CUC 2004

HIGH PERFORMANCE CLUSTER DISTRIBUTION DESIGN

Authors:
Nikola Pavković,
Valentin Vidic,
Karolj Skala

Table of Contents

1. INTRODUCTION

Linux clusters are being more and more adopted as supercomputing infrasructure facilities in scientific research laboratories, but also in other kinds of research work where high computing power is essential. With their price/performance ratio linux clusters more and more take over the market share of specialized super-computers.

In order to build a "linux cluster" from a number of standalone PCs, one must upgrade a standard linux distribution with some extra functionality which will provide easy installation, administration, enforcing the security policy and monitoring of elementary resources within a linux cluster. Althogh some of the tools already exist, there are a very few complete distributions of linux targeting high-perfomance users.

In this paper, we will cover technical issues regarding deployment of a Linux cluster as a high-performance infrastructural facility.

2. CLUSTER TYPES

2.1 SSI clusters

Clusters which do not have the operating system installed on every node's disk drive, but run only one single copy of the operating system on the master node, are called "Single System Image" clusters (SSI).
Usually, users but also applications, do not have to be 'aware' of the distributed environment, because SSI clusters provide some kind of transparent middleware which makes the bunch of nodes look like one big computer. The whole cluster shares the same PID space, running processes can migrate between the nodes, in order to balance the load on the cluster, UNIX domain sockets and also pipes are shared between the nodes etc...

One of the most promising project adressing the Single System Image cluster deployment is OpenSSI [http://openssi.org]. It integrates various technologies like OpenMosix [http://openmosix.sourceforge.net/] (used for dynamic process migration), OpenGFS [http://opengfs.sourceforge.net/] (used for sharing the same filesystem among all nodes), and also provides a programming interface for some cluster-specific functions.

The main problem with SSI clusters today is scalability. The concept of a single sistem image used by all nodes results with a bottleneck on the storage system interface as the number of nodes grow. Because of this (technological) limitation, largest SSI clusters today have up to 128 nodes.

2.2 non-SSI clusters

In large clusters, most usual method for accessing the system image is to have it installed on the local storage device (i.e. hard disk). Since every node in the cluster runs it's own copy of the operating system, it is pretty obvious that maintaining such a bunch of Linux machines is not an easy job, especially as the number of nodes grow. There are some specialized Linux distributions that integrate some extra functionality into the system in order to simplify maintainance of large clustered systems. The most used cluster-targeted distributions are NPACI ROCKS [http://www.rocksclusters.org/] and OSCAR [http://oscar.sourceforge.net/], both based on RedHat linux.

Let's address the real technical problems that one must face when building a Linux cluster.

3. PROBLEMS

3.1 Automatic node installation strategies

In order to minimize the time needed to deploy a linux cluster, one of the important features that should exist in a clustered system is an automatic node installation mechanism. There are several approaches to this problem, so let's see what can be done to automate the node-installation process.

3.1.1 RedHat KickStart file

As of version 5.0, RedHat [http://www.redhat.com/] has introduced the so-called KickStart installation method. The kickstart file (called ks.cfg) is used to hold every single information that is needed to install the operating system, so human attendance is not necessary. The kickstart file actually contains the following information:

- Language selection
- Mouse configuration
- Keyboard selection
- Boot loader installation
- Disk partitioning
- Network configuration
- NIS, LDAP, Kerberos, Hesiod, and Samba authentication
- Firewall configuration
- Package selection
- X Window System configuration

As the installation process begins, the kickstart file should be provided to the installation process. As the installation process gets all the required information, everything else is done automatically. The NPACI ROCKS distribution takes advantage of this functionality, which will be described later.

3.1.2 System Installation Suite (SIS)

Another approach to fast deployment of a linux cluster is to use SIS software. SIS is a software suite developed especially for automation of installation tasks. The suite consists of three components: systemimager, systeminstaller and systemconfigurator. Philosophy of SIS software is to build the target image on the image server, and then deploy it on a number of (similar) computers. This is almost exactly what we want to do in a linux cluster.

3.2 Distributed filesystem

In practice, some parts of the main filesystem located on the login-node of the cluster need to be accessible from all the work-nodes. For example, it is very convinient for the user to have his/her home directory content available on the work-nodes, because it is a common practice for the user, if using a non-system-wide installed application , to distribute the executables, libraries, datasets, licence files etc. within his/her home directory. In order to provide this funcionality, most of the clusters have some kind of network filesystem distribution. Speaking about production-level clusters today, the most often used method for distributing the filesystem contents to the work-nodes is the NFS (Network File System). [http://nfs.sourceforge.net/]

There are also some other approaches to this issue. For example, the PVFS (Parallel Virtual Filesystem) [http://www.parl.clemson.edu/pvfs/]. The main idea in PVFS is to have the storage units distributed almost symmetrically on all the cluster-nodes and provide a TCP communication interface in order to integrate those storage units logically, appearing as a big virtual filesystem, accessible from all the cluster-nodes. Although this idea is very promising, there still exist some performance issues, especially with applications that use a small I/O buffer when accessing the filesystem.

