Journal of Physics: Conference Series **664** (2015) 022030 doi:10.1088/1742-6596/664/2/022030

LHCb experience with running jobs in virtual machines

A. McNab[1], F. Stagni[2], and C. Luzzi[2]

[1] School of Physics and Astronomy, University of Manchester, UK
[2] CERN, Switzerland

E-mail: andrew.mcnab@cern.ch

Abstract. The LHCb experiment has been running production jobs in virtual machines since 2013 as part of its DIRAC-based infrastructure. We describe the architecture of these virtual machines and the steps taken to replicate the WLCG worker node environment expected by user and production jobs. This relies on the uCernVM system for providing root images for virtual machines. We use the CernVM-FS distributed filesystem to supply the root partition files, the LHCb software stack, and the bootstrapping scripts necessary to configure the virtual machines for us. Using this approach, we have been able to minimise the amount of contextualisation which must be provided by the virtual machine managers. We explain the process by which the virtual machine is able to receive payload jobs submitted to DIRAC by users and production managers, and how this differs from payloads executed within conventional DIRAC pilot jobs on batch queue based sites. We describe our operational experiences in running production on VM based sites managed using Vcycle/OpenStack, Vac, and HTCondor Vacuum. Finally we show how our use of these resources is monitored using Ganglia and DIRAC.

1. Introduction

A previous paper[1] in 2014 described the LHCb collaboration's initial design of and experiences with running jobs in virtual machines at Infrastructure-as-a-Service (IaaS) cloud sites. In this paper we describe a new design deployed in 2014 using the uCernVM[2] model, which is our current production implementation across all VM-based platforms.

We will set out how VMs are deployed by resource providers, explain the VM architecture itself including its interaction with LHCb DIRAC[3], and then describe our experiences.

2. Deploying VMs on Vacuum resources

The LHCb VMs follow the Pilot VM format and are designed to be used within the Vacuum model[4]:

> *The Vacuum model can be defined as a scenario in which virtual machines are created and contextualized for experiments by the resource provider. 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.*

We support Vac, Vcycle[5], and HTCondor Vacuum[6] implementations of vacuum-style VM lifecycle manager, all of which give the resource provider direct control of the mixture of VMs

run for different experiments. Instructions on how to write the LHCb section of the configuration file are provided to sites on a web page[1] and typically result in short additions to those files. This example shows the entire LHCb configuration used in production on the Vac-based site at the University of Manchester:

```
[vmtype lhcbprod]
vm_model = cernvm3
root_image =
https://lhcbproject.web.cern.ch/lhcbproject/Operations/VM/cernvm3.iso
root_public_key = /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
```

Some of the settings, such as the URL of the Squid caching proxy used by CernVM-FS[7], are provided by the site. The URLs of the root_image and user_data file are provided centrally by LHCb.

The root_image is an unmodified uCernVM boot image for use with the uCernVM system, which is fetched and cached by the VM lifecycle manager, and updated based on observed changes to the file's modification time. This mechanism allows LHCb to control when new versions of the boot image are used, and to update the image in use at all sites after suitable checks.

The user_data file is a template which the VM lifecycle managers fetch and customise using some standard substitutions and additional ones included in the user_data_option_xxxx and user_data_file_xxxx settings given. Again, this allows LHCb to update the file centrally and have it used immediately by all sites for all subsequent VM creations. The user_data file controls contextualization of the virtual machine once booted, and includes settings and a short script which causes the setting up of the LHCb VM architecture and the insertion of an X.509 certificate and private key in the VM filesystem.

3. VM architecture

Figure 1 shows the components of the LHCb VMs. In the new design of the LHCb VMs, three Unix accounts within the VM are assigned different roles and are used to control access to credentials.

The initial user_data script is run as root, which in turn runs a vm-bootstrap script downloaded as part of a set of contextualization files. This script sets up the operating system level environment. This includes

- Creating the script /etc/acpi/actions/power.sh to produce a shutdown message if the ACPI shutdown is triggered by the hypervisor.

