Testnodes – a Lightweight Node-Testing Infrastructure

Rob Fay (fay@hep.ph.liv.ac.uk), John Bland (jbland@hep.ph.liv.ac.uk)

The Problem

In late 2007 the largest part of the Liverpool HEP cluster was the MAP-2 supercomputer, which had been ranked 86th in the top500 when commissioned in 2003. By 2007 it had inevitably dropped out of the top500 but remained a significant computing resource.

But as it aged, system failures naturally became more frequent, a problem exacerbated by the size of the system – 24 racks of single-core worker nodes, 960 in total.

These often led to 'black holes' in the batch system – nodes that would receive jobs, but then instantly fail them, effectively emptying the queue and resulting in intermittently high failure rates, and relatively poor reliability and efficiency. This required a high level of monitoring and manual intervention to remove problem nodes from the batch cluster while they were repaired.

Solution

Nagios was in use for monitoring, but was under a heavy load, struggling to keep up with the level of testing already in place. Increasing the frequency of testing to a level sufficient to adequately minimise 'black holing' on the cluster was not feasible at the time with the resources available. Additionally, implementation of the desired framework within Nagios presented some challenges and limitations.

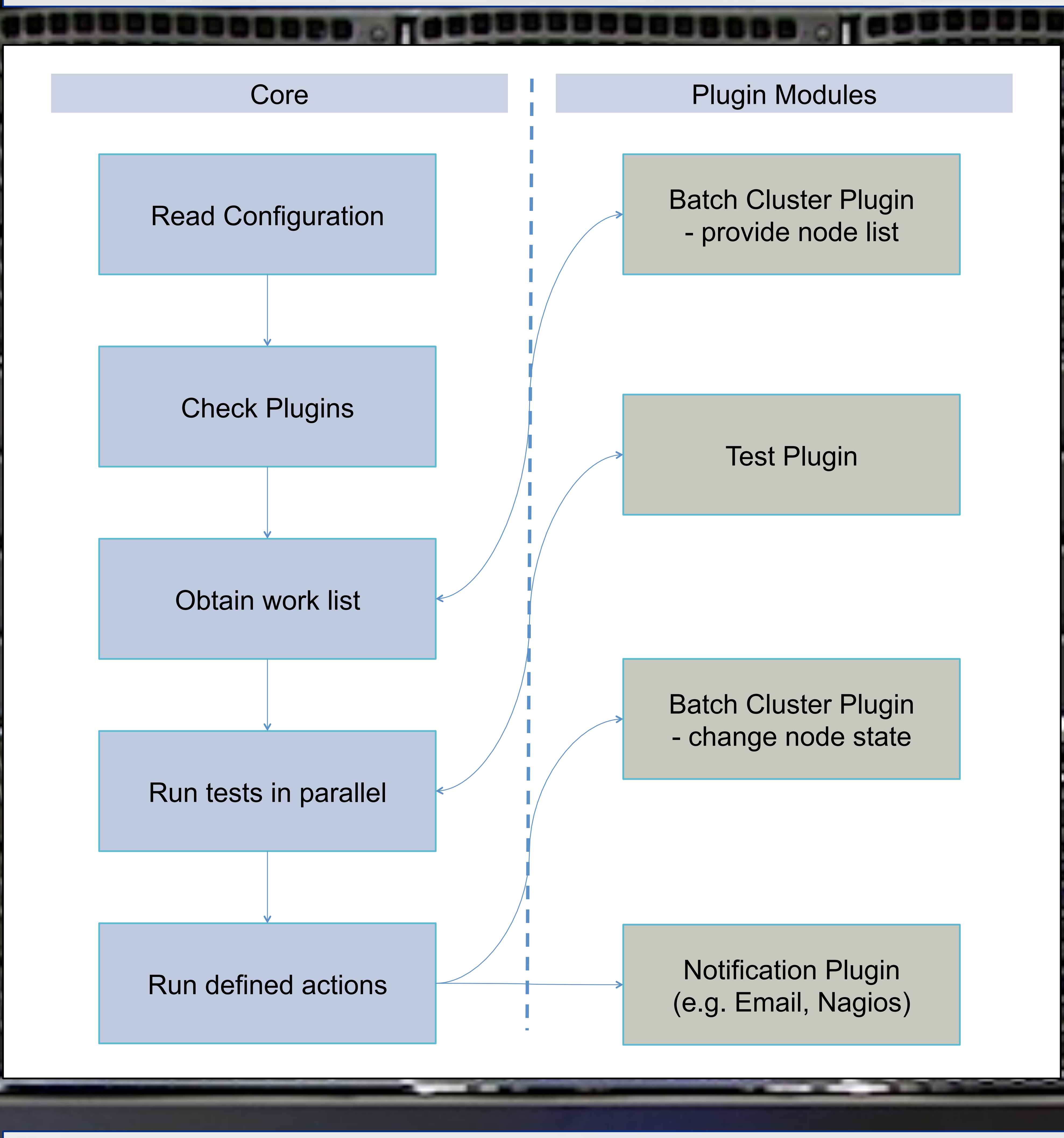
Monitoring the batch system itself, taking nodes offline when jobs failed, was an option but could only detect failures after the event, not before, could not detect when failures had been resolved without additional effort, and could not detect errors where the fault resulted in a failure within the job itself, but not from the perspective of the batch cluster.

