Virtual Cluster Management with Xen

Nikhil Bhatia and Jeffrey S. Vetter

Future Technologies Group
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN, USA 37831
{bhatia,vetter}@ornl.gov

Abstract. Recently, virtualization of hardware resources to run multiple instances of independent virtual machines over physical hosts has gained popularity due to an industry-wide focus on the need to reduce the cost of operation of an enterprise computing infrastructure. Xen is an open source hypervisor that provides a virtual machine abstraction layer which is very similar to the underlying physical machine. Using multiple physical hosts, each hosting multiple virtual machines over a VMM like Xen, system administrators can setup a high-availability virtual cluster to meet the ever-increasing demands of their data centers. In such an environment, the Xen hypervisor enables live migration of individual virtual machine instances from one physical node to another without significantly affecting the performance of the applications running on a target virtual machine. This paper describes a scalable Virtual Cluster Manager that provides such application agnostic cluster management capabilities to the system administrators maintaining virtual clusters over Xen powered virtual nodes.

1 Introduction

Recently, virtualization of hardware resources to run multiple instances of independent virtual machines over physical hosts has gained popularity as a potential solution to an industry-wide focus on the need to reduce the cost of operations of enterprise and scientific computing infrastructures [1]. Industrial and academic installations of these clusters can contain thousands of physical nodes. Such clusters are prone to changes in the availability of physical resources due to unfortunate conditions like node failure due to overheating, or system administrative tasks like dynamic load balancing and preventative maintenance. Meanwhile, applications and users on these infrastructures expect highly available, reliable, and transparent operation.

Accordingly, there has been a prolific rise in the research and development of Virtual Machine Monitors (VMMs) that employ virtualization at different levels in the system software stack (e.g., Xen [2], QEMU [3], VMWare [4], User Mode Linux [5]). One such system exemplifying this trend is Xen: an open source hypervisor that provides a virtual machine abstraction layer which is very similar to the underlying physical machine. Xen's type of virtualization, often termed as para-virtualization,

© Springer-Verlag Berlin Heidelberg 2008

overcomes the typical performance loss due to virtualization by maintaining hardware information per individual VM inside the VMM interface. On the other hand, this para-virtualization requires substantial modification to the hardware dependent code in the target operating systems running over the virtual machine. This virtualization provides many benefits including the ability to save the entire VM to persistent storage, and then restart it later on the same, or on a *different,* physical host [6]. When using large pools of physical hosts, with each physical host containing multiple virtual machines, system administrators can easily construct a high-availability virtual cluster to meet the ever-increasing demands of their data centers and scientific computing clusters. With these virtualization capabilities, system administrators can handle various management tasks, such as dynamic load balancing of virtual nodes, and eviction of applications from a physical nodes to prepare for maintenance or to preempt a expected failure, transparently to the individual applications running in the virtual machines.

This paper describes a Virtual Cluster Manager that provides such application-agnostic cluster management capabilities to the system administrators maintaining these virtual clusters. We demonstrate its capabilities that include remote migration and dynamic load balancing of VMs across a pool of physical nodes. This framework manages a virtual cluster powered by the Xen virtual machines across multiple physical nodes. Our prototype-- the Xen Virtual Cluster Manager, henceforth referred to as XCM -- provides several of these features. First, it provides an overview of the performance of virtual machines to the system administrators on a per physical node basis. Second, it provides capabilities to initiate administrative tasks like automatic load balancing of VMs across physical resources based on their utilization and eviction of VMs from physical nodes in preparation for maintenance at arbitrary intervals.

Following is the organization of this paper. Section 2 describes the Virtual cluster organization. Section 3 describes the related work in this area. Section 4 describes the XCM framework which is divided into two parts: the XCM client and the XCM Daemon. Section 5 describes the implementation decisions we made during the course of this project. Section 6 provides the reader with an overview of our framework in action. Finally, we discuss out future research goals in this area in section 7 and conclude in section 8.

2 Virtual Clusters

Virtual clusters [7] [8] are comprised of several physical nodes running a virtual machine monitor (e.g. Xen) and hosting multiple virtual machines (often referred to as DOM Us) which in turn are running several user-level applications. Such an infrastructure can easily run into managing hundreds (or even thousands) of virtual machine instances running over tens (or hundreds) of physical nodes. These configurations are beneficial for large scale computing facilities because it reduces the cost of operation of these data centers by replacing the need for hundreds of physical servers by having hundreds of virtual machine over tens of physical servers.

Also, this infrastructure can also be useful in managing distributed memory cluster environments where each physical node can host multiple virtual machines, with each hosting several distributed memory application processes (e.g., MPI tasks), depending on the application level topology.

