Virtual Clusters for Hands-on Linux Cluster Construction Education

Paul Marshall[2], Michael Oberg[1], Nathan Rini[1],
Theron Voran[2], and Matthew Woitaszek[1]
matthew@ucar.edu, oberg@ucar.edu, nate@ucar.edu

[1] National Center for Atmospheric Research, Boulder, CO
[2] University of Colorado, Boulder, CO

Abstract. This paper presents the design and implementation of a virtual cluster hosting platform for hands-on Linux cluster system administration education. Using operating system and network virtualization, a collection of physical hosts is used to provide an arbitrary number of Linux clusters suitable for supporting instructional exercises in system administration and cluster technology. The virtual cluster provisioning software installs and manages the physical nodes that serve as virtual machine containers and also automates the deployment and contextualization of virtual cluster nodes. Ethernet 802.1Q VLANs are used to completely isolate each virtual cluster's internal node network, supporting the inclusion of multicast-based software as part of participant exercises. The managed hosting approach provides the ability to mediate and monitor each virtual cluster's access to the Internet external from the virtual cluster itself, supervise student exercise progress from an instructional host, and assert low-level control over virtual hosts to assist participants when necessary. This virtual cluster hosting platform was successfully utilized to provide 26 clusters for over 100 participants in a Linux Cluster Construction tutorial at Supercomputing 2009.

1 Introduction

In 2008, we developed a tutorial that introduces the basics of Linux cluster system administration. Targeted for power users that administer their own small cluster systems, the tutorial encourages an understanding of the fundamental technologies and techniques used to run clusters and supercomputers by guiding participants working in small groups through the process of turning a small collection of networked hosts into a cluster ready to run MPI-based parallel applications. Emphasizing hands-on low-level experience, the exercises start with establishing uniform SSH authentication and a shared NFS file system, continue with the installation of an MPI distribution and a batch scheduler, and then examine performance using the Ganglia monitoring system while executing common MPI and system benchmarks.

In order to present the tutorial in an interactive format such as a conference session, however, many clusters must be available at once – roughly one for

every four or five participants. Clusters of virtual machines (VMs) present the advantage of allowing participants to experiment with the entire Linux software stack while isolating groups from one another and eliminating the inefficiencies of working with physical hardware. For example, tutorial instructors can intervene below the virtualization level to correct situations that would otherwise require a visit to the machine room. To create these virtual clusters, we have designed and implemented a virtual cluster hosting platform that can quickly and easily create any number of arbitrarily complex Linux clusters using hosts that were formerly part of a single Linux cluster. The infrastructure manages both the servers used to host the VMs (referred to as containers) as well as VM-based cluster nodes themselves.

The remainder of this paper is organized as follows. Section 2 presents relevant background emphasizing the use of VMs in educational contexts and the construction of Linux clusters using automated deployment technologies such as bootable CDs. Section 3 presents the high-level design of the virtual cluster hosting architecture, and Section 4 describes its implementation. The paper concludes with future work.

2 Background and Related Work

Virtual machines have been widely embraced by the educational community as an effective mechanism to provide an isolated environment for student exercises. System virtualization implements a virtual duplicate of a real machine. This allows students to explore every layer of the software stack from the operating system to applications in an environment isolated from others and production services.

For example, Nieh et al. [10] leverage VMware VMs to provide an environment for students in an operating systems course to perform Linux kernel development. The virtual environment prevents OS-level bugs introduced by the students from impacting others. The authors note that the experience has been positive for both instructors and students. In particular, students appreciated the relevance of hands-on experience with a real-world operating system kernel. Similarly, Kneale et al. [8] developed VELNET, a virtual laboratory built on VMware Virtual Machines connected by a virtual network to replace expensive networking laboratories built with real hardware. VELNET provides a secure and safe environment for students to learn about and experiment with computer networks. Begnum et al. [3] describe the use of User-mode Linux (UML) in the University of Amsterdam's graduate-level network administration course, noting that providing each student with a custom network environment, comprised of real hardware, would be unrealistic. Students in the course create networks with multiple routers, bridges, and hosts implemented as Linux-based UML VMs.

An isolated computing environment can also be provided by using bootable CDs (often termed live CDs). Like VMs, booting a system from a preconfigured CD provides similar protection for the host machine from user modification. The system booted from a CD can be run entirely from CD, memory, or both,

without the need to modify the system's hard drive. If something goes wrong, a clean system is achieved by rebooting the workstation from the CD. Gray et al. [4] developed the "Bootable Cluster CD" (BCCD) aimed at educational environments. BCCD is able to use almost any networked workstation environment, from a desktop computer lab to laptops on a wireless network, and includes packages such as MPICH, LAM-MPI, PVM, and PVFS2. Also included are MPI debugging and performance tools along with linear algebra libraries and visualization programs. The software environment may also be customized before creating the CDs [5].