[1] https://twiki.cern.ch/twiki/bin/view/LHCb/VacConfiguration

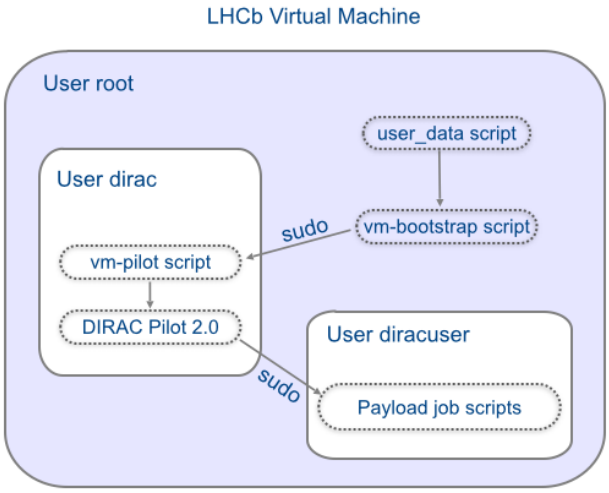

Figure 1. Architecture of the LHCb VMs

- Reading the machinefeatures and jobfeatures URLs from the HTTP metadata service which appears on 169.254.169.254 on IaaS platforms.
- Creating an iptables firewall rule to block HTTP access to 169.254.169.254 . This is to stop subsequent user jobs from accessing the user_data file which is avalable there on IaaS systems.
- Configuring the rsyslogd daemon to report log file messages to the resource provider if this has been requested.
- Creating and mounting a fast filesystem at /scratch if available
- Disabling the disk I/O scheduler within the VM
- Putting the CernVM-FS cache on the fast /scratch partition
- Creating a cron job to update the heartbeat file every 5 minutes
- Creating a 4GB swap file on the /scratch partition
- Setting the Linux "swappiness" value to minimise swapping
- Installing Ganglia[8] gmond version 3.4.0 and configuring it to report to the CERN-operated Ganglia service
- Creating the dirac and diracuser Unix accounts
- Symbolically linking /cvmfs/lhcb.cern.ch/etc/grid-security to /etc/grid-security to provide the certificate authority files directory.

vm-bootstrap also copies the X.509 host certifcate and private key or proxy into the location expected by DIRAC, and then runs the vm-pilot script as dirac using the sudo command.

Once vm-pilot returns, vm-bootstrap parses the vm-pilot.log file to determine which shutdown message to pass to the VM lifecycle manager. Finally, vm-bootstrap has the responsibility for shutting the VM down, first with the shutdown command to perform an orderly shut down, and if that fails using the Linux sysrq-trigger mechanism to force an instant halt of the VM.

4. DIRAC pilot

The vm-pilot script runs as the dirac Unix account and parallels the steps taken by LHCb DIRAC pilot jobs running at conventional grid sites.

Before running the DIRAC pilot framework client, the script constructs a Job ID from the space name, the VM type, and the UUID of the VM supplied by the VM lifecycle manager. This Job ID is included by DIRAC in its logging system for payload jobs and is normally the Job ID within a conventional batch system. We have found the creation of such a reference to be very useful for correlating VM instances at all levels and DIRAC jobs. The environment variables GLITE_LOCATION and VO_LHCB_SW_DIR, which are conventially set by the scripts installed on grid worker nodes, are also set.

vm-pilot then fetches the initial DIRAC pilot client and runs it with appropriate parameters. The credential used for contacting the central LHCb tasks queue at CERN is the X.509 certificate and private key or proxy supplied by the VM lifecycle manager. However, the vm-pilot script instructs the pilot client to run the payload job it downloads using the diracuser Unix account which does not have access to these privileged credentials. Instead, DIRAC's ProxyManager service provides the payload job with credentials derived from those of the user submitting the job.

5. Shutdown codes and messages

When the VM terminates, vm-bootstrap runs a script, ParseJobAgentLog, which determines why the VM has finished. The script outputs a message which is passed to the VM lifecycle manager based on this information. The message consists of a three digit code followed by human readable text, in a similar way to status messages in internet protocols such as HTTP[9] and SMTP[10]. The values are listed in Table 1.

Table 1. Shutdown codes and messages

100 Shutdown as requested by the VM's host/hypervisor
200 Intended work completed ok
300 No more work available from task queue
400 Site/host/VM is currently banned/disabled from receiving more work
500 Problem detected with environment/VM provided by the site
600 Grid-wide problem with job agent or application within VM
700 Transient problem with job agent or application within VM

As with HTTP codes, this scheme provides room to insert more numbers for finer grained information in the future. Currently the LHCb VMs only use the basic codes which are multiples of 100, and only identify certain outcomes within the wider categories listed in the table.

If the hypervisor sends a simulated hardware shutdown signal, then code 100 is returned. Otherwise, the ParseJobAgentLog script identifies successful completion of the intended work (code 200), that no work could be found (300), that a malformed certificate was passed in the environment given to the VM (500), or that an error was encountered communicating with the central matcher / task queue service suggesting a grid-wide problem running LHCb DIRAC jobs (600). For all other errors, the generic message "700 Failed, probably JobAgent or Application problem" is returned as these are typically transient problems encountered by the agents or applications within the VM.

6. Deployment at sites

The LHCb VMs have been deployed in production at the sites listed in Table 2.

