White Paper:

Kernel Control in a High Performance Compute Cluster

By Adam Carrgilson

February 2014

Background

The Linux servers (nodes) which comprise the High Performance Compute (HPC) clusters at the Norwich Bioscience Institutes (NBI) have their configuration managed using a system called Puppet[1]. This system performs operations in the background on every node, running tasks to keep each node up to date and in synchronisation with one another. Puppet is aware of any hardware or software features that may differentiate the nodes and will work to install any dependencies that might be required for those features to operate.

One such feature we worked towards incorporating into to the cluster and managing under Puppet, was the "plug and play" installation of Nvidia Tesla[2] dedicated general purpose graphics processing units (GPGPU).

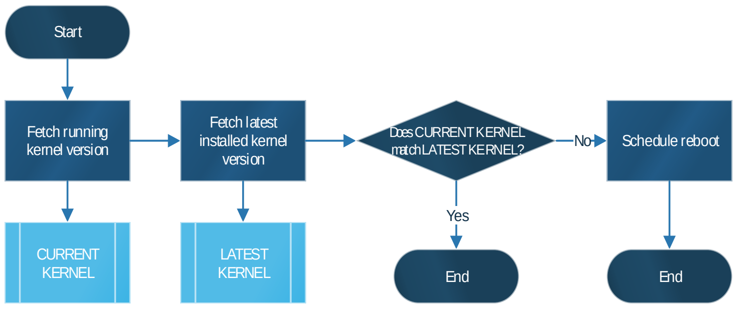
As a requirement of supporting this hardware on the compute nodes, they need to load a device driver in the form of a Linux Loadable Kernel Module[3] (LKM). As LKMs generally function only with the kernel against which they were compiled, every node in the cluster which could host a GPU had to be running with matching versions of the running kernel and source code for the installed kernel headers[4], so that an LKM for the Tesla could be compiled and successfully loaded.

Puppet, as a matter of course, frequently performs system updates, renewing every operating system component to the latest available version from a software repository, which will include changes to the kernel and kernel headers packages. However, the latest installed kernel will then sit dormant on disk until it is loaded upon the node's next reboot. This interferes with automating the installation of the Nvidia drivers, as until the node has been rebooted there is a risk that the installed version of the kernel headers could be out of step with the running kernel.

Systems & Method

To address these challenges we set about defining a new configuration manifest for Puppet to apply.

First of all the Puppet module needed to understand if a mismatch existed between the latest version of the kernel installed and that of the running kernel, i.e. has a kernel update been installed but not yet loaded?


Start
Fetch running kernel version
Fetch latest installed kernel version
Does CURRENT KERNEL match LATEST KERNEL?
No
Schedule reboot
CURRENT KERNEL
LATEST KERNEL
Yes
End
End

This was accomplished using a shell script which compared the version of the currently loaded kernel, obtained via uname[5], with the last installed version registered in the Red Hat Package Manager[6] (RPM) database. The script exits with a return code which Puppet will interpret as a signal that either a disparity exists or does not.

If such an inconsistency is detected, Puppet will then attempt to schedule a reboot for that node, which is then implemented via another shell script.

[1] http://puppetlabs.com/puppet/puppet-open-source
[2] http://www.nvidia.com/object/tesla-supercomputing-solutions.html
[3] http://www.tldp.org/HOWTO/html_single/Module-HOWTO/
[4] http://kernelnewbies.org/KernelHeaders
[5] http://en.wikipedia.org/wiki/Uname
[6] http://en.wikipedia.org/wiki/RPM_Package_Manager

Systems & Method cont.

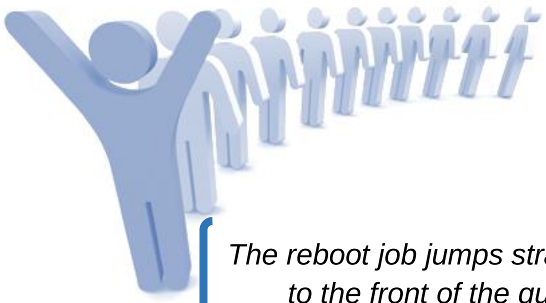
The reboot script is intrinsically the most important component for keeping the running kernel and the version of the kernel headers in sync with each other on a node, without disrupting the currently executing scientific workload, which can last for hours or even days.

The workflows executing on the HPC cluster are managed by the IBM Platform LSF[7] scheduling system, and it is also by using LSF that we can ensure the safe timing of an individual node reboot and the loading of the new kernel.

The reboot script dispatches a job to the LSF scheduler and we specify several flags to control how this will run. Firstly we ensure that the reboot 'job' will have exclusive use of the node it is running upon, safe-guarding that no scientific workload will be sharing the node at the same time as the reboot occurs. Then we specify that the job enters a specific '*administration*' queue which assigns the task the highest level of priority, such that LSF will dispatch it to run it before any other queued jobs which may get scheduled to execute on that node. Finally we specify the host on which the reboot job should run; that being the host generating the kernel mismatch signal identified earlier.

Finally we lock down the scheduling of the reboot so that it will only occur once per node, by writing a flag file to a transient tmpfs[8] memory file system location and signalling Puppet to cease scheduling additional tasks if the file exists.

This is necessary because Puppet may test and trigger the kernel mismatch condition several times over while waiting for the scheduled reboot job to be executed. When the node has rebooted the flag file previously stored on the (volatile) tmpfs file-system no longer exists and therefore Puppet can freely schedule another reboot should it be necessary, e.g. to resolve another discrepancy.



The reboot job jumps straight to the front of the queue.

Benefits

As well as enabling the cluster nodes to automatically install and run the loadable kernel module for the Nvidia Telsa devices, running the latest kernel also eases maintenance by ensuring that security fixes, stability improvements, updated drivers, new functions and concomitant performance improvements are continuously incorporated into the up-to-date kernel[9].

Development

This method and means of controlling numerous systems is part of a larger, on-going project at NBI. As the (Puppet) system is expanded, this level of control could be applied to other software, or perform other administration and maintenance tasks. One such future utilisation may be to implement the techniques described here to integrate Intel Xeon Phi[10] coprocessors within the HPC clusters.

The scope of the system may widen and even be rolled out to desktop machines. It is upon these foundations which we can grow and support further enhancements leading to a better service to science and to the institutes.

[7] http://www.ibm.com/systems/services/platformcomputing/lsf.html

[8] http://en.wikipedia.org/wiki/Tmpfs

[9] http://www.makeuseof.com/tag/5-reasons-update-kernel-linux/

[10] http://www.intel.com/content/www/us/en/high-performance-computing/high-performance-xeon-phi-coprocessor-brief.html

http://cis.nbi.ac.uk
adam.carrgilson@nbi.ac.uk, paul.fretter@nbi.ac.uk