Due to increased need of to improve the performance of virtualized servers, many microprocessor vendors like Intel and AMD are providing hardware support (Intel-VT and AMD *Pacifica*) for maintaining several hardware states for several virtual machines in the microprocessor itself. This would reduce the changes made to the guest operating system servicing a virtual machine to and at the same time reap the benefits of para-virtualization. A physical node hosts a fully loaded operating system which consists of device drivers for I/O devices and network interfaces. Such a fully loaded host operating system is often referred to as the Host Operating System or DOM 0. As described earlier, the VMM creates an abstraction of the hardware resources (like the CPU, the memory, the Network Card and the I/O devices) per virtual machine instance. Such a virtual machine instance is serviced by an Operating System which is often referred to as a Guest Operating System or a DOM U. The DOM U uses the device driver interface provided by DOM 0 to access the hardware abstraction provided by the VMM. The VMM is responsible for translating the per-domain system call to access the actual hardware through the hypercall interface.

3 Related Work

The Virt-manager [9] Project from Redhat has provided an infrastructure to build a Virtual Node Manager which provides the system administrator with a GUI to display the performance information of the virtual machines along with all of its domains running on a physical node under a VMM. It also provides a Virtual Machine configuration window through which the system administrator can configure and create new Virtual machines on a physical node. Their goal is to provide a per physical node administration tool which can help the system administrators to monitor the exiting virtual machines and create new ones depending on a certain configuration. In contrast, our efforts are concerned with the managing of a cluster of physical nodes, each hosting multiple virtual machines and providing an infrastructure that encapsulates VMM level details in the system management tasks like dynamic load balancing, node maintenance, and preemptive node failure from the system administrator. It also provides a framework for designing new load balancing algorithms and resilience policies for handling node failures based on a cluster's telemetry data.

The libvirt API [10] project is a step forward in unifying the interface to gather the performance metrics of a virtual machine running over the hypervisor layer of a VMM. It provides an API in C which hides the hypervisor layer abstraction from the tool builders who want to develop performance analysis tools and cluster managers over many hypervisor layers. Our work, in contrast, can be built on top of the libvirt API and target multiple VMMs like Xen, qemu, VMWARE etc. Currently, our framework works only with Xen powered clusters.

4 Overview of Xen Virtual Cluster Manager

XCM is a Virtual cluster wide resource manager for managing Guest Domains running as Virtual machines over physical nodes. The XCM is built using a client-server model where it runs as a client on a remote node called as the Monitoring Station. Each physical node runs a XCM daemon which gathers performance metrics for the virtual machines running on that physical node. The XCM daemon uses the hypervisor interface to gather the performance metrics. The XCM daemons connect to the XCM client on the Monitoring Station. Then, the XCM client gathers the performance metrics from physical nodes using this daemon and aggregates the cluster-wide information in its internal data structures. The XCM client is built using C++ over the wx-Widgets GUI toolkit. The XCM daemons are built using C over the virtual machine monitoring libraries built over the Xen hypercall interface.

4.1 XCM Client

The XCM client runs on the monitoring station. The XCM client connects with the XCM daemons running on the physical nodes. The XCM client gathers the performance metrics from the XCM daemons at regular intervals and arranges them into a per-physical node per virtual-machine information. The time intervals at which the client gathers the results from the daemons can be configured by the user. The internal data structures of the client are used to display the performance of virtual machines on a per-physical node granularity in two views: the summary view and the detailed view.

The system administrator using the framework can study the performance metrics to make intelligent decisions about what administrative tasks need to be performed for optimal utilization of the physical resources. The administrator is also given an option of performing those tasks via the framework. Currently, three types of administrative actions can be performed by the administrator. These are dynamic load balancing, preemptive node maintenance, and live migration of individual virtual machines (henceforth referred to as domains) across physical nodes (henceforth referred to as nodes).

Views. XCM client can display virtual cluster-wide information in two views. The summary view displays coarse-grained information about the virtual machines running on each of the virtual nodes in a cluster. The detailed view displays the fine-grained per virtual machine/per virtual node information for all the virtual machines in a cluster.

Actions. The XCM client provides the system administrator with three types of actions that can be performed for managing the virtual cluster: dynamic load balancing, preemptive node maintenance, and live migration of individual domains. First, the XCM client can perform live migration of domains across virtual nodes. The information required to perform live migration is entered through the GUI interface. Second, the XCM client can perform automatic load balancing by migrating domains across all available physical nodes. Currently, we are investigating the policies on which we will determine our migratory decisions. Finally, the XCM client can also enable the system administrator enables the clean shutdown of a physical node by migrating all it's domains to other physical nodes in the cluster when that node is being taken away for maintenance.

