Live Migration of Virtual Machines in Cloud

Ashima Agarwal, Shangruff Raina

Department of Computer, MIT College of Engineering Pune, India

Abstract- Migration of a virtual machine is simply moving the VM running on a physical machine (source node) to another physical machine (target node). It is done as, while the VM is running on the source node, and without disrupting any active network connections, even after the VM is moved to the target node. It is considered "live", since the original VM is running, while the migration is in progress. Huge benefit of doing the live migration is the very small (almost zero) downtime in the order of milliseconds.

There exists a model in which the load is balanced among the servers according to their processor usage or their IO usage and keeping virtual machines zero downtime. To increase the throughput of the system, it is necessary that the virtual machines are load balanced statically, i.e. the load is distributed to each part of the system in proportion to their computing IO capacity.

To migrate a virtual machine from one physical host to another, the control of virtual machines is converted to the management of services in Red Hat Cluster Suite. This creates a high availability and load balancing cluster services, to migrate a virtual machine from one physical host to another.

Software services, file systems and network status can be monitored and controlled by the cluster suite, services and resources can be failed over to other network nodes in case of failure. The cluster suite forcibly terminates a cluster node's access to services or resources to ensure the node and data is in a known state. The node is terminated by removing power or access to the shared storage.

I. INTRODUCTION

What is virtual machine?

One computer containing multiple operating systems loaded on a single PC, each of which functions as a separate OS on a separate physical machine. Virtualization [1] software does just that by creating and managing one or more virtual machines on a single, physical host PC. It can run its own operating systems and applications as if it were a physical computer. A virtual machine [2] behaves exactly like a physical computer and contains it own virtual (i.e. software-based) CPU, RAM hard disk and network interface card (NIC).

Benefits of virtual machines

1. **Isolation**: While virtual machines can share the physical resources of a single computer, they remain completely isolated from each other as if they were separate physical machines. If, for example, there are four virtual machines on a single physical server and one of the virtual machines crashes, the other three virtual machines remain available. Isolation is an important reason why the availability and security of applications running in a virtual environment is far
2. **Encapsulation:** A virtual machine is essentially a software container that bundles or "encapsulates" a complete set of virtual hardware resources, as well as an operating system and all its applications, inside a software package. Encapsulation makes virtual machines incredibly portable and easy to manage.
3. **Hardware Independence**: Virtual machines are completely independent from their underlying physical hardware. For example, you can configure a virtual machine with virtual components (eg, CPU, network card, SCSI controller) that are completely different from the physical components that are present on the underlying hardware. Virtual machines on the same physical server can even run different kinds of operating systems (Windows, Linux, etc).

II. WHAT IS LIVE MIGRATION?

Live migration [3] is the movement of a virtual machine from one physical host to another while continuously powered-up. When properly carried out, this process takes place without any noticeable effect from the point of view of the end user. Live migration allows an administrator to take a virtual machine offline for maintenance or upgrading without subjecting the system's users to downtime.

When a VM is running a live service it is important that this transfer occurs in a manner that balances the requirements of minimizing both downtime and total migration *time*. The former is the period during which the service is unavailable due to there being no currently executing instance of the VM; this period will be directly visible to clients of the VM as service interruption. The latter is the duration between when migration is initiated and when the original VM may be finally discarded and hence, the source host may potentially be taken down for maintenance, upgrade or repair. It is easiest to consider the trade-offs between these requirements by generalizing memory transfer into three phases:

Push phase: The source VM continues running while certain pages are pushed across the network to the new destination. To ensure consistency, pages modified during this process must be re-sent.

