Security in Virtual Machine Live Migration for KVM

Ms. YamunaDevi. L, yamunaloganathan@rediffmail.com
Ms. Aruna . P, aruna_sivan@rediffmail.com
Ms. Sudha Devi. D, dsudhadevi@yahoo.com
Ms. Priya. N, npriya_99@yahoo.com
Dept Of Computer Technology & Applications, Coimbatore Institute Of Technology,
Coimbatore, Tamil Nadu, India.

***Abstract*---Virtualization is a technical innovation designed to increase the level of system abstraction and enable IT users to harness ever-increasing levels of computer performance. The most fundamental part of virtualization is the hypervisor.**

To perform maintenance on a host server, we need to be able to move all of its VMs onto other host servers with minimum disruption to end users. Even more importantly, if our virtual workloads peak and need more resources, our virtual infrastructure should be able to automatically locate a host server with adequate resources and move the VMs to this host. And the entire process should be completely transparent to end users.

VM migration, especially live migration is a desirable feature in Grid and Cloud Computing systems for load balancing, elastic scaling, fault tolerance, and hardware maintenance. The security and reliability issues that reside in VM live migration is a critical factor for its acceptance by IT industry.

Though the performance is optimized through live migration, there are two important issues to be noted, reliability and security. In this paper, we propose how to migrate a VM Guest to another VM Host Server. We discuss the steps regarding the KVM live migration requirements and the security and reliability issues in live migration.

I. INTRODUCTION

At its simplest level, virtualization allows, virtually and cost-effectively, to have two or more virtual computing environments, running different operating systems and applications on one piece of hardware. For example, with virtualization, you can have both a Linux virtual machine and a Microsoft Windows virtual machine on one system. Alternatively, you could host a Microsoft Windows 95 desktop and a Microsoft Windows XP desktop on one workstation.

The most fundamental part of virtualization is the hypervisor. The hypervisor acts as a layer between the virtualized guest operating system and the real hardware. There are two types of hypervisors. Type 1 hypervisors run directly on the system hardware. Type 2 hypervisors run on a host operating system that provides virtualization services, such as I/O device support and memory management. Types1 hypervisor are typically the preferred approach because they can achieve higher virtualization efficiency by dealing directly with the hardware. Type1 hypervisors provide higher performance efficiency, availability, and security than type 2 hypervisors. Type 2 hypervisors are used mainly on systems where support for a broad range of I/O devices is important and can be provided by the host operating system.

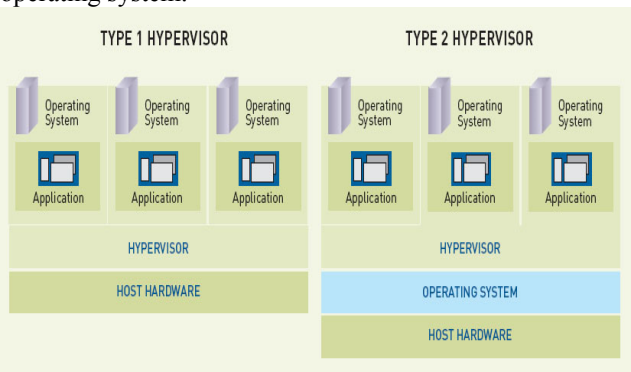

Fig. 1. Type 1 and Type 2 Hypervisor

A. ADVANTAGES OF VIRTUALIZATION

Virtualization consolidates workloads to reduce hardware, power, and space requirements. It runs multiple operating systems simultaneously and legacy software on newer, more reliable, and more power-efficient hardware. It dynamically migrates

978-1-61284-764-1/11/$26.00 ©2011 IEEE

workloads to provide fault tolerance.

The rest of the paper is organized as follows: The next section discusses about the major hypervisors available. Section III describes about the live migration of VM and its benefits. Section IV describes the steps involved in setting up a KVM host and building a virtual machine on the KVM host. Section V describes the steps in migrating a guest VM from a KVM host to another host. Section VI discusses the important issues in live migration. Section VII summarizes our conclusions and outlines our future directions.

II. MAJOR HYPERVISORS

There are three areas of IT where virtualization is making head roads, network virtualization, storage virtualization and server virtualization.

