International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:10, 2008

Scalable Deployment and Configuration of High-Performance Virtual Clusters

Kyrre M Begnum, Matthew Disney

***Abstract*—** Virtualization and high performance computing have been discussed from a performance perspective in recent publications. We present and discuss a flexible and efficient approach to the management of virtual clusters. A virtual machine management tool is extended to function as a fabric for cluster deployment and management. We show how features such as saving the state of a running cluster can be used to avoid disruption. We also compare our approach to the traditional methods of cluster deployment and present benchmarks which illustrate the efficiency of our approach.

***Keywords*—** Cluster management, clusters, high-performance, virtual machines, Xen

I. Introduction

VIRTUALIZATION has seen a recent, renewed and massive interest in computing. The technology is becoming increasingly integrated into operating system releases as well as hardware. Many fields, such as autonomic computing, service consolidation, security and education publish results that praise the benefits of virtualization. Recent publications discuss the prospect of high performance computing (HPC) coupled with virtualization[1][2].

Managing a virtual cluster is a two-fold challenge. First, there is the cluster of virtual machines that needs to be managed as an atomic unit, even though it is distributed among a set of servers. They have to be created, started, stopped and perhaps re-located or destroyed. Next, there is the network of physical machines that are to host the virtual machines. In the case of one virtual machine per physical machine, the administrator will have a total of twice as many operating systems to handle. One of the benefits of virtual clusters is the ability to run several different clusters at various constellations supported by the advent of multi-core CPUs. But with every new virtual cluster, there is an increase in the number of computer systems one has to manage. The challenge of virtual cluster management and deployment therefore inherits the challenges from physical clusters, including the problem of scalability. As the size of the cluster grows, the cost of the cluster and the complexity of the management grows proportionally. New tools are needed to ensure that the configuration and deployment of virtual clusters do not grow as the number of nodes increases. Ideally, such tools would be free.

Can scalable management of virtual clusters be achieved? What are the benefits of a virtual cluster? What design decisions and trade-offs must be considered? Also, what is the performance loss when running on a virtual platform? In this text, we demonstrate and discuss how the deployment of Xen virtual-machine-based clusters can be done efficiently in terms of management and design. We review a working virtual machine management solution as a framework for cluster management and present results and benchmarks from a 36-node cluster both when run on virtual and physical machines.

Authors are with the Faculty of Engineering, Oslo University College, Oslo, Norway (e-mail: kyrre.begnum@iu.hio.no, matthew.disney@stud.iu.hio.no).

The text is organized as follows: we first present a brief background on virtualization together with clustering. Next we detail how the physical machines are set up to accommodate virtualized clusters. In Sections IV and V we show how the virtualized clusters can be designed and managed efficiently. Performance benchmarks on both management metrics and the clusters themselves are presented in Section VI. A discussion on how traditional clusters may improve compatibility with virtualization concludes our work.

II. Background

Clusters are often deployed in a configuration consisting of a head node and compute nodes. A head node is a user gateway to the compute nodes, providing utilities to allow users to interact with the compute nodes. Additionally, head nodes provide certain important infrastructural services to the compute nodes, including network gateway, DNS, NFS, authentication service, and so on. For this reason, head nodes must have special knowledge (in terms of configuration) about the compute nodes and vice-versa.

A virtualized cluster scenario differs from other models in that it is actually two clusters instead of one. The foundation consists of the physical machines, which is not an HPC cluster in itself, but still offers a collective functionality to its user. In essence, its only job is to host and manage virtual machines. We call this the *virtual fabric layer*, and a physical machine which hosts virtual machines as part of this fabric is referred to simply as a *server*. When a server hosts a virtual machine, it provides the means to store the VM file system, start and stop the VM, and even monitor its resource usage. Most of these functions are available through the virtualization technology used. However, few software solutions offer the ability to manage large groups of virtual machines spread out over many servers.