Stop-and-copy phase: The source VM is stopped, pages are copied across to the destination VM, then the new VM is started. The time between stopping VM on source and resuming it on destination is called "down-time". Down-time of a VM during a live migration could be a few milliseconds to seconds according to the size of memory and applications running on the VM. There are some techniques to reduce live migration down-time such as using probability density function of memory change.[3]

Pull phase: The new VM executes and, if it accesses a page that has not yet been copied, this page is faulted in (.pulled.) across the network from the source VM. VM migration is initiated by suspending the VM at the source. With the VM suspended, a minimal execution state of the VM (CPU, registers, and non-pageable memory) is transferred to the target. The VM is then resumed at the target, even though the entire memory state of the VM has not yet been transferred, and still resides at the source. At the target, when the VM tries to access pages that have not yet been transferred, it generates page-faults, which are trapped at the target and redirected towards the source over the network. Such faults are referred to as network faults. Source host responds to the network-fault by sending the faulted page. Since each page fault of the running VM is redirected towards the source, it can degrade the applications running inside the VM. However, when pure demand-paging accompanied with the techniques such as pre-paging can reduce this impact by a great extent. Live migration is widely used in virtualization ready data centers and enterprise IT departments. It separates software's from physical servers, and provides desirable abilities such as online server maintenance, dynamic load balancing, and etc.

Live migration can be applied in both local area network (LAN) environment and wide area network (WAN) environment. Live migration in LAN environment is simpler, because live migration process avoids virtual storage migration by sharing a network storage. Furthermore, live migration in LAN skips the migration of network topology, and the migration process only needs to broadcast an unsolicited ARP reply from the migrated VM, in order to advertise the MAC relocation.

III. Seamless live migration

When down-time of a VM during a live migration is zero or a few millisecond which is not noticeable by end user, it is called a seamless live migration. Otherwise, the end user will feel a small or relatively long glitch in the service.

What is the purpose of live migration?

Migrating an entire OS and all of its applications as one unit allows us to avoid many of the difficulties faced by process-level migration approaches. With virtual machine migration, on the other hand, the original host may be decommissioned once migration has completed. This is particularly valuable when migration is occurring in order to allow maintenance of the original host.

Secondly, migrating at the level of an entire virtual machine means that in-memory state can be transferred in a consistent and (as will be shown) efficient fashion. This applies to kernel-internal state (e.g. the TCP control block for a currently active connection) as well as application-level state, even when this is shared between multiple cooperating processes.

Thirdly, live migration of virtual machines allows a separation of concerns between the users and operator of a data center or cluster. Users have `carte blanche' regarding the software and services they run within their virtual machine, and need not provide the operator with any OS-level access at all. Live OS migration is a extremely powerful tool for cluster administrators, allowing separation of hardware and software considerations, and consolidating clustered hardware into a single coherent management domain. If a physical machine needs to be removed from service an administrator may migrate OS instances including the applications that they are running to alternative machine(s), freeing the original machine for maintenance.

How is live migration achieved?

By using a *pre-copy* approach in which pages of memory are iteratively copied from the source machine to the destination host, all without ever stopping the execution of the virtual machine being migrated. Page level protection hardware is used to ensure a consistent snapshot is transferred, and a rate-adaptive algorithm is used to control the impact of migration traffic on running services. The final phase pauses the virtual machine, copies any remaining pages to the destination, and resumes execution there. We eschew a `pull' approach which faults in missing pages across the network since this adds a residual dependency of arbitrarily long duration, as well as providing in general rather poor performance.

The logical steps that we execute when migrating an OS are summarized in Figure 3. We take a conservative approach to the management of migration with regard to safety and failure handling. Although the consequences of hardware failures can be severe, our basic principle is that safe migration should at no time leave a virtual OS more exposed to system failure than when it is running on the original single host. To achieve this, we view the migration process as a transactional interaction between the two hosts involved:

Stage 0 Pre-Migration: We begin with an active VM on physical host A. To speed any future migration, a target host may be preselected where the resources required to receive migration will be guaranteed.

Stage 1 Reservation: A request is issued to migrate an OS from host A to host B. We initially confirm that the necessary resources are available on B and reserve a VM container of that size. Failure to secure resources here means that the VM simply continues to run on A unaffected.

Stage 2 Iterative Pre-Copy: During the first iteration, all pages are transferred from A to B. Subsequent iterations copy only those pages dirtied during the previous transfer phase.