All three types of virtualization require hypervisor software to make them work. The difference is in the details and the load that the hypervisor is expected to carry in managing each virtual machine.

Xen is the oldest open source virtualization technology and has been available for approximately five years. The most important advantage of Xen is Para virtualization, a technology that enables virtual machines to run efficiently and without requiring emulation (thus guests are aware of the hypervisor and can run efficiently without emulation or virtual emulated hardware). The most important disadvantage of Xen is that it is a complex product that can be difficult to integrate with the Linux kernel.

KVM (Kernel based Virtual Machine) provide a virtualization technology that resides in the Linux kernel itself. KVM is just a single module that has to be loaded in the Linux kernel. Once the module is loaded, virtual machines ca be created. KVM is a relative newcomer, but the strength and simplicity of its implementation, plus continued support of Linux heavyweights, make it worthy of serious consideration. KVM strengthens 1.Secuirty 2. Memory Management 3. Live Migration 4. Performance and Scalability 5. Guest Support. By adding virtualization capabilities to a standard Linux kernel, the virtualized environment can benefit from all the ongoing work on the Linux kernel itself. Under this model, every virtual machine is a regular Linux process, scheduled by the standard Linux scheduler.

KVM support is pre-built into the Fedora Linux kernel for Fedora release 7 and later. As a result KVM support is already available in the standard kernel negating the need to install and boot from a special kernel. To fully utilize the KVM support built into the kernel the following packages are required:

- qemu-kvm
- virt-manager
- virt-viewer
- python-virtinst

III. LIVE MIGRATION

Cloud computing is a popular trend in current computing which attempts to provide cheap and easy access to computational resources. Infrastructure as a Service (IaaS) cloud computing focuses on providing a computing infrastructure that leverages system virtualization to allow multiple Virtual Machines (VM) to be consolidated on one Physical Machine (PM) where VMs often represent components of Application Environments(AE).

VMs are loosely coupled with the PM they are running on; as a result, not only can a VM be started on any PM, but also, it can be migrated to other PMs in the data center. Migrations can either be accomplished by temporarily suspending the VM and transferring it, or by means of a live migration in which the VM is only stopped for a split second. With the current technologies, migrations can be performed on the order of seconds to minutes depending on the size and activity of the VM to be migrated and the network bandwidth between the two. The ability to migrate VMs makes it possible to dynamically adjust data center utilization and tune the resources allocated to AEs. Furthermore, these adjustments can be automated through formally defined strategies in order to continuously manage the resources in a data center with less human intervention.

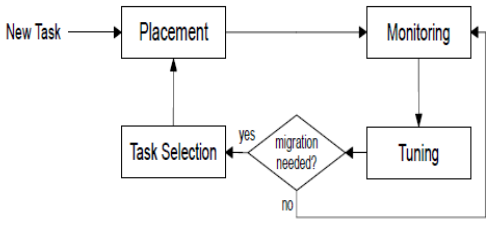

Fig. 2. Process of dynamic Allocation of Jobs to VMs

A. ADVANTAGES

- Load balancing - guests can be moved to hosts with lower usage when a host becomes overloaded.
- Hardware failover - when hardware devices on the host start to fail, guests can be safely relocated so the host can be powered down and repaired.
- Energy saving - guests can be redistributed to other hosts and host systems powered off to save energy and cut costs in low usage periods.
- Geographic migration - guests can be moved to another location for lower latency or in serious circumstances.

IV. SETTING UP THE KVM HOST AND BULIDING A VM

Virtual systems can easily be configured using either the virt-install command-line tool, or the virt-manager GUI tool. We can launch virt-manager either by selecting the Applications->System Tools->Virtual Machine Manager or from the command-line in a terminal window by running virt-manager.