For the tutorial, it would have been possible for participants to construct clusters out of their laptops using live CDs or by running VMs on their laptops during the tutorial session. However, the use of remotely hosted virtual clusters allows participants to experience a realistic network environment on hosts with consistent and known performance characteristics. In addition, we have observed that participants often enjoy working in small groups and relying on each other to solve problems that they encounter during the exercise. We thus opted to provide remotely hosted homogeneous small Linux clusters for use in small groups with self-guided laboratory exercises. The virtual cluster hosting platform builds upon our earlier work with managed hosting environments [6], [9] to create virtual clusters that can be secured and managed in a uniform and automated fashion, giving each group of participants a consistent and independent experience.

The short-term demand for large numbers of networked VMs makes the use of virtual clusters for hands-on educational activities a potential candidate for deployment using infrastructure-as-a-service (IaaS) cloud computing [2] such as Amazon EC2 or a science-oriented cloud [7] enabled by Eucalyptus [11] or the Nimbus toolkit [1]. For example, our full-day tutorial at SC09 could have been hosted by Amazon for about $120 excluding initial development activities and data upload and storage charges. Nimbus also provides robust VM contextualization services to automate the construction of virtual clusters such as done manually here. We plan to investigate deploying tutorial clusters on cloud infrastructure using common cloud resource management toolkits but will probably continue to utilize locally-hosted systems for this purpose as long as they are available.

3 Design

The target cluster architecture for performing the tutorial exercises consists of a single Internet-exposed head node with a collection of compute nodes on a private (RFC-1918) network (see Figure 1). These target clusters are mapped onto the hosting platform using VMs for host virtualization and VLANs for network connectivity (see Figure 2). The container servers are networked using a separate VLAN using the untagged interface to support DHCP and PXE boot.

For the tutorial, each group is assigned a virtual cluster with one head node and four homogeneous compute nodes. As our hosting hardware consists of servers with two single-core single-socket processors, two VMs are placed on

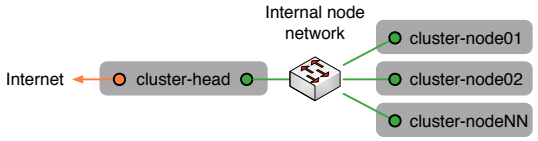

Fig. 1. Simple cluster architecture for tutorial exercises with a single exposed head node and compute nodes on a private network

each physical node. The nodes in each virtual cluster are distributed among separate containers to ensure realistic network access times between hosts. The head nodes are connected to a VLAN with external network access, and all nodes in a cluster are placed on a private VLAN with a private subnet. The VLAN numbers and subnet addresses are globally unique within the hosting platform so that all of the VLANs and subnets can be instantiated on a single host if desired. This feature was utilized to provide one machine for instructional monitoring of the tutorial participants' progress across all clusters.

Although the hosting platform was used to implement simple virtual clusters for the tutorials, its design and implementation are capable of creating and hosting heterogenous clusters including nodes with arbitrary network configurations and specific virtual host-to-container bindings. For example, a virtual cluster could be created with a head node, compute nodes, and storage nodes on specific containers, and the storage nodes could also be exposed to the Internet

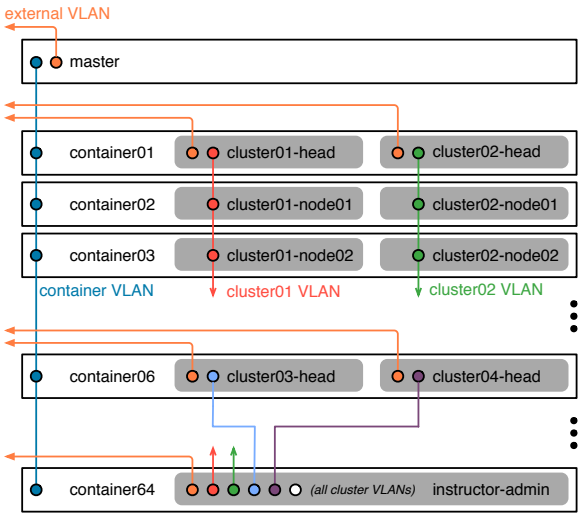

Fig. 2. Virtual cluster hosting architecture and network VLAN configuration

for file system sharing. Moreover, the contextualization process that performs node-specific configuration is customizable, allowing virtual clusters to be deployed as bare-bones networked hosts or fully functional clusters as required by the exercises.