Stage 3 Stop-and-Copy: We suspend the running OS instance at A and redirect its network traffic A to B. As described earlier, CPU state and any remaining inconsistent memory pages are then transferred. At the end of this stage there is a consistent suspended copy of the VM at both A and B. The copy at A is still considered to be primary and is resumed in case of failure.

Stage 4 Commitment Host B indicates to A that it has successfully received a consistent OS image. Host A acknowledges this message as commitment of the migration transaction: host A may now discard the original VM, and host B becomes the primary host.

Stage 5 Activation: The migrated VM on B is now activated. Post-migration code runs to reattach device drivers to the new machine and advertise moved IP addresses.

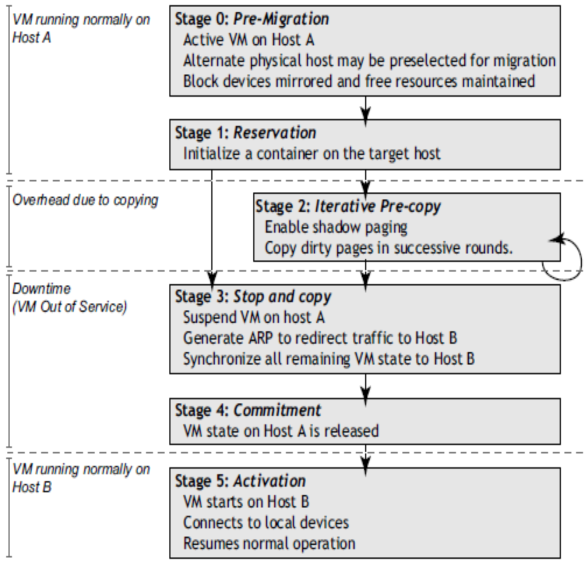
VM running normally on Host A
Stage 0: Pre-Migration
Active VM on Host A
Alternate physical host may be preselected for migration
Block devices mirrored and free resources maintained
Stage 1: Reservation
Initialize a container on the target host
Overhead due to copying
Stage 2: Iterative Pre-copy
Enable shadow paging
Copy dirty pages in successive rounds.
Downtime (VM Out of Service)
Stage 3: Stop and copy
Suspend VM on host A
Generate ARP to redirect traffic to Host B
Synchronize all remaining VM state to Host B
Stage 4: Commitment
VM state on Host A is released
VM running normally on Host B
Stage 5: Activation
VM starts on Host B
Connects to local devices
Resumes normal operation

Figure 3: Migration Timeline

This approach to failure management ensures that at least one host has a consistent VM image at all times during migration. It depends on the assumption that the original host remains stable until the migration commits, and that the VM may be suspended and resumed on that host with no risk of failure. Based on these assumptions, a migration request essentially attempts to move the VM to a new host, and on any sort of failure execution is resumed locally, aborting the migration.

To solve the problem of load balancing amongst the physical hosts in cloud by adaptive live migration of virtual machine, there exists a model in which the load is balanced among the servers according to their processor usage or their IO usage and keeping virtual machines zero downtime. To increase the throughput of the system, it is necessary that the virtual machines are load balanced statically, i.e. the load is distributed to each part of the system in proportion to their computing IO capacity. The System Architecture of such a model is given in next chapter.

IV. System Architecture

By the use of EUCALYPTUS [4] one or more VLAN can be created, each VLAN may be across many physical hosts and may include many virtual machines, so when a virtual machine is migrated from one physical host to another, an inclusion relationship can be kept between VLAN and virtual machine.
Following diagram represents the system architecture of an adaptive intra-cloud load balancing model.

Lblog: The program lblog is a small cronjob which runs regularly on each host to monitor predefined resources, such as CPU and IO load, and log their usage to the log directory on the shared storage.

DLBA (Distributed load balancing algorithm): It runs on each physical host separately and dynamically migrates virtual machines of local host to other physical hosts according to resource uses.

Dump: When virtual machines are migrated, their memory contents are dumped to the dump directory of shared storage.

