FABRIC MANAGEMENT WITH DISKLESS SERVERS AND QUATTOR IN LHCb

P. Schweitzer*, E. Bonaccorsi, L. Brarda, N. Neufeld, CERN, Geneva, Switzerland.

Abstract

At CERN LHCb experiment (Large Hadron Collider beauty), in order to reconstitute and filter events, a huge computing facility is required (currently ~1500 nodes). This computing farm, running SLC (Scientific Linux CERN) [1], a Red Hat Enterprise Linux (RHEL) [2] derivate, is net booted and managed using Quattor [3] to allow easy maintenance. LHCb is also using around 500 diskless Credit Card PCs (CCPCs) embedded in the front end electronic cards. To go farther than the limited Red Hat provided (and now deprecated) netboot tools, a new solution has been designed and will be shortly presented in this paper.

LINUX DISKLESS

All these computing nodes get their operating system (Linux) from some dedicated servers (each server is controlling a slice of the farm).

Boot Process

The node's boot ROM sends a DHCP query. Its server replies with the IP address to use and the location of the pxelinux.0 stage 2 loader file. The node gets this file by TFTP and starts it. The stage 2 boot loader then loads the Linux kernel and initial ramdisk and gives the control to the initialisation script located in the ramdisk.

This script has to setup the networking, mount the root file system from the server and switch to this NFS root. It then passes the control to the usual Linux initialisation scripts.

For many reasons, the running Linux needs to write files in many locations. We will now see the Red Hat way of making these files writeable and the problems and limitations of this method. Then we will talk about what was developed to circumvent these problems and limitations.

Diskless: Red Hat Way

Up to RHEL5, Red Hat had a package named system-config-netboot to setup diskless servers. It was a set of python and bash scripts that were setting up the dhcpd and tftpd servers, customising the shared root file system and making the initial ramdisk for the diskless nodes.

To make some files of the root file system writeable for the nodes, with possibly different contents, the initialisation script from this package was making the following actions after having mounted the root file system:

- Mount the 'snapshot' directory from the server, in read/write mode. This directory contains one sub-directory per node and two files with the list of the files that need to be writable.
- Remount (using the bind mount option) each of these files from the node's snapshot to the root file system.

There are two problems with this method:

- Only files or folders of the fixed list can be writeable. To add a file to that list, we have to reboot the nodes after the file list modification.
- The mount table is 'polluted' by all these remounts.

Diskless: New LHCb Way

To add flexibility to the diskless nodes handling, we had the idea of using file system union, which is usually used on Linux live media (CDs, DVDs or USB keys). As LHCb is using Scientific Linux CERN 5 in production and SLC 6 has been released and will go in production, the new system had to support both.

FILE SYSTEMS UNION

The principle of the file system union is to join several file systems, at least one read-only (generally) and one read-write and use that union as a normal Linux file system. The behaviour is the following:

- When creating a file, the file is directly created on the read-write file system.
- When writing a file, if the file only exists only on the read-only file system: the modified file (called copyup) is written on the read-write file system with the same name and hides the read-only version.
- When erasing a file, if the file only exists on the read-only file system: a whiteout file (filename prefixed by '.wh.') is written on the read-write file system to hide the file.

Table 1: Union File Systems Evaluation

		Feature				
		mode	Copy on write	NFS branches	Kernel specific rebuild	Metadata separation
Implementation	UnionFS [4]	kernel	Yes	Yes	Yes	No
	Aufs [5]	kernel	Yes	No	Yes	No
	Funionfs [6]	FUSE [8]	Should be	Yes	No	No
	Unionfs-fuse [7]	FUSE	Yes	Yes	No	No

- Renaming a file is done by copying the old file under the new name on the read-write file system and hiding the old file with a whiteout file.

Several implementations of file system fusion were evaluated. Table 1 shows the result of this evaluation.

We first tried to make the union on the nodes during diskless initialisation but finally choose to do it on the server, and NFS exports the "unioned" file system. Client side union was just using too much memory on the CCPCs.

An Union File System Designed for Diskless

While evaluating union file systems implementations, it became clear that none was perfect for net booted nodes. All were designed with totally different goals than ours.

One of the big problems was that too many copyups were made on the read-write file system.

So we decided to implement an union file system designed for diskless systems, with the following functionalities:

- union between only one read-only and one read-write file systems
- if only the file metadata are modified, then do not copy the whole file on the read-write files system but only the metadata (stored with a file named as the file itself prefixed by '.me.')
- check when files on the read-write file system can be removed

As a proof of concept, a first version of this new union file system, named PierreFS (until we find a better name) as been developed and successfully tested: the read-write file system was only containing configuration files configured by our Quattor configuration system and runtime files in /var. Even library pre-linking, which was making a lot of copyups in other implementations did none in PierreFS.