3.3 Centralized account management

In order to provide full user-account information to the work-nodes, the central user-account information database must be distributed within the cluster. There are several possibilities to solve this problem.

3.3.1 File replication

Every time a change is made in the central user-account database located on the login-node, a script is triggered, which copies all the changed files to all the work-nodes, at least to the ones that are alive at that very moment. Beside that trigger-based replication, a regular automatic synchronization must be done.

3.3.2 NIS (YP)

Another way for user-account information distribution is to use NIS (Network Information Services) protocol [http://www.linux-nis.org]. The NIS server resides on the login-node, and provides all the user-account information to the work-nodes.

3.3.3 LDAP

Lightwight Directory Access Protocol is becoming more and more popular for accessing different kinds of directory-structured information within big organizations. An interesting approach when building a cluster is to store user-account information in a LDAP server on the login-node, so that the work-nodes contact the server when authenticating a user, or accessing user'saccount-information. The database can easily be replicated to additional LDAP servers within the cluster, in order to balance the network utilisation in big clusters. Since LDAP is very popular for information distribution in grid-computing, it makes a good choice to use it as a centralized account management tool also. An interesting point is that LDAP, for it's openness, has some very interesting application possibilities beside these mentioned above [Debconf [debian configuration management system] database can be stored within a LDAP directory, so it could be possible to centralize the whole cluster-wide system configuration within a single database.]

3.4 Resource management

In order to integrate the cluster-nodes in a unified computational resource, some kind of central resource management system is needed. The resource management system takes care of load-balancing within the cluster fair-usage enforcement etc. Different resource management systems exist today, some of them are Torque, SGE, DQS...

3.5 Health & performance monitoring

One of important issues when deploying a high-performance cluster is the performance and health monitoring system. There are many things a system administrator will want to monitor when putting the cluster to production. Graphical web-enabled interface is also desirable, so that all the monitored data can be visually displayed.

3.5.1 Ganglia

One of the most used tool for monitoring system health and performance within linux clusters is called ganglia [http://ganglia.sourceforge.net/]. It's architecture allows it to operate in very large clusters, minimizing network I/O overhead and using multicast channels. The ganglia system is very extesible because of the 'gmetric' modules that provide a script-interface, so one could write a simple script to monitor almost anything.

3.6 Software packages distribution management

When deciding which software should be installed on the cluster-nodes, a comfortable environment for software selection, configuration and testing is desireable. There are few approaches to this issue which we'll be addressed now.

3.6.1 NPACI ROCKS - an XML approach to system configuration

The ROCKS cluster distribution, one of the most mature cluster-distributions on the open-source market today uses a interesting technique for software configuration, involving XML. Since the ks.cfg (the RedHat kickstart file, on which ROCKS installation automation system strongly depends) holds all the information about the software that is to be installed on a cluster-node, it (ks.cfg) is being build dynamically as the node requests the file from the install server. The XML configuration files get parsed and the resulting ks.cfg is built on the fly, providing the node all the information it needs to install itself automatically.

3.6.2 System Installation Suite (SIS)

The mechanism behind SIS suite is very simple. Since the image of the target operating system is built on the image-server, one can easily modify the contents of the image by simply 'chrooting' [see section **4.3.1**] into the image's direcory and issuing commands just as one was sitting at the target-node console. After the modifications to the image are done, the administrator exits the chroot jail, and issues a command to deploy the new image on the target nodes.

4. Our efforts within the DCD project

4.1 Debian as a base - our choice

There are many linux distributions available on the market today, and every one of them has some pros and cons. But, some quality characteristics make Debian GNU/Linux a prime choice when selecting a platform for intensive "Mission-critical" applications.

From practical expirience, it is very well known that Debian handles the security patches in a very unique, admin-friendly way. The average system can be upgraded to up-to-date package versions in a few minutes, issuing only one single command. This is essential, because no administrator wants to spend his/her time manually resolving inter-package dependancies. There are some tools developed for this task targeted for other distributions, but APT (Debian's package tool) has been integrated into the system a long time ago, and is prooved to be very reliable tool, which is capable of resolving most complex inter-package dependecy problems.