Above the fabric layer, we find one or more clusters of virtual machines spread out among several servers. This is where the computational software resides, such as mathematical libraries and job-control mechanisms. It is possible to completely remove the configuration of the HPC cluster from that of the fabric layer. This gives the user the freedom to have complete control of the software versions on its own cluster while not having to focus on the underlying hardware and drivers.

Xen[3] is an open source virtualization framework which has gained significant popularity in recent years. Much of it is due to its impressive performance and availability features. A virtual machine running Xen can be visible on the network just like a physical machine.

III. Virtual Fabric Layer Deployment

We consider the virtual fabric layer to have a minimal OS installation and to offer a virtual machine management interface to the user. MLN (Manage Large Networks) [4] is a software

```
global {
       project oneVM
       $variable = 10.0.0.1
}

host one {
     xen
     memory 64M
     free_space 500M
     service_host xen2.iu.hio.no
     network eth0 {
           address 10.0.0.2
           netmask 255.255.255.0
           gateway $variable
     }
     users {
           matt jflaA/623(JBGS
     }
}
```

Fig. 1. An MLN project file describing a single VM.

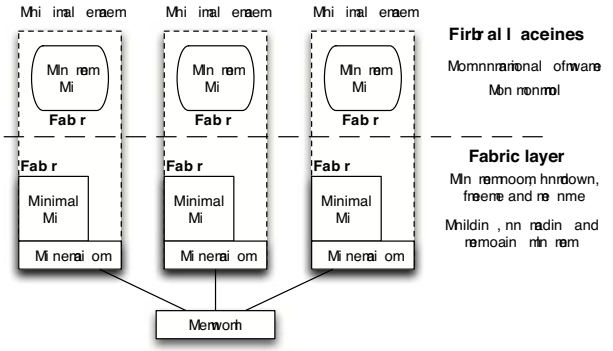

Fig. 2. The fabric layer is composed of physical machines running a minimal OS (dom0 in Xen terminology) installation together with virtual machine management software. The virtual machines (domUs) create the actual cluster where the computations take place.

tool designed for large virtualized networks using either Xen or User-Mode Linux[5] and can run as a service on most Linux distributions. Central management can be achieved, letting one MLN instance manage virtual machines on several servers at the same time as long as the MLN service runs on all servers.

A group of virtual machines, e.g. a cluster, is called a *project* in MLN terms. MLN builds virtual machines in a project using a cloning technique, where file system images are cloned and configured according to the project specification. Users can create their own file systems based on their specialized needs and those file systems are automatically configured into virtual cluster nodes based on a project configuration file. Figure 1 is an example of a simple MLN project.

The project file is written in declarative form. MLN supports general virtual machine settings, such as amount of memory and file system size, as well as system configuration variables including users, passwords and network settings. Most of the MLN syntax is self-explanatory, but notice the `service_host xen2.iu.hio.no`, which dictates the actual placement of the virtual machine. The syntax and configuration domain of MLN is expandable in the form of plug-ins written in the Perl programming language. Plug-ins can be used for two main purposes: 1) Introduce changes to the project before the project is built, e.g creating more virtual machines in the project, or changing some of their settings. 2) Add to the system configuration capabilities, e.g. configuring a special software package inside the virtual machine. We will demonstrate the use of both these features.

Interaction with MLN, such as creating, starting and stopping projects, is achieved via the command line. A distributed status command will collect status information from all the MLN daemons, including the amount of available memory for virtual machines as well as the status of currently running VMs.

MLN supports an upgrade function to projects, where the project can be modified based on changes in the project file. Examples of modifications can be increasing the file system size or memory, changing configurations, such as user passwords or even assigning a new service host, resulting in a migration of the virtual machine. More on MLN and its applications can be found in [6].

In terms of deployment, we modified an Ubuntu Linux [7] installer CD to include both Xen and MLN and to offer an automated install routine. A URL to a script can be supplied at boot time, which will make the installer download and run it at the end of the installation. This way it is easy to specialize the installation to local needs, e.g. distributing public keys for automated log-in. Using this CD it is straight-forward to create the fabric layer, as the configuration of these machines is relatively simple.