The next step will be the implementation of this file system directly as a module for the kernel, without the use of FUSE. This is a huge work that will make it more efficient using less memory. Of course, it will have to be compatible for both SLC5 and SLC6 kernels.

QUATTOR COMPONENT

The LHCb experiment is using the Quattor toolkit to manage the configuration of all its Linux servers and nodes.

In the previous Quattor component in charge of the diskless configuration, supporting the Red Hat way, many things were done by the Red Hat system-config-netboot package. Some of the scripts in this package were buggy, with no output in case of problems. Red Hat totally removed that package in RHEL6, the distribution SLC6 is based on.

As the Quattor component itself was not really maintainable, it was decided to completely rewrite it to make it more efficient and modern, and handle completely the new diskless model, of course without using the system-config-netboot scripts.

System-config-netboot Replacement

In this package, several scripts were used by the Quattor component:

- pxeos & updateDiskless were in charge of copying the kernel and building the diskless specific initial ramdisk (initrd.img): in the new component, updateDiskless repackaged, is still used for SLC5 while rracut is used for SLC6. CCPCs are a special case as they do not support pxe: the wraplinux utility is used to create the nbi file containing the kernel and the ramdisk together.
- pxeboot was in charge of making the pxe configuration files for the nodes. The functionality is now in the Quattor component.
- mkdiskless was used to customise the shared root file system for diskless use. The functionality is now in the Quattor component, with a possible call to an external script.

Other Improvements

Rewriting the component gave us the possibility to greatly improve it and simplify the setup of diskless servers.

In the previous setup, the root file system shared by all nodes was created by copying the server root file system at the end of its installation. It was not possible to have clients with a different architecture or a different version of the Linux distribution. We added the functionality to create the shared root file system for any Linux distribution and architecture, by using the '--installroot' parameter of the yum command. The only thing we have to do is to provide the component a yum configuration file. We can even have one server with many shared root file systems, each with different versions of the distribution and/or different settings. Like this, we can have a node running SLC5, just change some settings on the server, and then have a restart under SLC6.

The dhcpd configuration part now has better support for subnets and the generated files follows more closely the standard, using dhcpd groups to avoid repeating the same options for all hosts.

The diskless component also keeps track of the created and administrated nodes. That way, in case of deletion of a node in the configuration, it can properly remove it, without leaving leftovers. It also handles configuration changes on the nodes (MAC address change, IP address change).

Changes to Another Component

To configure the shared root file systems, the Quattor component runs the Quattor utilities "chrooted" in these root file systems. To ensure proper run of Quattor inside of Quattor, the proc pseudo file system is mounted just before the change root and the Quattor run. This allows locks consistency, but also enables mounting/dismounting. That is why the Quattor NFS component has been fixed to add an option that makes this component only write configuration and not attempt any mount or dismount.

CCPC Support

This new diskless component also comes with a great improvement since it adds CCPCs management into Quattor. Some more tweaks are done on the CCPCs shared root file system and snapshots, as disabling Quattor on them (that would have taken too many resources).

SIDE EFFECTS

On a side note, the use of a FUSE file system as root file system has exposed a bug in the Linux kernel (that appears to be well-known). A workaround has been deployed in the ramdisk init script.

Also, the use of this file system helped revealing security vulnerability in the Linux kernel. Effects of that security vulnerability are really limited and only allow someone who does not have access on a directory to read its contents. Since it is hard to reproduce it out of the box, it has not been reported yet (no easy test-case).

CONCLUSIONS

The concept of using file system union to make diskless systems easier to manage has been successfully tested. We even have new use cases where the way of doing diskless nodes system does not work (eg: Dell management tools, which want to write some inventory files in many places).

The new Quattor diskless component, together with PierreFS will go in production during the next winter shutdown. It will allow us to better manage the Credit Card PCs, which are not supported by the current system. It will also ease the management of the LHCb event filtering farm. With the support of several Linux distributions on the same server, we will be able to easily test SLC6 on the farm nodes, with the ability to go back to SLC5 with a simple reboot of the nodes.

The new file system has been published on SourceForge [9] and the work to integrate it to the kernel will start in October 2011.

REFERENCES

[1] http://cern.ch/linux
[2] http://www.redhat.com/rhel/
[3] http://www.quattor.org
[4] http://www.fsl.cs.sunysb.edu/project-unionfs.html
[5] http://aufs.sourceforge.net/
[6] http://funionfs.apiou.org/
[7] http://podgorny.cz/moin/UnionFsFuse
[8] http://fuse.sourceforge.net/
[9] http://sourceforge.net/projects/pierrefs/