Migrating to virtual technology with VAC

S H Jones, R Fay, J Bland
High Energy Physics Group, Oliver Lodge Laboratory, University of Liverpool, L69 7ZX, UK
E-mail: sjones@hep.ph.liv.ac.uk

Abstract.
Traditional T2 grid sites still process large amounts of data flowing from the LHC and elsewhere. More flexible technologies, such as virtualisation and containerisation, are rapidly changing the landscape, but the right migration paths to these sunlit uplands are not well defined yet. We examine a recently developed migration route to virtual technology that is currently available to sites called VAC. We installed and tested VAC on a production class cluster and we ran it with a set of VOs for a period of months. We report our findings and conclude that VAC is suitable for large scale deployment.

1. Introduction
Compatibility problems have bedevilled traditional grid cluster solutions to large scale computing. The challenge of orchestrating the deployment, operation and maintenance [1] of a compatible technological baseline has been formidable due to a multitude of seemingly simple interfacing glitches and consequential failures. Virtual payloads isolate the jobs from the underlying technology. This reduces the requirements on the worker node systems. It is expected that virtual or containerised payloads would reduce the scope for version mismatches and other interface disconnects.

VAC [2] is a runtime framework that gives a Plain Vanilla Worker Node the ability to run multifarious virtual machine payloads on behalf of client experiments. We have been testing VAC on a section of our production cluster for three months to probe various aspects of its performance and reliability.

One of VAC’s design goals is to avoid specific clustering middle-ware. VAC can be hosted on a Plain Vanilla Worker Node (PVWN) that is built to a site’s standard but which may omit or disable cluster specific modules, and thus only needs to provide a basic hardware and operating system that is networked and which supports the creation of Virtual Machines.

2. Principles of operation
The basic premise of VAC is to take a PVWN or similar, install and configure VAC on it, and set it in action to collect and run virtual machine payloads. VAC is configured to query various suppliers of VM payloads according to some fair share allowance such that experiment usage coarsely follows a desired workload ratio.

When it finds an experiment with a VM payload ready, VAC downloads it and starts it up. The VM payload executes its own work, while VAC controls the VM life cycle, applying various constraints such as memory or wall clock time and providing runtime information for the guidance of the payload. Once the VM payload ends, VAC is free to download another image.
(or reuse an existing one) and thus the cycle continues. No central coordination exists. VAC hosts can be set up opportunistically wherever and whenever computational resources become available.

It is usual, but not necessary, to use sets of co-located VAC nodes in a cluster, but no head node is necessary. The machines use peer-to-peer communications to status each other and coordinate the workload ratios. We tested a version of VAC that was constrained to run single core jobs, but a multi-core version is now available. Several VOs already include VAC in their pilot frameworks.

3. Test outcomes
Our test took place over three months on a section of our production cluster and consisted of 60 nodes providing a total of 530 jobs slots. The test probed the functionality and performance of VAC. We did not probe traceability requirements, but A McNab (author of VAC) has confirmed that VAC image files are securely fetched via HTTPS from the VO and logged. Furthermore, VAC can be configured to validate CernVM-based images signed using X.509. Traceability inside the VM then relies on the model used by the VO, which is not in the scope of resource providers.

No scalability issues were encountered and VAC achieved near full slot occupancy due to a consistent supply of VM payloads, shown in Figure 1. Better occupancy was obtained towards the end of the test, as we learned to use VAC optimally.

![Figure 1. VAC Cluster Usage](image)

During the formal test, jobs were processed from three VOs: ATLAS, LHCb and GridPP, which is an umbrella VO that various smaller experiments use. The vast majority of the work, 93.4%, came from LHCb. ATLAS used 5.5% of the effort and 1.1% was used by GridPP.
Further data collected more recently shows that use by ATLAS has increased greatly and that another LHC experiment, ALICE, has started to run jobs. For comparison over time, the balance of work over three recent days is shown in Figure 2, where ALICE, ATLAS and LHCb obtained 13.8%, 37.3% and 46.9% respectively (GridPP usage was negligible in this sample.) We observe that these actual usage shares for ALICE, ATLAS and LHCb at least coarsely relate to the provided workload ratios, which were 10%, 50% and 30% respectively. We need to trend this over a longer period than 3 days to confirm that relationship.

VAC may not completely eliminate the need for all middle-ware on the node; for example, VOs normally need peripheral middle-ware to publish accounting records to a central collector, such as APEL [3]. File caching software may also be desirable for CVMFS [4] for reasons of efficiency.

Nonetheless, we found that VAC largely meets its design goals. The total payload traffic from the experiments who participated was found to be consistently high, and the VAC system itself is highly reliable and straight-forward to configure and use. We can recommend VAC to sites that want to run virtual payloads with minimal ongoing maintenance effort.

References
[1] Bird I et al 2014 Update of the Computing Models of the WLCG and the LHC Experiments (Geneva: CERN) p 95
[2] McNab A 2014 Running jobs in the vacuum J. Phys.: Conf. Ser. 513 032065
[3] Rey P, Lopez J, Fernandez C, Kant D, Gordon J The Accounting Infrastructure in EGEE
[4] Blomer J, Buncic P, Meusel R 2013 The CernVM File System