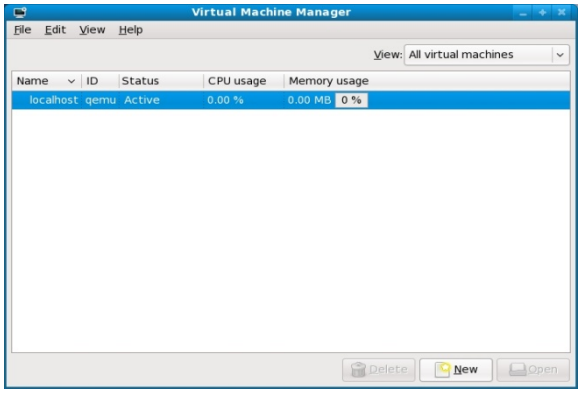

Fig. 3. Launching virt-manager

The main virt-manager screen lists the current virtual machines running on the system. To create a new virtual system, click on the new button to display the first configuration screen. Click the Forward button to proceed to the Naming screen and enter a name for the virtual system which can be any name you choose. On this screen, also select the location of the media from which the guest operating system will be installed. This can either be a CD or DVD drive, an ISO image file accessible to the local host or a network install using HTTP, FTP, NFS or PXE.

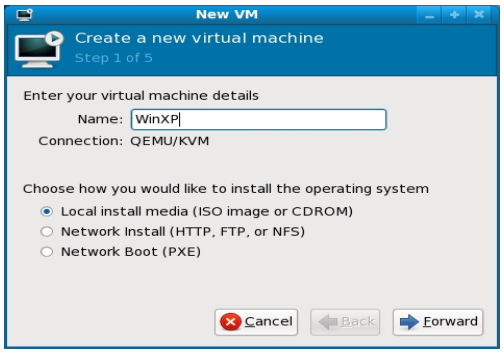

Fig. 4. Creation of new virtual machine

Clicking Forward once more will display a screen seeking additional information about the installation process. The screen displayed and information required will depend on selection made in the preceding screen. For example, if a CD, DVD or ISO was selected, this screen will ask for the specific location of the ISO file or physical device. This screen also asks to specify the type and version of the guest operating system to be installed (for example Windows Vista or Ubuntu Linux).

A. CONFIGURING FEDORA SYSTEM

Once these settings are complete, click forward to configure CPU and memory settings. The optimal settings will depend on the number of CPUs and amount of physical memory present in the host and the requirements of other applications and virtual machines that will run in parallel with the new virtual machine.

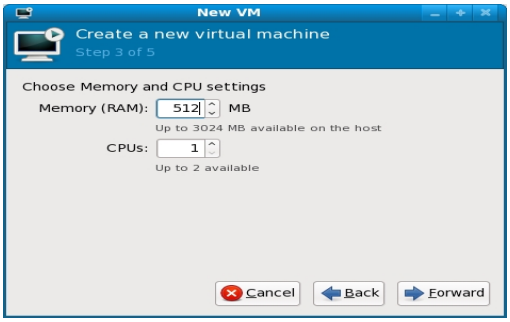

Fig. 5. Choose memory and CPU settings

The last item to configure before creating the virtual machine is the storage space for the guest operating system and corresponding user data. This

takes the form of a virtual disk drive. A virtual disk drive is essentially an image file hosted on the file system of the host computer which is seen by the virtual machine as a physical disk drive.

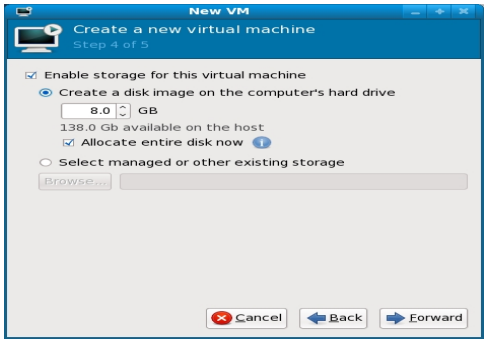

Fig. 6. Enable storage for VM

Once these settings are configured, click the Forward key once more. The final screen displays a summary of the configuration. Review the information displayed. Advanced options are also available to configure bridged networking (where the virtual machine has direct access to a host network adapter) and to configure a specific MAC address if required.

Fig. 7. Summary of Configuration

B. STARTING THE VIRTUAL MACHINE

Click on the Finish button to begin the creation process. The virtualization manager will create the disk and configure KVM before starting the system. Once started, the guest OS installation will begin.