Table 2. LHCb use of VM-enabled sites

CERN	Vcycle & OpenStack
CC-IN2P3	Vcycle & OpenStack
Imperial College	Vcycle & OpenStack
Birmingham	Vac
Lancaster	Vac
Manchester	Vac
Oxford	Vac
University College	Vac
STFC RAL	HTCondor Vacuum

The OpenStack-based[11] capacity at CERN and CC-IN2P3 is managed by a Vcycle instance running on one of the LHCb central "VO Box" machines at CERN. The OpenStack capacity at Imperial College, London is managed by the GridPP Vcycle instance at the University of Manchester.

7. Monitoring and accounting

The VM instances are monitored by a combination of the lgm.cern.ch Ganglia service provided by CERN and LHCb DIRAC's standard monitoring and accounting system used for jobs running at conventional sites.

Figure 2 shows a web page view of the Ganglia monitoring for the LHCb tenancy of the CERN OpenStack site, showing plots of the CPU, memory, and network activity over a designated time range.

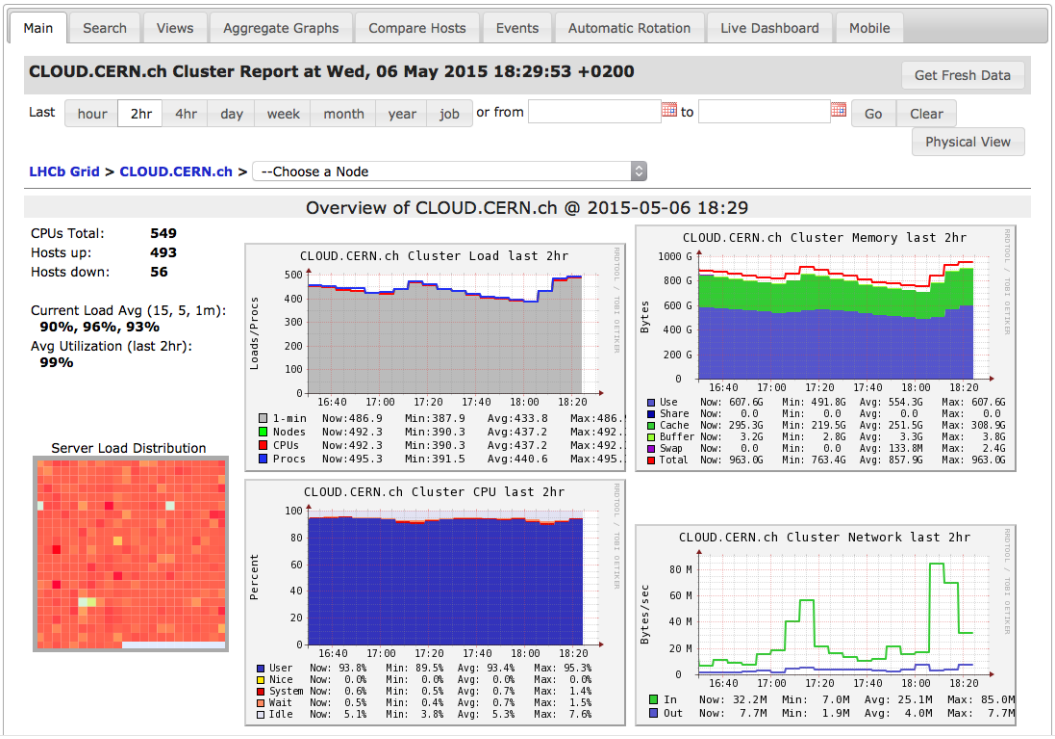

Figure 2. Ganglia monitoring of LHCb VMs at CERN

Figure 3 shows plots of the CPU time delivered by the VM-based sites between 1st May 2014 and and 1st May 2015, using the DIRAC accounting system. After initially starting at a modest level, the number of concurrent VMs is now around 1000 which is comparable to the number of job slots supplied to LHCb by large Tier-2 sites.