4 Implementation

Simplicity in the design and operation of the implementation is of fundamental importance due to the nature of hosting real-time cluster construction tutorials. If something goes wrong during or immediately before a tutorial session, it is often faster to disable the affected systems and reconfigure the remaining systems (perhaps via a full reimage) than to debug a node's individual problems. Therefore, the hosting platform and its virtual machines are managed using as many general system administration tools as possible, with only essential complex operations automated using Bash or Perl scripts.

The physical hardware is a cluster of x86_64 nodes connected via gigabit Ethernet. A Myrinet interconnect was present but not used for this tutorial. The cluster hardware was homogenous except for slight variations caused by ongoing maintenance (e.g., node hard drive count and capacity variations), so the automation code required sufficient flexibility when manipulating storage devices. We chose Debian (Lenny) as the Linux distribution for the cluster and used its included Xen support.

4.1 Xen Container Installation and Configuration

Due to the age of the hardware being used to host the tutorial, the container systems were not expected to be reliable. A major requirement was therefore supporting the rapid deployment and configuration of an entire cluster of Xen nodes from a single master node, which must itself be interchangable with any other container node. All of the container nodes are installed with the cluster imaging services required for serving as the master, including DHCP, TFTP, and Apache (chosen over NFS for high-throughput image distribution), but these services are run only on the single active master node.

The first node was, of course, installed manually using a Debian network-based installation CD. To generate images for distribution to other nodes, a system image of the master node was taken using a live imaging script. Using bind mounts, only the relevant files are cloned, and temporary files and special devices are omitted from the image. The image is then saved in an Apache-readable directory. The node installation process uses a System Rescue Linux distribution chosen for its ease of use, allowing us to do a full bare metal install on generally any x86_64 system over the network. The installation script was modified to detect if an existing operating system was installed, and if so, wipe the existing boot records and reboot if necessary. Then, the script detects the presence of disk drives, creates a mirrored RAID if possible, partitions the disk using LVM, and formats the appropriate partitions using XFS. Finally, the

system image from the master node is downloaded and extracted onto the file system.

After the base image is installed, the servers are contextualized to make them members of the physical cluster running as Xen containers. The installation script uses netcat to contact a Perl daemon on the master node. The daemon modifies the PXE configuration to mark the node as installed so that it boots locally instead of running the installation process. Finally, the node reboots.

Distributing the operating system this way presented several security challenges. Because the master server must be able to control the other container servers, it needs to be able to connect to those servers using SSH keys. To retain the security of this privilege, SSH keys are not distributed as part of the container base image, but are copied by an administrator manually. This allows the SSH keys to remain safely on the master server without having to trust any of the other container-hosting nodes. At a lower level, a sophisticated attacker could use a man-in-the-middle-attack and corrupt the PXE booting process to take control of a node as it is being reimaged. Access is thus limited to the VLAN used for the VM container server boot process. The guest VMs are isolated with exposed or private VLANs, and the containers (except for the master) have no external connectivity and are firewalled at virtual network device level to stop all communication with the guest VLANs. The only exception to this is the use of masquerading for installation services described later (see Section 4.3). This does not protect against a container that has been compromised, but doing so requires compromising the master server or breaking out of a VM container.

Our experience with this distribution system demonstrated that TFTP became the bottleneck to server installation. This was not unexpected and could be mitigated with multiple servers as is common in large network-booted clusters. Full reimaging of the hardware servers was best performed in chunks of about 16 nodes and thus required about 40 minutes to reimage a 64-node cluster. Fortunately, the container servers are reimaged only occasionally – the distribution and provisioning of the VM images used to construct the virtual clusters does not rely on TFTP and is notably faster.

4.2 Virtual Cluster Node (VM Guest) Provisioning

The deployment of the VM hosts comprising the nodes in the virtual clusters is specified in two global configuration files on the master node and facilitated through VM installation scripts on each container. The first configuration file, the *cluster configuration*, contains a line for every virtual host including the hostname, its source image, its root password, and tuples of MACs, VLANs, and IP addresses for each desired network connection. The second configuration file, the *VM deployment configuration*, contains a line for every container and specifies which VM images should be deployed on each. In typical usage, the VM deployment script is executed on all of the container servers using a parallel shell (e.g., `pdsh`) or a script to fully populate the system all at once, or on one container at a time manually to fix problems. Each execution of the deployment script performs the following actions:

1. **Retrieve the current configuration files from the master node.** Thus, the entire virtual cluster specification is centrally controlled.
2. **Check for running VM nodes by name.** As the script is intended to be executed on a live system, the VMs running on each container are listed before the installation begins. VMs that are running are skipped, and any VMs that are not running but should be are installed or reinstalled.
3. **Prepare the hosting container network.** A list of VLANs required on the container is generated from the cluster configuration file and checked against the existing VLANs. Any missing VLANs or corresponding bridges are created and the firewall is configured appropriately. Unneeded VLANs are not removed in the current implementation, as this situation occurs rarely in practice.
4. **Prepare hosting container disk.** LVM volumes are created for the guest VM and formatted using XFS.
5. **Install the node image.** After mounting the VM node locally, the specified image is retrieved from the master server via HTTP and expanded into the newly mounted volume.
6. **Perform node contextualization.** System configuration files are generated to specify the host's identity in the target virtual cluster. For example, the node's network is configured, a `/etc/hosts` file is generated containing only the nodes in the virtual cluster, and preselected SSH host keys are installed.
7. **Boot the guest operating system.** The guest VM's file system is unmounted from the container and the VM is started.

This VM imaging process supports redeploying all or portions of the virtual cluster exceptionally quickly. As the process does not rely on TFTP or physical hardware reboots, and all data transmitted over the network is served by Apache, every container can download VM images from the master server at the same time. For example, if a change is made to the node prototype image, all of the virtual clusters can be regenerated. However, this feature is useful only before the tutorial session has started. Rapid redeployment also partially addresses the concern that hardware might fail during (or just before) the tutorial and the guest VMs would need to be moved to a different container. If a container demonstrates a problem, the cluster deployment configuration file is simply edited to move the guest VMs from that container to another, and the change is automatically effected when the deployment script is executed. This does not allow for live migration (which would require a shared file system), and thus does not account for maintaining student progress should a container fail and take its VM nodes with it, but it provides sufficient flexibility for the tutorial instructors to reconfigure the cluster up to the last minute before starting to ensure a fully operational system. For our SC09 tutorial, two nodes had hardware failures in the weeks before the tutorial session, but none failed during the session itself.

4.3 Providing Masqueraded Access to a Local Debian Mirror

One issue that frequently plagues clusters is that typical operating system software installation processes such as Debian's `apt-get` cannot function on compute nodes without network access. Although external access can be provided using NAT, relying on external OS distribution mirrors to satisfy hundreds of simultaneous identical installation operations during a time-limited tutorial session is risky at best. To avoid this problem, the tutorial VMs were configured to contact an internal Debian mirror when software installation is performed by participants as part of the exercises.

To provide access to the mirror from all of the virtual hosts, network masquerading was used from the private VLAN configured on each VM container (see Figure 3). Because the masqueraded IP is a potential route out of the otherwise isolated private VLANs, only specific targets are allowed (such as the path from each VLAN to the proper port on the mirror). One issue encountered with this scheme was ensuring that masqueraded IP accesses go through each host container and not over the network to another host masquerading on the subnet, especially because the same masqueraded IP was used for the mirror on each host. To solve this problem, Ebtables (an analogue program to iptables, used for managing Ethernet frame rules) was installed and configured with a special drop command on every bridge to avoid propagating ARP packets for the masqueraded IP. This solution works quite nicely, as all of the nodes can access the mirror directly instead of relying on a centralized node to route traffic. With this setup, any network service could be masqueraded and act as if it was a local service to each VLAN should the need arise.

5 Discussion and Future Work

The virtual cluster hosting platform was successfully used to provide 26 virtual clusters for over 100 participants at SC09 [12]. The container and virtual cluster management software was sufficiently robust to support all of the requirements for the tutorial, and we plan to expand the software to better support heterogeneous clusters by expanding and detaching the contextualization process. Thus, the infrastructure could be easily used to produce fully functional clusters in addition to the pools of networked hosts used for entry-level tutorials.

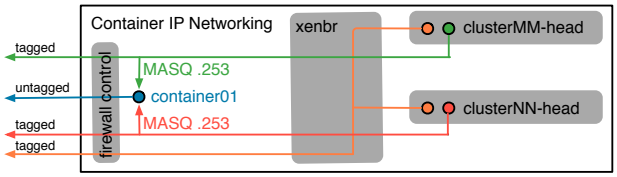

Fig. 3. Xen and container network configuration for IP masquerading to provide access to a local Debian installation mirror

The primary advantages to using the virtual cluster hosting platform are that we retain control of the container-based networking as well as the ability to quickly edit a virtual host's file system administratively if necessary. The ability to halt a virtual cluster node and mount the file system locally on the container is invaluable in a real-time tutorial. For example, in any exercise that requires participants to copy `/etc/passwd` and `/etc/shadow` files between nodes even once, at least one group typically accidentally overwrites a shadow file with a passwd file rendering a node or their entire cluster unusable. With the nodes hosted in virtual machines, the affected node can quickly be halted and the error corrected so the participants can resume the exercises with minimal delay.