The solution Liverpool adopted was a simple monitoring framework written from scratch.

Initial Outcome

'Testnodes' was implemented in Python with a single focus to testing the cluster; essentially asking the question, "Is there any reason why this node should **not** be available for jobs?" and then marking the node off or on-line in the batch system accordingly. With the tests run in parallel from a management node, it became possible to thoroughly test the entire MAP-2 cluster within a few minutes.

Since introduction, testnodes has carried out over half a million tests of the entire Liverpool cluster. It had an immediate impact on availability and reliability, which increased from 87% and 89% in the prior two quarters to 93% and 96% in the next respectively. By October 2008, with refinement of testnodes, Liverpool was one of only six (out of 263) sites to achieve 100% reliability and availability, despite running one of the older clusters at the time. To date, Liverpool has typically maintained over 98% availability and 99% reliability on a consistent basis.


Core
Plugin Modules
Read Configuration
Check Plugins
Obtain work list
Run tests in parallel
Run defined actions
Batch Cluster Plugin - provide node list
Test Plugin
Batch Cluster Plugin - change node state
Notification Plugin (e.g. Email, Nagios)

New Design

While successful in achieving its goals, the original implementation of testnodes was relatively crude. Developed as a proof-of-concept and quick solution to an immediate problem, it was written in largely monolithic code with embedded configuration and did not lend itself to being easily adapted for other batch systems or other purposes.

The redesign uses a more modular approach, with the core framework relying on configurable plugin modules to interact with the local batch system, as well as defining the test(s) to be carried out and any resulting actions or notifications to be performed.

Both the original implementation and the redesign are written in Python, with the original using Python 2.4 and the new version Python 3.3.

Configuration

- Maximum number of simultaneous test threads
- Timeouts for individual tests and complete testing
- Plugin directory
- Input plugin(s)
- Test plugin(s)
- Action/notify plugin(s) to act on individual test results
- Optional summary plugin(s) to act on all results

Input Plugin

Returns a list of subjects to be tested. Other information (e.g. Initial state) can be maintained within the plugin module for use by subsequent plugin functions (e.g. actions).

Test Plugin

Defines the test to be carried out on a subject. Must be thread-safe. The current implementation uses a test script installed and kept up to date on the nodes by Puppet, executed securely via SSH.

Action/Notify Plugin

Performs an action, either one conditional on test results, or a default action, and an (optional) obligatory action.

Conclusions and Future

- Testnodes has proven to be a useful solution at Liverpool, for both grid clusters and local batch systems
- Improved reliability, availability, and efficiency of jobs
- Reduced load on system administrators
- Redesign (in progress) provides platform for further adaptations and optimisations
- Could do with a better name!

UNIVERSITY OF LIVERPOOL

GridPP
UK Computing for Particle Physics