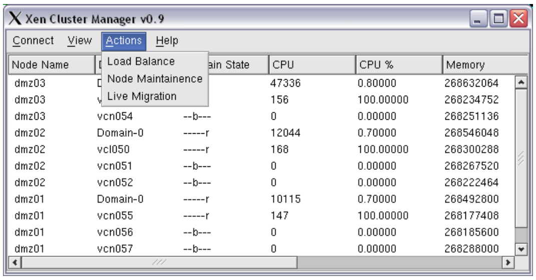
4.2 XCM Daemon

The XCM daemon runs on the physical nodes' host operating system. The daemon gathers performance metrics of all the virtual machines using the hypercall interface of the VMM. Currently, we gather per-VM performance metrics like total number of domains hosted by the VM, the state of each domain, the memory that has been allocated to each domain, the current CPU utilization of a VM, information about the number of virtual CPUs allocated to a VM, the number of virtual block devices and the information about the virtual network interface.

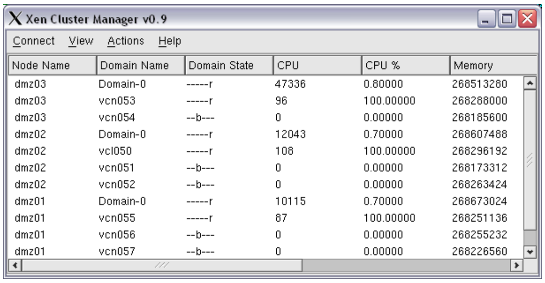
This information is obtained at regular intervals from the Xentop utility that is shipped along with the Xen distribution. The Xentop utility is built over the Xenstat library that provides an API for accessing the above mentioned performance metrics from the hypercall interface of the Xen VMM. The daemon is invoked using the command line interface and it connects to the monitoring station using TCP/IP sockets. The time interval that elapses between two successive performance data collect operations can be configured by the administrator using command line options. The daemon communicates the local performance data to the XCM client which organizes that data into its local data structures and displays that information on its GUI.

5 Xen Cluster Manager in Action

The XCM client provides the System Administrator of a virtual cluster with a capability to perform various administrative tasks like automatic load balancing, clean shutdown of nodes while system maintenance and live migration of individual domains from one node to the other across the virtual cluster. We have utilized the XCM client to manage a Virtual Cluster consisting of 3 physical nodes and hosting 8 virtual machines in all. The physical nodes are dual core AMD opteron machines running XEN-3.0.3 as a VMM. We have performed system administration tasks like live migration of domains across the three physical nodes, automatic load balancing of the virtual cluster based on a simple load balancing strategy, and clean removal of physical nodes during system maintenance. These experiments have been done as a proof of concept of our framework and demonstrate the scalability of our framework. The policies for various administrative actions are discussed next.



Node Name	D	ain State	CPU	CPU %	Memory
dmz03	D		47336	0.80000	268632064
dmz03	v		156	100.00000	268234752
dmz03	vcn054	--b---	0	0.00000	268251136
dmz02	Domain-0	-----r	12044	0.70000	268546048
dmz02	vcl050	-----r	168	100.00000	268300288
dmz02	vcn051	--b---	0	0.00000	268267520
dmz02	vcn052	--b---	0	0.00000	268222464
dmz01	Domain-0	-----r	10115	0.70000	268492800
dmz01	vcn055	-----r	147	100.00000	268177408
dmz01	vcn056	--b---	0	0.00000	268185600
dmz01	vcn057	--b---	0	0.00000	268288000



Node Name	Domain Name	Domain State	CPU	CPU %	Memory
dmz03	Domain-0	-----r	47336	0.80000	268513280
dmz03	vcn053	-----r	96	100.00000	268288000
dmz03	vcn054	--b---	0	0.00000	268185600
dmz02	Domain-0	-----r	12043	0.70000	268607488
dmz02	vcl050	-----r	108	100.00000	268296192
dmz02	vcn051	--b---	0	0.00000	268173312
dmz02	vcn052	--b---	0	0.00000	268263424
dmz01	Domain-0	-----r	10115	0.70000	268673024
dmz01	vcn055	-----r	87	100.00000	268251136
dmz01	vcn056	--b---	0	0.00000	268255232
dmz01	vcn057	--b---	0	0.00000	268226560

Fig. 1. XCM client in the Detailed view and also depicting administrative actions menu

5.1 Live Migration

Many times an administrator might want to remove a particular domain from a physical node and transfer it to run on some other physical node based on a certain performance metric. For example, a node may be hosting too many CPU intensive domains. This might reduce the throughput of a certain critical user-level application running on that domain. At the same time, there might be some node that might be only hosting a less CPU-centric application (e.g., an MPI application doing ping-pong communication). The XCM client gives the administrator a migration functionality in which they can select a source node, a source domain, and a new destination node for a domain. The live migration occurs transparently to the user-level application. Also, due to para-virtualization and hardware state saving of a virtual machine the overhead due to live migration is about 60ms.