The virtual fabric layer is similar to a regular cluster. One server is selected as the main server which the other servers accept commands from. The user can log into this server and manage its projects.

IV. Designing Virtual Clusters

A traditional HPC cluster will be installed (directly or indirectly) from a set of install media, usually either located on CD-ROMs or in a networked location. During the install process the user can select from set of software packages that comes with the installation media. In a virtualized cluster, the approach is different. The user creates a file system which is to act as an MLN *template* for the virtual machine file systems. We installed a set of high performance computing (HPC) and clustering software tools on a virtual machine and then extracted its file system as a template. This tool-set included torque for resource management [8], an MPI implementation [9], [10], linear algebra libraries [11], and a benchmark utility [12]. The template acts as a virtual appliance. It can either be very small and specialized, or larger with a great number of software packages.

The software on a virtual cluster needs to be configured specific to each case according to local network settings, user accounts and the size of the cluster. These settings are usually written to the head node and propagated from there to the compute nodes. Instead of leaving the configuration of the compute nodes to the head node, we assume that the cluster has no internal configuration management system and let MLN build and configure them in the same process. Figure 3 is a complete project file for a 36 node virtual cluster.

The original MLN syntax is expanded with two plug-ins, both expressed in the `global` block. The first, called `autoenum`, is a general-purpose plug-in used to automatically expand the project with a defined number of hosts. (Notice, that the project contains no hosts in it to begin with). It is essentially an iterator, as it creates a number of hosts and increments the last octet of its IP address and the number at the end of the hostname. Its main use is to reduce the need

```
global {
       project mpi_cluster
       $console = screen
       $gateway_address = 128.39.74.1
       autoenum {
          superclass hosts
          numhosts 36
          address auto
          addresses_begin 150
          net 128.39.74.0
          service_hosts {
              #include /var/mpi/servers.txt
          }
       }
       cluster {
          head node1
       }
}

superclass hosts {
        term $console
        template ubuntu-server-cluster.ext3
        size 700M
        memory 128M
        xen
        lvm
        network eth0 {
              netmask 255.255.255.0
              broadcast 128.39.74.255
              gateway $gateway_address
        }
        users {
              disney VUmaegY1g8q4I
        }
}
```

Fig. 3. AN MLN project file describing a 36 node cluster of Xen VMs.

for repetitive entry and configuration maintenance for the administrator, while abbreviating the configuration and keeping large number of hosts consistent in terms of configuration. Additionally, the number of nodes in the cluster is arbitrary to the complexity of the project file, as one only has to increase the number at `numhosts` in the project file.

The second plug-in, called `cluster`, is specialized to the template file system we have created. It assumes an installation of torque and then creates the appropriate system-level torque configuration necessary for the head node to distribute jobs to the compute nodes. This includes an automatic generation of a file containing a list of the network addresses for all nodes. Additionally, the cluster plug-in configures a network file system (NFS) server on the head node to export user home directories to all nodes. On all nodes in the project, the automounter is configured to mount home directories from the head node. This all happens automatically, independent of which or how many nodes are specified. The only parameter needed for that plug-in is an indicator of the head node. Other plug-ins can be written that are specialized to other software packages or to perform other administrative functions.

A superclass, `hosts`, is defined for this project. It defines the general configuration of a virtual machine. A superclass will not result in a virtual machine, but hosts in a project may inherit settings from a superclass. We see that the amount of memory, users, file system-size and general network settings all are defined in the superclass. The *autoenum* plug-in will cause all hosts to inherit from that superclass, making their configurations consistent.