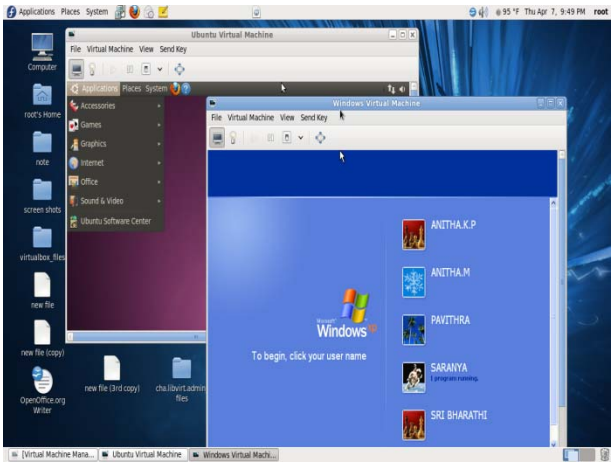

Fig. 8. Running Windows and Ubuntu on Fedora

V. LIVE MIGRATION IN KVM HOST

A. LIVE MIGRATION REQUIREMENTS:

Migrating guests requires the following:

1. A virtualized guest installed on shared networked storage using one of the following protocols
   - Fiber Channel
   - iSCSI
   - NFS
   - GFS2
2. Two or more Fedora systems of the same version with the same updates.
3. Both systems must have the appropriate ports open.
4. Both systems must have identical network configurations. All bridging and network configurations must be exactly the same on both hosts.
5. Shared storage must mount at the same location on source and destination systems. The mounted directory name must be identical.

Migrations can be performed live or offline. To migrate guests the storage must be shared. Migration works by sending the guests memory to the destination host. The shared storage stores the guest's default file system. The file system image is not sent over the network from the source host to the destination host.

An offline migration suspends the guest then moves an image of the guests memory to the destination host. The guest is resumed on the destination host and the memory the guest used on the source host is freed. The time an offline migration takes depends network bandwidth and latency. A

guest with 2GB of memory should take an average of ten or so seconds on a 1 Gbit Ethernet link.

A live migration keeps the guest running the source host and begins moving the memory without stopping the guest. All modified memory pages are monitored for changes and sent to the destination while the image is sent. The memory is updated with the changed pages. The process continues until the amount of pause time allowed for the guest equals the predicted time for the final few pages to be transfer. KVM estimates the time remaining and attempts to transfer the maximum amount of page files from the source to the destination until KVM predicts the amount of remaining pages can be transferred during a very brief time while the virtualized guest is paused. The registers are loaded on the new host and the guest is then resumed on the destination host. If the guest cannot be merged (which happens when guests are under extreme loads) the guest is paused and then an offline migration is started instead.

The time an offline migration takes depends network bandwidth and latency. If the network is in heavy use or a low bandwidth the migration will take much longer.

B. LIVE MIGRATION WITH VIRSH

A guest can be migrated to another host with the virsh command. The migrate command accepts parameters in the following format

 # virsh migrate --live Guest Name Destination Name

The Guest Name represents the name of the guest which you want to migrate.

The Destination Name is the hostname of the destination system. The destination system must run the same version of Fedora, be using the same hypervisor and have libvirt running.

1. Verify the guest is running

```
[root@test1 ~]# virsh list
Id Name                 State
----------------------------------
 10 RHEL4                running
```

2. Migrate the guest

Execute the following command to live migrate the guest to the destination, test2.example.com. Append /system to the end of the destination URL to tell libvirt.

virsh migrate --live *RHEL4test qemu+ssh://test2.example.com/system*

3.Wait

The migration may take some time depending on load and the size of the guest. virsh only reports errors. The guest continues to run on the source host until fully migrated.

4. Verify the guest has arrived at the destination host.

From the destination system, test2.example.com, verify RHEL4test is running:

```
[root@test2 ~]# Virsh list
Id Name                 State
----------------------------------
 10 RHEL4                running
```

The live migration is now complete.

C. LIVE MIGRATION WITH VIRT-MANAGER

1. Connect to the source and target hosts.
2. Add a storage pool with the same NFS to the source and target hosts.
3. Create a new volume in the shared storage pool
4. Create a virtual machine with the new volume, and then run the virtual machine
5. The Virtual Machine window appears.
6. In the Virtual Machine Manager window, right-click on the virtual machine, select Migrate, then click the migration location.
7 .The Virtual Machine windows display the new virtual machine location.

