the level of Projects/Tenancies
  - Vcycle instances can support multiple tenancies, perhaps at multiple remote sites
  - Different target share pools in the different tenancies
  - eg Vcycle instance at Manchester manages ATLAS, CMS, and LHCb VMs at IC and CERN separately.
- The target share model is instantaneous, as with Vac

Accounting with Vac and Vcycle

- Many sites need to report usage through APEL and EGI Accounting
- Well established for conventional gatekeeper+batch grid sites
  - There is also an EGI Cloud accounting activity
- Vac and Vcycle write out APEL SSM record into a directory when each VM finishes
- These can be sent to the central APEL service by the standard APEL ssmsend tool (as ARC does)
- For Vcycle, this provides an alternative to installing accounting reporting tools for OpenStack etc at the site
- For Vac, it makes the accounting at the site completely self-contained at the level of the individual physical machine
  - No longer need to depend on a site APEL service

Mapping VMs to APEL records

- Vac and Vcycle write out APEL records ready for ssmsend
- Site admin can give the “GOCDB” name in the configuration, included in the messages
  - UKI-NORTHGRID-MAN-HEP
- The space name and vmtype are used to construct the SubmitHost
  - vac02.tier2.hep.manchester.ac.uk/vac-atlasprod
- The VM UUID is used as the LocalJobId
- The space name is converted into a “user” DN
  - /DC=uk/DC=ac/DC=manchester/DC=hep/DC=tier2/DC=vac02
- The FQAN for the experiment is included if given in configuration
  - /lhcb/Role=NULL/Capability=NULL
- The net result is the records end up associated with the site in exactly the same way as records from other conventional grid/batch CEs at the site

Syslog prototype: machinefeatures

- Done as part of WLCG Traceability TF
- Added to LHCb VMs
- If /etc/machinefeatures/syslog exists, then it is used to create configuration file /etc/rsyslog.d/vm.conf in VM to send a copy of syslog messages to an external rsyslogd
  - Vac can be configured to add custom values to machinefeatures like this
- Current tests with rsyslogd on each factory machine
  - Site can then store messages on factory for X days, filter and forward to a central service, etc. Entirely their choice.
  - Simplest security model: over internal NAT net within the factory
- UDP syslog protocol probably a good choice as universally supported now and very easy to gateway to other protocols in the future

Syslog prototype: contextualization

- Machinefeatures has the advantage of being VM-architecture neutral
- However, script to access it is only run at the end of the boot up
- To get earlier messages, need vm.conf file created at the start
- To address this for us, there is now a prototype with the ability to supply files to the uCernVM configuration
  - Set of files supplied as base64-encoded blob
- Looking at how best to provide this:
  - Site creates the required base64 blob? (Always the same for Vac since always 169.254.169.254:514)
  - Or make Vac/Vcycle aware of syslog with a syslog_server option and do it all automagically for the site admin?

Multiprocessor VMs in Vac

- These are supported by Vac as a parameter in the node's configuration file
- Each factory node has one current value in force
- Can be changed at any time, and new VMs will be created using it
- Vac keeps track of existing VMs' geometry to avoid overcommitting
- VM processor count taken into account by target shares mechanism
- This system is designed to allow dynamic repartitioning of a Vac space into, say, single processor and 8-processor VM subspaces
  - Just update node's configuration parameter in Puppet etc and wait
- This parallels the dynamic partitioning models being evaluated by the WLCG Multicore Task Force for conventional batch systems
- Vcycle has HEPSPEC06 per VM parameter which is the basis of target shares, and implicitly takes multiprocessor weighting into account

SAM tests

- This isn't Vac or Vcycle specific but it's important for WLCG…

- WLCG SAM tests are jobs submitted to sites via CREAM or ARC CE by the experiments
- This model isn't possible with the Pilot VMs since they are started by the site rather than the experiment
- Solution needed for VM-only sites otherwise they will not be able to demonstrate their availability and reliability for WLCG pledges
  - Also important for identifying problems too of course …
- Looking at running SAM tests in the VM as part of initialisation
  - Perhaps just run it a fraction of the time
  - Reporting via messages as with conventional SAM tests

Next steps

- New VacQuery UDP protocol
  - Scalable to factories with very many VMs and many more vmtypes
  - Resilient to high packet loss
  - Once done, we will release Vac 1.0.0
- CentOS 7 + Python 2.7 port/validation of Vac
- Features for overnight use of Vac (eg on interactive machines)
- OCCI (and EC2) plugins for Vcycle
- Multiprocessor VM tests
  - Needs multiprocessor VMs
- Container support in Vac
  - Probably using libvirt API again

Summary

- Vac provides a simple way for sites to run VMs
- Vcycle provides a simple way to manage VMs on OpenStack
- Continuing development but “Simplicity is a feature” too
  - Remote user_data templates and boot images since last time
  - Working on multiprocessor, OCCI, CentOS 7, containers
- For Vac http://www.gridpp.ac.uk/vac/ for RPMs, Yum repo, links to GitHub, docs, man pages etc
- For Vcycle http://www.gridpp.ac.uk/vcycle/ has links to source, man pages etc

- See CernVM workshop Thursday morning and afternoon talks also:
  - https://indico.cern.ch/event/348657/other-view