On the other hand, Debian security team is one of the most responsive security teams among other linux distribution's security teams. All the disclosed security issues are patched and put to the official Debian APT mirror sites in less than 48 hours. Debian is often the first linux distribution that releases a patched package when a security problem occurs. In order to keep the system's security level high, this is a very important issue.

Legal aspect also sugests Debian. Debian tries hard to be the 'purest' GNU distribution. The social contract assures that all the developed software is to be held within Open Source. Since it is driven by 'phylosophy', rather than the market, the concept is fully functional for more a decade. Unlike some other distributions, the first goal for Debian is quality of released software, and since it is not driven by the market, there is always plenty of time to assure the quality of software.

Looking from technical, legal and security points of view, Debian makes the first choice when selecting a linux distribution for mission-critical deployment. Therefore, we decided to develop some extra tools for the Debian system that will provide cluster-deployment functionality. A small OSS project is born, for now we call it DCD, which stands for the Debian Cluster Distribution.

4.2 DCD structure

DCD is meant to be a add-on bunch of packages that are to be installed on top of a regular Debian Sarge distribution. By adding an entry to the sources.list configuration file, and issuing one single command one can have a fully functional cluster-front-node ready for automatic installation and deployment of working-nodes in the cluster.

Let's see how DCD handles different issues mentioned above.

4.2.1 Automatic installation and deployment of working-nodes

DCD strongly depends on SIS for automatic deployment of a system-image on the work-nodes. For the SIS suite to become fully functional we had to patch some parts of it, namely some parts of the SystemInstaller component. The image is built issuing only one single command, and the resulting image is very easy to modify. We developed a SIS add-on command called 'editimage' which puts the administrator into the chroot jail of the desired image so he/she can modify the image just the same way if the image was on a real work-node [apt-get ...] As the admin has finished modifying the image's content, he/she simply exits the shell and issues another command 'cpushimage' [a part of c3 suite] in order to update the image on the real work-nodes.

As opposed to the XML-approach, we find this way of image-modification much more natural and intuitive. One doesn't have to learn the concepts of XML in order to maintain a linux-cluster, and since the actual full image of the working node filesystems is available right on the image server, it is natural to enter a chroot-jail within the images' directory and behave just like one was sitting in front of a real work-node console.

We also developed some tools for dynamic work-node insertation, leaving behind the need to manually build the SIS database using the "mksirange" command everytime a new node is put into the cluster.

4.2.2 Automatic queueing system configuration

Since fine-tuning the queueing system can never be fully automated, management of the queueing system does not depend on the cluster suite, but is better done manually. For less expirienced users, we decided to include generic automatic queueing system configuration by default.

4.2.3 User files distribution in the cluster

For distributing users' home directory contents within the cluster the Network File System (NFS) was chosen. NFS has prooved to be very stable and reliable, and for it's robustnes it is our prime choice according this issue.

4.2.4 Monitoring the resources and alarming in critical situations

For it's scalability and flexibility, ganglia was chosen for system performance monitoring. As ganglia itself is capable of acquiring almost any thinkable information from all the cluster-nodes in a very scalable way, but is, on the other hand unable to provide reasonable alarming functionality, we are

considering integration of glaglia as a information-aggregator and nagios or similar software for alarming in critical situations. Some debian packages, particulary for ganglia-web-frontend, have already been assembled within the DCD project, you can find more information in the Appendix section.

4.2.5 Managing NIS information within a cluster

One of the most challenging issues when building a cluster is the underlying authentication infrastructure. Since LDAP has prooved to be very stable in production deployment, it's flexibility and openness made it the prime choice for authentication and user-information distribution within the cluster.

LDAP has a very promising future. For it's openness it is applicable in almost every layer of the IT infrastructure. Some pretty interesting functionality can be provided by LDAP, such as debconf integration, SIS integration (as a beckend for the SIS database), but this work is still in early development.

We are considering LDAP replication on cluster-nodes, in order to balance the network load in large clusters.

4.3 The dcd-utils software package

The dcd-utils software package consists of the tools developed within the DCD project. These tools are used to simplify different tasks when administering a cluster of computers and addressing some issues mentioned above. Let's name some of them:

4.3.1 editimage

The editimage tool is a simple shell script that takes care of all the possible issues when editing an work-node system image. The image of the working node is located on the front-node's filesystem and represents an exact replication of the real-work-node system image. Therefore it is possible to enter a so-called 'chroot-jail', which means that the interactive shell 'sees' that directory as the root directory of the system. Editimage takes various actions when executed:

- enters such a chroot-jail within the images directory,
- mounts /proc filesystem within it,
- replaces the start-stop-daemon script with the dummy script in order not to (re)start services when upgrading a package within the image, and
- gives a shell.

