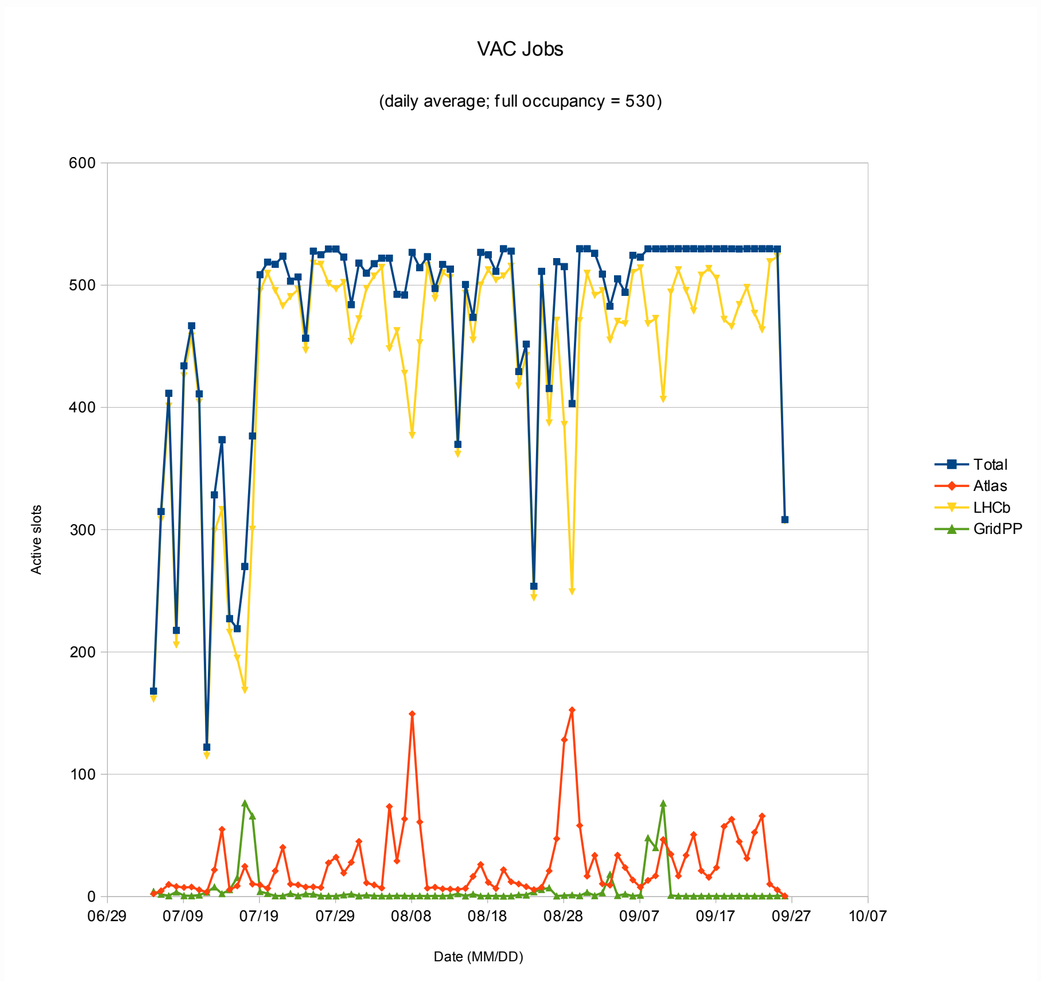
GridPP
UK Computing for Particle Physics

TESTING THE VAC VM PAYLOAD FRAMEWORK AT LIVERPOOL

VAC 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 tested VAC on a section of our production cluster for some months to probe various aspects of its performance and reliability.

Compatibility problems have bedevilled traditional grid cluster solutions to large scale computing. The challenge of orchestrating the deployment, operation and maintenance 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 system. It is expected that virtual or containerised payloads would reduce the scope for version mismatches and other interface disconnects.

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.



The test took place over four months on a section of our production cluster and consisted of 60 nodes providing a total of 530 jobs slots. No scalability issues were encountered and VAC achieved near full slot occupancy due to a consistent supply of payloads.

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 presently comes from ATLAS.

VAC does not completely eliminate the need for all middle-ware on the node; for example, some peripheral middle-ware software was needed such as APEL accounting and http file caching for CVMFS.

Nonetheless, we found that VAC largely meets its design goals. The total payload traffic from the three 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 require a easy migration to virtual payloads that involves minimal ongoing maintenance.

VAC Principles of Operations - The basic premise 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 pre-set ratio.

When it finds an experiment with a payload ready, VAC downloads it and starts it up. The 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 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 computation 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. Presently, VAC is constrained to run single core jobs, but multicore jobs are planned, and other VOs have plans to include VAC in their pilot frameworks.