Conf: This directory contains the configuration files.

Private: This directory contains the file system.

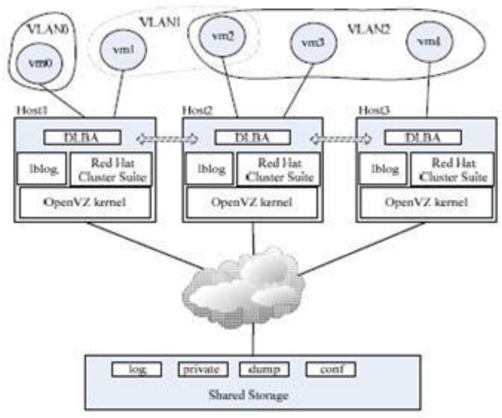

Figure 4: System Architecture

To migrate a virtual machine from one physical host to another, the control of virtual machines is converted to the management of services in **Red Hat Cluster Suite**[5]. This creates a high availability and load balancing cluster services, to migrate a virtual machine from one physical host to another. The cluster suite forcibly terminates a cluster node's access to services or resources to ensure the node and data is in a known state. The node is terminated by removing power or access to the shared storage. The detailed explanation of the working of Red Hat Cluster Suite is given in next chapter.

V. Red Hat Cluster Suite

Introduction

High-availability Service Management

High-availability[7] service management provides the ability to create and manage high-availability *cluster services* in a Red Hat cluster. The key component for high-availability service management in a Red Hat cluster, rgmanager, implements cold failover for off-the-shelf applications. In a Red Hat cluster, an application is configured with other cluster resources to form a high-availability cluster service. A high-availability cluster service can fail over from one cluster node to another with no apparent interruption to cluster clients. Cluster-service failover can occur if a cluster node fails or if a cluster system administrator moves the service from one cluster node to another (for example, for a planned outage of a cluster node). A cluster service can run on only one cluster node at a time to maintain data integrity. Failover priority can be specified in a failover domain. Specifying failover priority consists of assigning a priority level to each node in failover domain. The priority level determines the failover order — determining which node that a cluster service should fail over to. If you do not specify failover priority, a cluster service can fail over to any node in its failover domain. Also, if a cluster service is restricted to run only on nodes of its associated failover domain. (When associated with an unrestricted failover domain, a cluster service can start on any cluster node in the event no member of the failover domain is available.)

VDE (Virtual Distributed Ethernet)

Introduction

The acronym VDE [6] is self explaining: it is a *Virtual* network because it is built completely in software, it is *Distributed* as parts of the same network can run on different physical (real) computers and it is an *Ethernet* as the entire virtual software structure is able to forward, dispatch and route plain Ethernet packets. The main features of VDE are the following:

- VDE is Ethernet compliant.
- VDE is general; it is a virtual infrastructure that gives connectivity to several kinds of software components: emulators/virtual machines, real operating systems and other connectivity tools.
- VDE is distributed.
- VDE does not need specifically administration privileges to run.

With VDE it is also possible to integrate real computers in the emulated network. When a real computer is connected to a VDE a virtual interface (based on tuntap) is visible from the operating system. This virtual interface appears exactly as it were a hardware interface and behaves as a physical ethernet interface. This operation however changes the network behavior of the host computer and thus need administrative privileges to be completed.

Currently VDE supports User-Mode Linux virtual machines, qemu, Bochs and MPS.

Network Connection for Zero-Downtime Migrated Virtual Machine

EUCALYPTUS provides support for VLAN across multi physical hosts via VDE virtual switches. There is a VDE switch on each physical host. Ports of VDE switch are classified into two categories: tagged port and untagged port. Tagged ports transmit packages belonging to multi VLAN, for example the port 3 on Host0 and untagged ports transmit packages belonging to a VLAN, for example the port 1 and 2 on Host0. On single physical host there may be multi VLAN, and each VLAN binds to an untagged port of VDE switch on the host dynamically, or example VLAN0 and VLAN1 on Host0 respectively bind to untagged port 1 and 2,. Bridge device connects to untagged port of VDE switch via virtual network card tap. VDE switches on different physical hosts connect to each other via vde plug which is part of VDE switch software suite and implements package transmission, encryption and authentication functions.