The list of servers, i.e. the physical nodes that will host the virtual machines, is in the text-file `/var/mpi/servers.txt`. It could just as well be in the project file, though the result would be unnecessarily long and less readable. The syntax of the fabric list is both simple and flexible. One host address

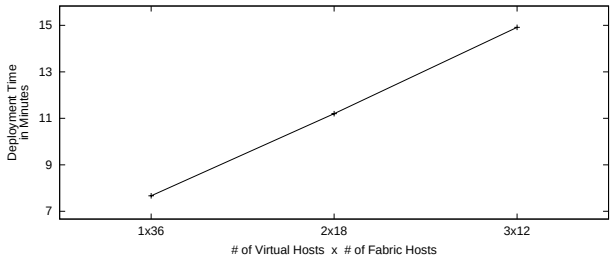

Fig. 4. This graph shows the deployment time for virtual clusters in various configurations to a fabric consisting of single-processor hosts. In all cases, a cluster of the same size is created but with a different distribution. 1x36 means that there is one virtual machine per server.

per line, with an optional parameter (separated by whitespace) providing the number of virtual machines that should be hosted on that particular node.

A remarkable feature about the combination of these configuration language capabilities is that the complexity of the specification is static despite the growth of the size of the cluster. Whether the cluster is a single node or hundreds, the project file need not grow in size at all.

V. Virtualized Cluster Management

The deployment of a virtual cluster is done with the command:

```
mln build -f mpi_cluster.mln
```

MLN will contact all the physical servers involved in the MLN project and handle the distribution of the project. The creation of a distributed project of this sort is handled in parallel by all the servers. Consequently, the build time is not dependent on the number of nodes in the cluster per se, but on the number of nodes per physical server. As an example, building a 36 node cluster spread out over 36 physical servers will take as long as the time it takes to build one node. Building the same virtual cluster on 18 physical servers, meaning two virtual machines per server, will take longer, as we can see from Figure 4.

A project can be started and stopped atomically with a single command:

```
mln start -p mpi_cluster

mln stop -p mpi_cluster
```

Xen supports stopping a virtual machine and saving its state. This can be harnessed by MLN to snapshot the entire cluster while it is running and to restore it later. The command for MLN to attempt to save the entire projects state is:

```
mln stop -p mpi_cluster -S
```

We found that this does not disrupt the computation of a running cluster. MLN also supports migration of virtual machines between physical servers. The typical form of migration is to shut down the virtual machine first, and then to boot it at its new destination. This has the benefit, that one can migrate between compatible CPU architectures and change other host parameters in the process. Virtual machines that have been saved can migrate between machines of the exact same CPU architecture.

The combination of migration and freezing a project is interesting for clusters. In our experiments, we were able to start a job in a cluster, freeze the cluster, migrate some of the

nodes to different servers, and continue it again to complete the job. This *disruption avoidance* technique can be used to remove failing physical machines from a cluster without having to cancel the job that is running. It can also be used to upgrade the hardware of a cluster without having to re-install the software or disrupt the actual computations.

The templates used by the project have to be available to all the servers before building it. This can be achieved, either by manually copying them to the servers beforehand, or to share a file system with the templates over the network. The latter approach is more practical if the templates are updated frequently, but it will impose an impact on the network during the build process.

Long-term management is possible through the `mln upgrade` command. Through it, parameters can be adjusted on projects that are already built. The same mechanism is used to migrate virtual machines between servers. The MLN command `mln reconfigure` can be used to bring a project back to its project description and intended configuration. Thus it can function as a tool for maintenance of a cluster, though that is not its main purpose. Dynamic system configuration tools, such as cfengine[13], can be used inside a cluster and be set up through a MLN plug-in. A maintenance scheme that is internal to the virtualized cluster can be beneficial if the users of the cluster have no access to the virtual fabric layer itself and therefore have no access to MLN.

VI. Performance Benchmarks

While work has already been done to show the viability of virtualization from a performance perspective [1], any recommendation of a new way to create and manage clusters would be incomplete without a look into both performance and the benefits gained from proposed methods.