5.2 Automatic Load Balancing

Sometimes, manual migration of virtual machines to across multiple nodes is a cumbersome task if the virtual cluster consists of a large number of physical nodes or virtual machines. Also, due to the arbitrary scheduling of virtual machines to host user-level applications, there might be inefficient usage of hardware resources. To ensure optimal hardware resource utilization, the administrator might develop some policies and algorithms which perform the load balancing of the virtual cluster by redistributing the cluster workload across multiple physical nodes. This typically requires the live migration of hundreds of domains across different physical nodes in the virtual cluster.

The XCM client enables the system administrator to perform automatic dynamic load balancing. Currently, the client uses a very simple resource management policy to define workload imbalance on a certain physical node. The initial load balancing policy seeks to balance the number of domains across the available physical nodes. We are currently extending our XCM framework to allow users to input specific policies, composed of the information provided by XCM. In this way, the administrator will be able to configure these policies based on weights given to various performance metrics collected by the XCM daemons. The current load balancing policy is based on the total number of domains in the cluster and the number of nodes in the cluster. Now, consider these three administrator defined metrics: Let N be the total number of physical nodes and D be the total number of domains in the virtual cluster. Also, let X be the number of maximum number of domains per physical node. Now, according to this policy, X will be evaluated by the expression: $X = (D / N) + 1$.

This policy limits the number of domains that can be hosted by a single physical node. We reiterate that this simple policy has been adopted to demonstrate this capability of our framework and may not be employed by real world installations of our framework. Now, whenever the system administrator selects the automatic load balancing menu item from the client's Actions menu, the XCM client parses through its internal data structures and makes a list of *migration information* (a tuple having *{source node, target domain, destination node}* information). Any node which contains more than D domains is selected as a source node and the nodes which have less than D domains are selected as the destination node. The target domains are randomly chosen from amongst the domains hosted on the source nodes. The XCM client maintains a migration information dispatch queue which stores all the pending live migration requests which have been requested by the system administrator. The "*Domain Migration Thread*" as described in section 5.1 clears this dispatch queue and sends the migration requests to the XCM daemons running on the source nodes. Hence, this dynamic load balancing strategy is application-agnostic and helps system administrator to devise load balancing policies in a virtual cluster.

5.3 Node Maintenance

Frequently, system administrators must service a physical node to upgrade and/or replace its hardware or software. Also, sometimes by monitoring metrics like heat generated per physical node, an administrator can make wise policies that can predict node failure in advance. Both these scenarios would require the administrator to remove a physical node from the virtual cluster. To make a clean removal of physical nodes from virtual clusters, the virtual machines hosted by the target physical node must be moved to a safe candidate node. Live migration can be used to evacuate a physical node of its VMs. The XCM client provides this feature to the administrator. The administrator chooses a particular physical node to be removed from the virtual cluster. The XCM client then rearranges the workload on a target node to be migrated to a safe destination node by filling out entries in the migration dispatch queue. Currently, a very simple algorithm is used to design this policy. The XCM client chooses the physical node which is hosting the minimum number of virtual machines as the destination node for the target node's domains. After these domains are migrated from the target node, the target node is proclaimed dead by the XCM client. This information is transmitted to the XCM daemon which causes it to terminate. The node can be safely switched off from the virtual cluster at that moment. Eventually, the dead node collector thread in the XCM client removes all the data structures related to the target node from its internal cluster-wide data structures.