In figure 10 below, when vm2 belonging to VLAN1 migrates from Host0 to Host1, we need to delete virtual network device veth2.0 from bridge br1 on Host0. If there were no other network devices in bridge br1 except tap1 and veth2.0, we also would needed to unbind VLAN1 to untagged port 2, thus port 2 could be used by other VLAN in the future. On the target Host1, because VLAN1 has existed, before Red Hat Cluster Suite starts vm2, we only need create a pair of virtual network devices on Host1 for vm2: eth0 and veth2.0 (eth0 in virtual machine vm2 and veth2.0 on the Host1), then add veth2.0 into the bridge br0. If there were not VLAN1 on
Host1, we would need to find an untagged port in VDE switch and bind it to VLAN1, also create the corresponding bridge br0.

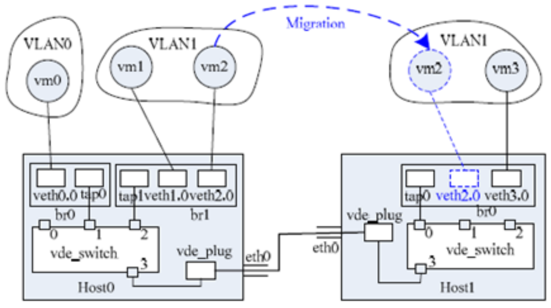

Figure 8: Virtual Machine Migration

Advantages of Live Migration

1. Reduce IT costs and improve flexibility with server consolidation.
2. Decrease downtime and improve reliability with business continuity and disaster recovery.
3. Increase energy efficiency by running fewer servers and dynamically powering down unused servers with our green IT solutions
4. Accessing more processing power (in the sense of load balancing),
5. Exploitation of resource locality (for performance), resource sharing (meaning sharing of expensive or rare.
6. Resources - such as telescopes or medical equipment – or large amounts o free memory by processes over a network), fault resilience.
7. Simplified system administration and mobile computing (for instance as used by commuters from office to home).

VI. Conclusion

Thus, Live Migration is the movement of a virtual machine from one physical host to another while continuously powered-up. This helps in decreasing downtime and improves reliability with business continuity and disaster recovery. To migrate a virtual machine from one physical host to another, the control of virtual machines is configured to the management of services in **Red Hat Cluster Suite**. VDE (virtual distributed Ethernet) virtual switches are used which are connected between the physical machines. Using VDE, users can create VLAN for each virtual machine instance. Hence, the motive of zero downtime of virtual machines is achieved by live migration.

REFERENCES

[1] http://www.vmware.com/virtualization/virtual-machine.html
[2] http://searchservervirtualization.techtarget.com/definition/virtual-machine
[3] C Clark, KFraser, S Hand, J Hansen, and E Jul., "Live migration of virtual machines", Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI), pages: 273-286, 2005
[4] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk., "The EUCALYPTUS Open-source Cloud computing System", in Proceedings of Cloud Computing and Its Applications [online], Chicago, Illinois, October 2008
[5] Yi Zhao, Wenlong Huang " Adaptive Distributed Load Balancing Algorithm based on Live Migration of Virtual Machines in Cloud"
[6] Davoli,R., "VDE: virtual distributed Ethernet", Test beds and Research Infrastructures for the Development of Networks and Communities, pages 213-220, February 2005
[7] http://www.centos.org/docs/5/html/5.1/Cluster_Suite_Overview/s1-service-management-overview-CSO.html

AUTHORS

First Author – Ashima Agarwal, Third year, computer engineering, MIT College of Engineering, Pune, Email: ashimaagarwal30@gmail.com

Second Author – Shangruff Raina, Third year,computer engineering, MIT College of Engineering, Pune, Email: shangruff@gmail.com