A popular benchmark for clusters is Linpack, a set of Fortran routines for solving linear systems [14]. Specifically, an implementation called the High Performance Linpack (HPL) benchmark [12] is used in comparing supercomputers in the form of the Top500 list, published twice annually [15]. Therefore, HPL is a suitable tool by which we can size virtual clusters in various configurations and compare them to physical clusters. We consider the following:

- How does a 36-node Xen virtual cluster compare to a 36-node physical cluster using identical HPL configuration?
- What is the performance degradation of using 2 or 3 Xen virtual machines per physical host when using an identical HPL configuration?
- How does an optimized Xen virtual cluster HPL benchmark for 64 virtual nodes in a configuration of 2 hosts per each physical host compare to an optimized HPL benchmark of 36 single-processor physical nodes?
- What are the performance advantages of a dual-core processor architecture in the context of HPC applications? What is the performance degradation of running 2 or 3 competing virtual HPC nodes on a multi-core server? What is the difference in performance when allocating virtual CPUs in different quantities when running competing virtual clusters?

A. Single-processor virtual fabric

Our initial HPL configuration includes a problem matrix order N of 16000, a partition blocking factor NB of 80, a number of process rows P of 6, and a number of process columns Q of 6. These tests were run on on a laboratory of 36 hosts with 2.8 GHz Intel P4 processors, 512 MB RAM, a standard 7200rpm harddisk, and 100BaseT Ethernet.

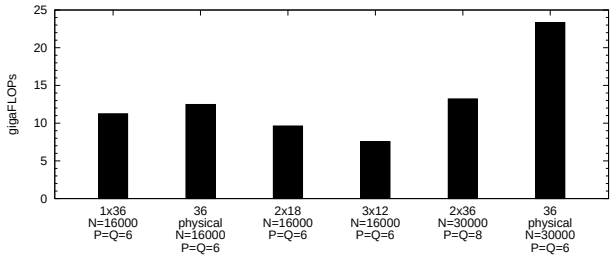

Fig. 5. Performance in gigaFLOPs of each cluster configuration benchmarked using HPL.

The virtual cluster of 36 nodes (1 node per physical host, i.e. 1x36) achieved 11.25 gigaFLOPS (units of billions of floating point operations per second) while the physical cluster obtained 12.38 gigaFLOPS. The physical cluster was approximately 10% faster.

Leaving the same HPL configuration in place, we tested 2 new configurations of virtual cluster deployment to the fabric. It is interesting to note that due to the separation of the virtual hosts from the fabric, the existing virtual machines could easily be redeployed using MLN in new combinations per physical host without the need to rebuild the cluster or reconfigure our MPI environment. Results can be seen in Figure 5, comparing the plots for 1x36, 2x18, and 3x12. Our results suggest that performance difference increases as competing virtual nodes are added to the fabric.

To further examine the performance degradation due to competing virtual machines, we configured our virtual cluster so that each server hosted 2 virtual nodes and changed the HPL configuration accordingly. The results of that test can be compared with a similar benchmark run on the physical servers (seen in Figure 5, contrasting the data for 2x36 and 36 physical nodes). The performance of the physical cluster was almost twice that of the virtual cluster.

B. Dual-core virtual fabric

While most of our laboratory equipment is single-processor and allows us to adequately pursue issues of scale, the emergence of multi-core processors is significant and should not be overlooked when discussing matters of economy in virtual clusters. Accordingly, we performed HPL benchmarks (in a configuration where P and Q are both 1) on a server with a dual-core 2GHz AMD Pacifica and 2GB RAM.