VI. ISSUES IN LIVE MIGRATION

Live migration is a useful feature and natural extension to virtualization technology that allows for the transfer of a virtual machine from one physical machine to another with little or no downtime for the services hosted by the virtual machines. The migration functionality implemented by vendors such as XEN, and KVM now exposes the entire machine state of a VM to device module which listens to the incoming live migration requests from remote platforms. An attack can easily hijack the device module process or hypervisor where these migrations occur. If the process is hijacked, the information of the migrated virtual machine including states of operation system kernel, applications and services running within the operating system, the sensitive data currently being used by those applications and even the inputs from keyboard are accessible to the hackers.

Another problem resides in migration is the reliability. There will be a problem when the connection between two devices is shutdown unexpectedly. This may occur when people have an emergent requirement. Without doing a regular migration process, the target machine will not have the same status as the original platform. Currently this can lead the Guest OS to restart in target to rebuild all the environments which make it unavailable for some time and the loss the work context eventually.

A. VMM-ENFORCED PROCESS PROTECTION

Research efforts are made to build the trustworthy execution environment in the face of a malicious platform owner or a compromised operating system. VMM-based security systems are advocated that provide isolated execution environments for Applications [9, 10, 11]. However, though the VMs are isolated from each other, the operating systems running inside the VM is not isolated from the applications, and thus should be trusted. As the commodity operating systems are too complex and vulnerable to be trusted, a practical way is to reduce the size of the operating system so as to increase its trustworthiness. However, this approach limits the functionality of the operating system. The tradeoff of functionality and trustworthy hinderes the prevalence of VMM-based Trusted Computing systems.

Recent research [5, 6, 7, 8] suggests that it is possible to provide a trustworthy execution environment for applications without trusting the operating system. In these systems, the applications are protected directly by the VMM. Even in the face of a totally compromised operating system, the privacy and integrity of the protected binary/data files, memory contents, and CPU contexts are still protected.

VII. CONCLUSION

Successful migration on KVM has implemented. By Integrating live migration on KVM work is shared among computers in order to optimize computer resources**.**

In this paper, we have discussed the related issues in live migration of VM and proposed the VMM- enforced process protection technique for preserving secure live migration. In near future, we are going to alleviate the security issue in live migration through the VMM-enforced process protection.

VIII. REFERENCES

[1] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hanseny,Eric July, Christian Limpach, Ian Pratt, Andrew War_eld "Live migration of Virtual Machines"

[2] Jon Oberheide,Evan Cooke,Farnam Jahanian "Empirical Exploitation of Live Virtual Machine Migration"

[3] Uri Lublin, Qumranet,Anthony Liguori "Kvm Live Migration"

[4] Alan Murphy "Virtualization define-Eight Steps"

[5] H. Chen, J. Chen, W. Mao, and F. Yan. Daonity–grid security from two levels of virtualization. *Information Security Technical Report*, 12(3):123–138, 2007.

[6] H. Chen, F. Zhang, C. Chen, R. Chen, B. Zang, P. Yew, and W. Mao. Tamper-Resistant Execution in an Untrusted Operating System Using A Virtual Machine Monitor. Technical Report 2007-08001, Parallel Processing Institute, Fudan University, Aug. 2007.

[7] X. Chen, T. Garfinkel, E. Lewis, P. Subrahmanyam, C. Waldspurger, D. Boneh, J. Dwoskin, and D. Ports. Overshadow: a virtualization-based approach to retrofitting protection in commodity operating systems. *Proceedings of the 13th international conference on Architectural support for programming languages and operating systems*, pages 2–13, 2008.

[8] J. Yang and K. Shin. Using hypervisor to provide data secrecy for user applications on a per-page basis. P*roceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 71–80, 2008.

[9] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: a virtual machine-based platform for trusted computing. In *Proc. of SOSP'03*, pages 193–206, 2003.

[10] D. Kuhlmann, R. Landfermann, H. Ramasamy, M. Schunter, G. Ramunno, and D. Vernizzi. An open trusted computing architecture - secure virtual machines enabling user-defined policy enforcement. Technical Report RZ3655, IBM Research, 2006.

[11] D. Murray, G. Milos, and S. Hand. Improving Xen security through disaggregation. *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 151–160, 2008.