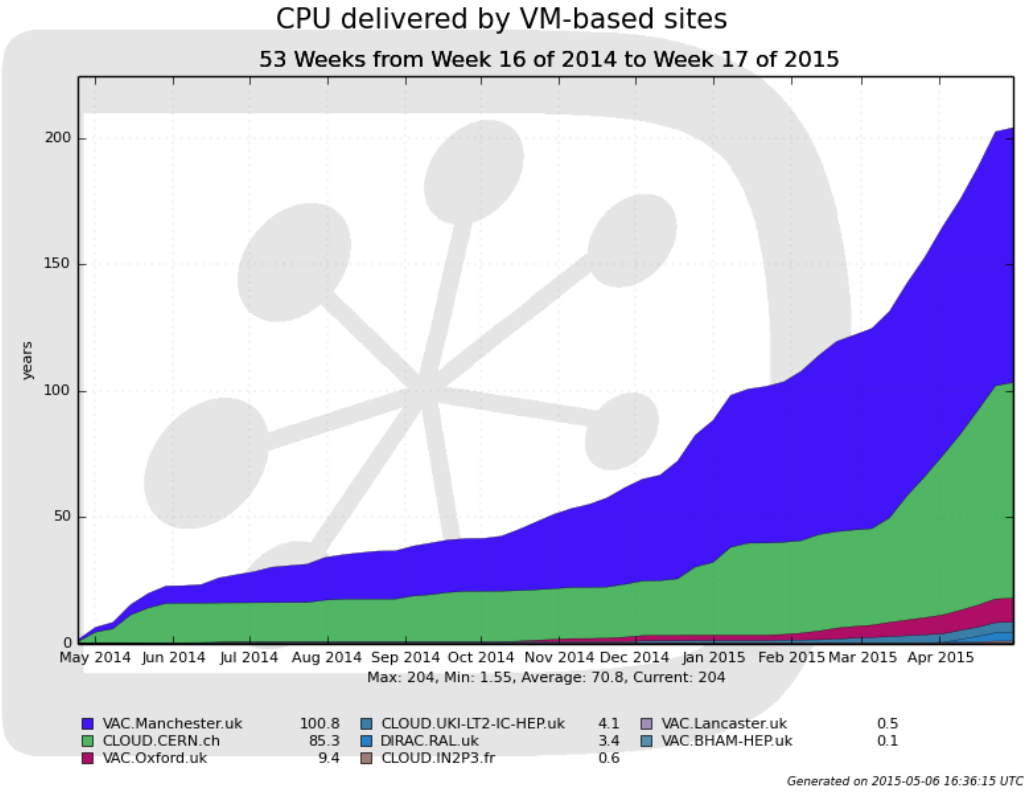

Figure 3. LHCb job execution rate at VM-based sites

8. Operational experience

In adding sites to our virtualized infrastructure we encountered several issues on more than one occasion:

- Before the VM lifecycle managers supported remote user_data templates the resource provider had to use a script to create the user_data file. Despite designing a minimal user_data file we discovered that changes were still often needed which would have required a manual "broadcast" channel to request updates to the file by resource providers.
- At OpenStack sites managed by Vcycle, the original pattern was to manually upload the uCernVM boot image to the OpenStack image service. As with user_data templates, the ability to have this managed automatically eases the deployment to new sites and of new versions of the boot image. We believe that the absence of an automated solution would limit the scalability of the infrastructure.
- During the commissioning of significantly new types of site we find it is extremely convenient to have ssh command line access within the VMs. However, all of the scenarios we have observed (access control on Squid proxy caches, firewalls, insufficient disk space) could have been identified with retrospective access to the VM lifecycle manager log file.

- If the site does not have an unusual configuration, then it can be extremely quick to add to our virtual infrastructure, with human effort measured in minutes to add a new section to the local and central LHCb configuration files. A new site can be brought into production within hours including running test jobs to ensure the configuration is correct.

9. Other experiments

The LHCb VM architecture has also been the basis for the ATLAS and GridPP DIRAC VMs developed at the University of Manchester, and for the CMS VMs developed at STFC Rutherford Appleton Laboratory. All four flavours of VM are fully interoperable and can be run by any of the three implementations of VM lifecycle manager which use the Vacuum model. This commonality between experiments avoids inconvenient and unnecessary additional complexity for people operating sites, but also has the significant advantage that VMs from multiple experiments can be run in the same tenancy or space to facilitate load balancing if one or more of them has reduced requirements which the others can exploit to avoid resources sitting idle.

10. Conclusion

We have presented a description of the virtual machines used by the LHCb experiment at nine sites. These VMs can be run by any of the three vacuum-based systems, Vac, Vcycle/OpenStack, and HTCondor Vacuum. We have explained that these VMs have been used in production for the past year and how they have contributed at the scale of 1000 job slots or a large Tier-2 site.

References

[1] Mario Ubeda Garcia et al 2014 *J. Phys.: Conf. Ser.* **513** 032099
[2] J. Blomer et al. 2014 *J. Phys.: Conf. Ser.* **513** 032007
[3] F Stagni et al 2012 *J. Phys.: Conf. Ser.* **396** 032104
[4] A. McNab et al 2014 *J. Phys.: Conf. Ser.* **513** 032065
[5] A.McNab et al 2015, "Managing virtual machines with Vac and Vcycle", presented at the CHEP 2015 conference
[6] A. Lahiff 2015, "Implementation of the vacuum model using HTCondor", presented at the CHEP 2015 conference
[7] J. Blomer et al. 2012 *J. Phys.: Conf. Ser.* **396** 052013
[8] Ganglia Monitoring System, http://ganglia.sourceforge.net
[9] T. Berners-Lee et al, RFC2616 "Hypertext Transfer Protocol - HTTP/1.1" (Internet Engineering Task Force) Section 10
[10] J. Postel, RFC 821 "Simple Mail Transfer Protocol" (Internet Engineering Task Force) Section 4.2
[11] http://www.openstack.org/