Our results are demonstrated in Figure 6. We found that the virtual node marginally outperformed the physical CPU and that there was little difference between a virtual node running on one versus two logical CPUs. Additionally, we found that two virtual nodes on the dual-core server performing the HPL benchmark at the same time each performed as well as the physical host. Predictably, running three virtual hosts on the dual-core server exhibited a substantial drop in performance. In the first case, each host was designated to use a single logical CPU. One virtual node approached the performance of the previous tests, while the other two virtual nodes each demonstrated half the gigaFLOPs of the successful node. In the second case of three virtual nodes on the dual-core server, the

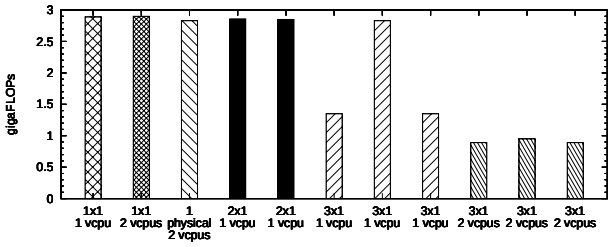

Fig. 6. Results from HPL benchmarks on various single-node cluster configurations (using a P and Q of 1) on a dual-core server. Plots with identical shading represent benchmarks run simultaneously, competing for physical resources.

virtual hosts were each assigned *two* logical CPUs. The nodes this time performed much worse.

These tests suggest, unsurprisingly, that the best resource distribution for a set of servers with multiple logical CPUs is to run n number of virtual hosts per server, where n represents the number of logical CPUs per server. Our findings that Xen performs suitably for HPC are consistent with and complement the initial findings of Barham et al [3], as their work uses benchmarks that are not specific to HPC.

C. Management Metrics and Benefits

We analyzed the benefit of a tool, MLN, to manage whole networks of virtual machines in regards to deployment time, availability, mobility, and simplicity of management and configuration (including any increase in complexity as the cluster increases in size).

Our customized installer CD installed hosts in approximately 10 minutes each. The complete installation of 36 physical nodes for our virtual fabric required less than 2 hours, even when manually moving CDs and initiating installations. In an environment with PXE-based network install capabilities and/or network infrastructure faster than 100 Mbps, deployment of the virtual fabric would take less time.

With the fabric in place, we were ready to deploy clusters. A virtual cluster of 1x36 was deployed in 7 minutes, 40 seconds. A cluster of 2x18 deployed in 11 minutes, 12 seconds; 3x12 in 14 minutes, 55 seconds. This indicates a degradation in deployment efficiency relative to number of virtual hosts per single-processor node but shows that a virtual cluster can be deployed extremely quickly using MLN. The time it takes to build a virtual machine depends primarily on the size of the file system template and the speed of the storage device. The template used in our cluster was around 620MB. For completeness, consider an additional 2 minutes for the modifications necessary to MLN configuration files each time the cluster configuration was changed.

In a physical cluster, the removal of a node or the full cluster for maintenance can be disastrous for long-running jobs. Such maintenance can result in the necessity to repeat a full lengthy job. This challenge, *disruption avoidance*, is effectively addressed by a Xen virtual fabric created and managed by MLN. We tested three scenarios for disruption avoidance:

- Taking an entire virtual cluster offline.
- Taking a single node offline.
- Relocating a single virtual machine to a different physical node.

In each case, the job completed successfully without error, any need for special adjustment by the user, or any kind of interruption planning whatsoever. Such options for availability alone may justify any performance degradation that may be experienced when using a virtual rather than physical cluster. This shows it is possible to execute a two-phase hardware deployment to upgrade all the physical hosts constituting the virtual fabric without compromising the completion of any jobs running on the virtual cluster hosted on that fabric. Such an upgrade could be so dramatic as to include a compatible systems architecture change, provided the virtual fabric remains intact.

VII. Discussion

A traditional approach of installing clusters has been to install the head-node, and then to leverage it to install and configure the compute nodes. When is it favorable to first deploy a fabric for virtual machine administration, next specialize a file system template and finally spread it out among the servers? Clearly, the management aspects and benefits come first into play when several virtual clusters are to be managed, either at the same time or in succession. Such cases would include:

- An organization that does not own its own cluster, but rents access to a virtual fabric from a third party.
- Someone who would like to easily interchange the currently running cluster with a different configuration (e.g. applications, memory, network information, etc...).
- Developers of cluster and HPC software who are interested in deploying test-scenarios quickly.
- Administrators who face the challenge of maintaining a cluster for a variety of customers, which have application requirements that conflict with each other.