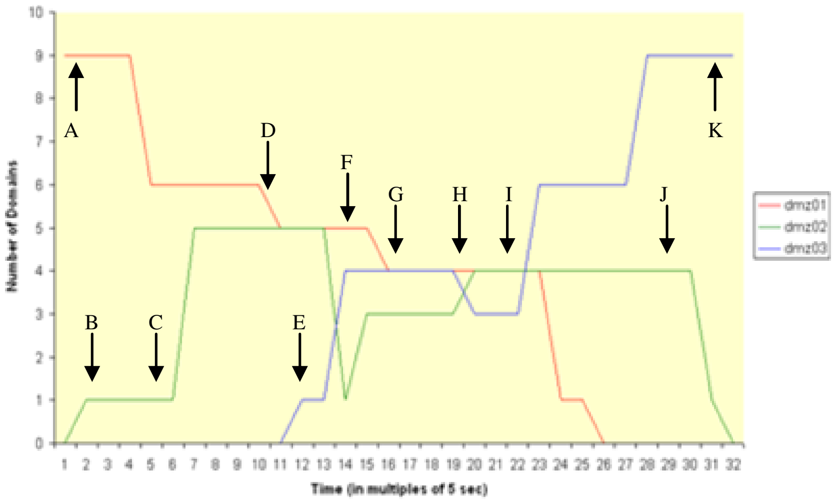


Fig. 2. Activity timeline for Xen migration experimental results

6 XCM Experiment

Using the XCM framework, we conducted an experiment comprising a variety of system administrative tasks on a virtual cluster configured to run 8 virtual machines across 3 physical nodes. Each physical node is a dual core AMD opteron machine running Red Hat Linux.

Each physical node hosts a Xen 3.0.3 VMM. On each of the 3 physical nodes, a patched fedora core 5 image was used to run as the host domain. The guest domains were also patched fedora core 5 images. In our experiments, the images were shared over NFS fileserver accessible to both the nodes. The nodes were connected through a gigabit Ethernet. The virtual machines running on these physical nodes are configured to run Linux fedora core 5 images.

Figure 2 represents a graph showing the virtual cluster workload distribution as a function of the number of domains on each of the three physical nodes as a function of time. We start of out experiment by starting a physical node and launching 8 DOM Us running on that virtual machine represented by "dmz1". At point A in the figure, there are total 9 domains running on dmz01. At time instance represented by pint B, a new physical node "dmz02" is added to the cluster. This physical node has only one domain (DOM 0) running on it. At point C, the administrator decides to balance the workload of the virtual cluster which causes the migration of 4 out of the 8 DOM Us to the newly available dmz02. The load balancing finishes at point D in time. Then at some point in time E, a new physical node, dmz03, is added to the virtual cluster. At point F, the administrator again decides to balance the cluster workload due to the availability of a free physical node. At point G in time, the virtual cluster is fully load balanced with each physical node hosting virtual machines. According to our load balancing policy discussed in section 5.2, the resources of the virtual cluster are being optimally utilized at this point in time. At time H, one domain is migrated from node dmz03 to the node dmz02 based on the cluster-wide performance metrics displayed by the XCM client. At point I in time, the administrator decides to schedule node dmz01 for system maintenance. All the domains of dmz01 are migrated to dmz03 depending on our policy as described in section 5.3. At point J in time, the node dmz03 is hosting 5 DOM Us. Then at point J in time, the administrator decides to remove node dmz02 as he decides to change the RAM configuration in that node. At point K in time, the virtual cluster only consists of one physical node, dmz03, which is now hosting all 8 DOM Us along with its DOM 0.

7 Conclusion

By providing a user-friendly framework built on top of TCP/IP sockets and the Xen hypercall interface for virtual cluster management we relieve the system administrator of a virtual cluster facility of manual interaction with individual virtual machines for performing administrative tasks like cluster performance monitoring, cluster load balancing, node maintenance, node failure etc. We also provide the administrator with a framework which can be used to automatically perform administrative actions by configurable policies designed using a variety of algorithms. This would reduce the burden of handling and maintaining a virtual cluster and would enable rapid decision making for optimal usage of hardware resources in such an environment.

References

[1] Bar, M.: Xen, the virtual machine monitor. Free Software Magazine, issue (June 5, 2005), `http://www.free/software/magazine/articles/focus-xen`

[2] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles SOSP 2003, Bolton Landing, NY, USA, October 19 - 22, pp. 164–177. ACM Press, New York (2003), `http://doi.acm.org/10.1145/945445.945462`

[3] `http://fabrice.bellard.free.fr/qemu/`

[4] Devine, S., Bugnion, E., Rosenblum, M.: Virtualization system including a virtual machine monitor for a computer with a segmented architecture (VMWARE). US Patent Office, Ed., USA (1998)

[5] `http://user-mode-linux.sourceforge.net/`

[6] Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live Migration of Virtual Machines. In: Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI), Boston, MA (May 2005)

[7] Werner Fischer and Christoph Mitasch. High availability clustering of virtual machines – possibilities and pitfalls. Paper for the talk at the 12th Linuxtag, May 3rd-6th, Wiesbaden/Germany Version 1.01 (2006)

[8] Youseff, L., Wolski, R., Gorda, B., Krintz, C.: Paravirtualization for HPC systems. UCSB Computer Science Technical Report Number (2006)-10

[9] http://virt-manager.et.redhat.com/

[10] http://libvirt.org/