Control over the VM hosting containers also allows us to enforce firewall rules or perform network monitoring outside the scope of the participants' control. During our SC09 tutorial, one group set their cluster's root password to "qwerty" in place of the strong randomly-generated initial password. The weak password was discovered by an otherwise routine password dictionary attack on the hosting subnet and the participant's cluster was quickly compromised. In the updated implementation, almost all outbound connectivity from the tutorial cluster head nodes to the Internet is restricted in order to eliminate the possibility of even the worst-case compromised participant node from being used to propagate external attacks against remote systems.

One feature of the cluster hosting platform is its flexibility in defining the network configuration for every virtual host. Using this facility, a specialized node for the course instructors was created with direct access to all of the nodes on all of the clusters that can be used to monitor participant progress during the exercises. For example, the instructors can inspect system files and services to determine if a group has started or successfully completed each exercise. To date, we have performed these checks using custom scripts and distributed shell invocations, but we intend to expand this system into a "tutoracle" featuring more formally defined exercise unit tests, a graphical display of group progress, and eventually the ability to offer evaluations and suggestions directly to users during the tutorial.

6 Conclusion

Our virtual cluster hosting platform provides the ability to create multiple independent Linux clusters on a collection of locally-managed Linux systems, essentially subdividing one large Linux cluster into many smaller clusters using VMs and VLANs. The management software automates the configuration of VM containers as well as VM instances serving as cluster nodes, making configuration changes and cluster deployments with varying degrees of contextualization rapid and painless. Successfully used to host an interactive introductory cluster system administration tutorial for a large audience, we intend to further refine the design and implementation and continue to use it for our cluster and system administration training activities.

Acknowledgments

We would like to thank Rich Loft, Anke Kamrath, Henry Tufo, and Irfan Elahi for their support of our activities in Linux Cluster Construction education and Guy Cobb for his assistance with preparing this manuscript. Support for this project was provided by the University of Colorado as well as NSF sponsorship of the National Center for Atmospheric Research.

References

1. Nimbus Open Source IaaS Cloud Computing Software. http://workspace.globus.org/.
2. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the clouds: A Berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.
3. K. Begnum, K. Koymans, A. Krap, and J. Sechrest. Using virtual machines in system administration education. In *Proceedings of the 4th International Systems and Network Engineering (SANE) Conference*, Netherlands, 2004.
4. S. M. Diesburg, P. A. Gray, and D. Joiner. High performance computing environments without the fuss: The bootable cluster CD. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 13*, page 252.1, Washington, DC, USA, 2005. IEEE Computer Society.
5. P. Gray, J. Chapin, and T. McNulty. Building of a GNU/Linux-based bootable cluster CD. In *Proceedings of the 7th LCI International Conference on Linux Clusters*, 2006.
6. B. House, P. Marshall, M. Oberg, H. M. Tufo, and M. Woitaszek. Grid service hosting on virtual clusters. In *Proceedings of the 9th IEEE/ACM International Conference on Grid Computing (Grid 2008)*, Tsukuba, Japan, September 2008.
7. K. Keahey, R. Figueiredo, J. Fortes, T. Freeman, and M. Tsugawa. Science clouds: Early experiences in cloud computing for scientific applications. In *Proceedings of Cloud Computing and its Applications*, Chicago, IL, August 2008.
8. B. Kneale, A. Y. D. Horta, and I. Box. VELNET: Virtual environment for learning networking. In *Proceedings of the 6th Conference on Australian Computing Education*, 2004.
9. D. Leverman, M. Oberg, H. M. Tufo, and M. Woitaszek. Experiences with managed hosting of virtual machines. In *Proceedings of the 10th LCI International Conference on High-Performance Clustered Computing*, Boulder, Colorado, March 2009.
10. J. Nieh and C. Vaill. Experiences teaching operating systems using virtual platforms and Linux. In *SIGCSE Bulletin*, 2005.
11. D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov. The Eucalyptus open-source cloud-computing system. In *Proceedings of Cloud Computing and Its Applications*, Chicago, IL, October 2008.
12. M. Woitaszek, M. Oberg, T. Voran, and P. Marshall. Linux cluster construction, Tutorial M06 presented at the 2009 ACM/IEEE Conference on High Performance Computing, Networking, Storage and Analysis (SC09), Portland, Oregon, November 2009.