With the two plug-ins, `autoenum` and `cluster`, we were able to define the settings of a cluster easily, and to re-use it in many scenarios. The `autoenum` plug-in is general purpose and can be re-used for most scenarios. The other plug-in concerns configuration of the job-control software, and might be replaced based on the software on the given template. Rather than relying on specialized plug-ins, it is also possible to create file system templates that hard-code the needed functionality.

Once the fabric is running, we were able to build new clusters in less than ten minutes. Saving the state of a running cluster takes a only a few seconds. While booting and shutting down a cluster takes somewhat longer, it is still achieved within minutes, making transitions from one cluster to another very fast and convenient. The question remains whether this is a big enough gain for a technician who has to specialize a template and perhaps write a plug-in for it.

Template distribution is a new type of challenge regarding clusters, but is a relatively known problem generally. It is also apparently the only central bottleneck in our proposed deployment and management methods that could restrict scalability. There are several approaches for addressing the problem; one would be to have a network file system that contains the templates, thereby removing the synchronization problem, but inducing a network cost at build time. Another approach would be to do a manual synchronization of the templates using tools such as the bittorrent system to spread them rapidly. Multicast communication, coupled with an appropriate application to handle synchronization issues, could also be used to quickly distribute large files to many hosts.

MLN and Xen can also be installed on other Linux machines that are not dedicated to a cluster and still participate as fabric

servers. One scenario for this usage is that regular workstations can act as part of a virtualization fabric during off-peak time periods when they are under-utilized.

In comparing our work in the context of deployment and management of virtual Xen clusters built from commodity hardware to other popular cluster deployment suites, five offerings stand out: FAI[16], OSCAR[17], Rocks[18], VMPlants[2], and XenEnterprise[19].

FAI, OSCAR, and Rocks target the bootstrapping of clusters from bare-metal, though each are capable of deploying Xen nodes in different ways. OSCAR can apparently now install a Xen head node to be used for testing, but not compute nodes. Rocks reports alpha capability for installing virtual compute nodes, which can be managed as regular Rocks compute nodes, but the head node must be a physical host. FAI has been leveraged to deploy Xen nodes, though FAI does not include an ongoing system configuration component; it only manages configuration at deployment time. Each of these solutions are limited to specific GNU/Linux distributions, while MLN can support any distribution through its modular plug-in architecture. FAI and Rocks are apparently too different in architecture to ever be integrated with MLN, but OSCAR's use of filesystem images makes it a good candidate for integrating with our methods. Indeed, informal OSCAR suggestions for future work on that project indicate similar goals to our management methods.

The VMPlants[2] work is similar to ours, though the software itself is not available to the general public. Accordingly, we are unable to make sufficient comparisons to our work.

The currently available software most closely related to our method is the newly released XenEnterprise[19]. The XenEnterprise product appears to offer many similar benefits to our methods (as well as a graphical management interface), though it is neither open-source nor modular. It is dedicated to the Xen platform, which is certainly the focus of our work but other competitive virtualization architectures may emerge, and MLN will be suitable to deploy and manage those new virtualization technologies as well. Nonetheless, there may be lessons to learn from the XenEnterprise product, including interface improvement and more extensive support of Xen features.

VIII. Conclusions and Future Work

Scalability is a core challenge of cluster management. Our approach offers a simple way to manage virtual clusters so that the cluster administrator is not affected by issues of scale. This method was shown to be scalable in terms of cluster design; a cluster can be designed accurately, simply, and independent of the number of nodes in the cluster. Deployment of a virtual cluster is automatic, the time required for deployment of all nodes is nominal, and future work will further improve the scalability of filesystem template distribution. Large distributed clusters can be easily started, stopped, and even suspended in a running state from a single point of user interaction. Using MLN's modular architecture, we extended the virtual network management tool to suit specific requirements, in this case HPC cluster configuration. Leveraging Xen as a virtualization platform has enabled the running of two or more clusters on the same hardware, which we showed is clearly advantageous on multi-core computers. Finally, financial scalability is achieved as well, since our entire approach was built from free and currently available software.