As the wanted changes are made to the image after exiting the shell,editimage takes following actions:

- unmounts the /proc filesystem,
- puts back the original start-stop-daemon script, and
- removes the lock file.

This approach makes management of the work-node system configuration very comfortable, because it is done pretty much in the same way as sitting on the console of a real computer and making changes to a live system. After the task is completed, the 'cpushimage' command may be issued in order to apply the changes to the work-nodes.

As opposed to the XML-style work-node configuration found in ROCKS Linux cluster-distribution, our approach does not need any extra knowledge about XML and is therefore easier to adopt by less expirienced users.

We have also noted significant performance boost in administration tasks regarding work-node system configuration and maintainance while using this mechanism.

4.3.2 dcd_adduser

Since we use LDAP to store user account information, we needed to create a tool that eases the user-management tasks when dealing with LDAP. Also some other issues are covered. Automatic SSH key generation is preffered right after the account has been opened, and the public key is copied to the `$HOME/.ssh/authorized_keys` file which makes a transparent access to every node in the cluster possible. It should be mentioned that using NFS, the users' home directories are available on all the nodes, so that the `.ssh/` directory is available, and the authentication can be done with SSH keys.

4.3.3 discover_node

System Installation Suite, which is used as the core autoinstallation mechanism for the work-nodes keeps host information (hostnames, IP addresses, MAC addresses, Network configuration...) within the SIS database. A typical database content is shown below:

```
Machine definitions.....
Name            Hostname             Gateway         Image
-----------------------------------------------------------------
node1           node1.cluster        10.0.0.1        sarge
node2           node2.cluster        10.0.0.1        sarge
node3           node3.cluster        10.0.0.1        sarge
node4           node4.cluster        10.0.0.1        sarge

Adapter definitions.....
Machine         Adap  IP address      Netmask         MAC
-----------------------------------------------------------------
node1           eth0  10.0.0.10   255.255.255.0   00:02:b3:9c:35:30
node2           eth0  10.0.0.11   255.255.255.0   00:02:b3:9c:31:89
node3           eth0  10.0.0.12   255.255.255.0   00:02:b3:9c:a6:95
node4           eth0  10.0.0.13   255.255.255.0   00:02:b3:9c:40:ad
```

SIS comes with all the necessary tools for managing that database, but although a very practical tool, SIS lacks dynamic node insertation mechanism. It means that every time a new node is added to the cluster, one must manually enter the MAC address information to the database in order to make the autoinstallation work properly. Therefore we developed a handy tool to automate the node insertation process. The script expects the DHCP server logs on the standard input, parses the contents, and after a DHCP request is found, the MAC address is extracted. A typical DHCP log entry of interest looks like:

```
Aug 28 14:15:02 fk-grozd dhcpd: DHCPDISCOVER from   \
                    00:02:b3:9c:40:ad via eth1: network\
                    10.0.0.0/24: no free leases
```

After that, various actions are taken. First, the SIS database is being queried to check for possible existance of the discovered MAC address within the database. If such a MAC address already exists it ignores it, and goes on. If the MAC address is not already in the database, the new node is given a name, an IP address, and as all the information are available, the database is being updated with the new record. As we have now a consistent database, we can easily issue 'mkdhcpdconf' command in order to synchronize the DHCP server configuration and the database. Also, there is a possibility to replace existing nodes in the database. If so, the node name has to be given as the argument, so discover_node makes pretty the same thing as described above, but now it doesn't add a new node, but replaces an existing record in the database with the new one.

After the database and DHCP server configuration is over, the DHCP server is being restarted for the changes to take effect, and the queueing system is configured according to these changes.

For some generic configurations (as most ad-hoc clusters are built), this tool frees the administrator from learning how the SIS mechanism works, and speeds up the deployment process.

5. Conclusion

Open Source software is a great way to express creativity because other people from the OSS community can take advantage of one's work, adopt it to his/her own needs, add new functionality and publish these changes giving it back to the community for further development. This results in a distributed development model which has prooved to be very effective because all the prior work from all the OSS projects is taken into consideration before choosing the right solution. If none of the available solutions are satisfying, one can decide to develop something new, or (as the regular case is) extend the functionality of some existing project to adopt it to specific needs. That is exactly the way this project works, and we are proud to share the results of our efforts with the OSS community.

We expect to develop a toolset for easier cluster management, based on Debian GNU/Linux distribution. This involves development of automation mechanisms that provide a flexible platform for high-performacne computation tasks, but also provide a system-administrator to have a secure, easy to maintain, reliable and good supported cluster distribution. All the automation mechanisms and extra funtionality will be available as debian packages on the APT repository, and the vision for the future is to integrate our efforts with the Debian project.