MLN supports only Xen and User-Mode Linux as virtualization frameworks. Future work will investigate how other popular technologies, such as VMWare[20] can be utilized in the same context.

Acknowledgments

The authors extend their thanks to Prof. Mark Burgess and Dr. Tore Jonassen for helpful discussions and to the IT department at the Oslo University College Faculty of Engineering for making the equipment available. This work is supported by the EC IST-EMANICS Network of Excellence (#26854)

References

[1] Mark F. Mergen, Volkmar Uhlig, Orran Krieger, and Jimi Xenidis. Virtualization for high-performance computing. *SIGOPS Oper. Syst. Rev.*, 40(2):8–11, 2006.

[2] Ivan Krsul, Arijit Ganguly, Jian Zhang, Jose A. B. Fortes, and Renato J. Figueiredo. VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing. In *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, page 7, Washington, DC, USA, 2004. IEEE Computer Society.

[3] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM Press.

[4] K. Begnum and J. Sechrest. The MLN homepage. http://mln.sourceforge.net. Last accessed August 28, 2006.

[5] Jeff Dike. A user-mode port of the Linux kernel. In *Proceedings of the 2000 Linux Showcase and Conference*, October 2000.

[6] Kyrre Begnum. Manage large virtual networks. In *To appear in: Proceedings of the 20th Large installation system administration conference*. USENIX, 2006.

[7] Ubuntu Linux. http://www.ubuntu.com/. Last accessed August 27, 2006.

[8] TORQUE Resource Manager 2.0. http://www.clusterresources.com/pages/products/torque-resource-manager.php. Last accessed August 27, 2006.

[9] Greg Burns, Raja Daoud, and James Vaigl. LAM: An Open Cluster Environment for MPI. In *Proceedings of Supercomputing Symposium*, pages 379–386, 1994.

[10] Jeffrey M. Squyres and Andrew Lumsdaine. A Component Architecture for LAM/MPI. In *Proceedings, 10th European PVM/MPI Users' Group Meeting*, number 2840 in Lecture Notes in Computer Science, pages 379–387, Venice, Italy, September / October 2003. Springer-Verlag.

[11] R. Clint Whaley and Jack J. Dongarra. Automatically tuned linear algebra software. In *Supercomputing '98: Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM)*, pages 1–27, Washington, DC, USA, 1998. IEEE Computer Society.

[12] A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary. HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. http://www.netlib.org/benchmark/hpl/. Last accessed August 27, 2006.

[13] Mark Burgess. Cfengine: a site configuration engine. In *Computing systems, Vol8, No. 3*. USENIX, 1995.

[14] Jack Dongarra. The LINPACK Benchmark: An Explanation. In *Proceedings of the 1st International Conference on Supercomputing*, pages 456–474, London, UK, 1988. Springer-Verlag.

[15] Top500 Supercomputing Sites. http://www.top500.org/. Last accessed August 27, 2006.

[16] FAI - Fully Automatic Installation. http://www.informatik.uni-koeln.de/fai/. Last accessed August 28, 2006.

[17] OSCAR (Open Source Cluster Application Resources). http://oscar.openclustergroup.org. Last accessed August 28, 2006.

[18] Rocks Clusters. http://www.rocksclusters.org/. Last accessed August 28, 2006.

[19] XenSource - XenEnterprise. http://www.xensource.com/products/xen_enterprise/. Last accessed August 28, 2006.

[20] VMWare - An EMC Company. http://www.vmware.com. Last accessed August 28, 2006.