Live Virtual Machine Migration Based on Cow Checkpointing Using XEN Hypervisor

Blessy R & Benedict Jayaprakash Nicholas

Department of Computer Science & Engineering, Rajalakshmi Engineering College, Chennai, India.
E-mail : rh.blessy@gmail.com

Abstract – **Live Virtual Machine (VM) migration usually involves Migrating physical memory image, the network connections and virtual device state. Live Migration has been proposed to reduce the downtime of migrating the virtual machine from host to destination by encountering Pre-copy approach. If dirty memory generation is high, it takes long time for live migration to be accomplished .In extreme cases it might also lead to its failure when memory generation rate is faster than the pre-copy speed. Along with the existing approaches, the incorporation of a new standard of Copy on Write checkpointing technique is used to recover long running desktop application.**

Keywords – Live Migration, Virtualization, Restore, Virtual Machine.

I. INTRODUCTION

Virtualization abstracts the underlying physical structure of various technologies. Virtualization, in computing, is the creation of a virtual (rather than actual) version of something, such as a hardware platform, operating system, a storage device or network resources. Virtualization is a combination of software and hardware engineering that creates virtual machines, an abstraction of the computer hardware that allows a single machine to act as if it where many machines. Virtualization refers to technologies designed to provide a layer of abstraction between computer hardware systems and the software running on them. By providing a logical view of computing resources, rather than a physical view, virtualization solutions make it possible to do a couple of very useful things: They can allow you, essentially, to trick your operating systems into thinking that a group of servers is a single pool of computing resources. And they can allow you to run multiple operating systems simultaneously on a single machine. Virtualization has its roots in partitioning, which divides a single physical server into multiple logical servers. The combination of virtualization and migration significantly improves manageability of data centers and clusters in a LAN environment In addition, we are also interested in provisioning of high service availability when the whole data center becomes unavailable entirely, due to data center wide maintenances, failures, security compromises, or other catastrophic events. Moreover, there are certain scenarios that may require migrating computation engines rather than transferring data set to computation engines for policy limitations (e.g., a data set that is embargoed from export), or capacity limitations (e.g., a data set is exceedingly large, thus adding an unwieldy preamble to the computation phase). For such cases, VM migration between data centers across WANs is also necessary.

II. RELATED WORKS

The number of applications running upon the virtualized system increased, the virtual network circumstance becomes more and more complicated the consequent security problems thereby have been a concern for industrial and academic fields [2]. However, the current solutions are mostly confined to the enforcement of several patchy-works on system which still requires proficient hacking skills for administrators and cannot ensure continuous protection for VM, resulting in potential security risks. In this paper we present a framework (VNSS) which provides both guarantee of distinct security level requirement and full-lifecycle protection for VM.

The trace/replay technologies are used to provide fast, transparent VM migration for both LAN and WAN environments [1]. With execution trace logged on the source host, a synchronization algorithm is performed to orchestrate the running source and target VMs until they reach a consistent state.

The trusted virtual machine which introduces trusted computing technology into virtual machine instances. The means of partly porting the VTPM components in Xen into an administrative read-only domain; where they are protected from illegal operations in Domain 0, consequently enhancing the confidentiality of guest virtual machines [3]. This strengthens the integrity of virtual TPM itself and optimizes trusted computing base of Xen.

Server virtualization helps us system and network administrators manage computer resources and reduce power consumption. To make efficient use of virtualization, its various loads, especially I/O operations, require careful consideration [4]. The performances of I/O operations are performed on a virtual machine and power consumption of the host machine, varying the I/O schedulers on both machines.

Virtualization is being used for a variety of purposes. The ability of system-level virtual machines (VMs) to decouple the operating system from the hardware has spurred their use in the area of server consolidation to improve system utilization. This can help to reduce the number of physical machines and reduce the associated operation costs, e.g., power, cooling, etc. The encapsulation of VMs can be leveraged to assist in system management and provide user customizable environments. Virtualization also provides interesting opportunities for fault tolerance, e.g., VM migration for proactive fault tolerance [5]. These capabilities are being leveraged for research and development in a range of domains, to include High-Performance Computing (HPC). It consists of the implementation of a new hypervisor mechanism for loading dynamic shared objects (modules) at runtime. These loadable hypervisor modules (LHM) are modelled after the loadable kernel modules used in Linux. We detail the current LHM implementation based on the Xen hypervisor. Potential use cases for this LHM mechanism include dynamic hypervisor instrumentation for debug tracing or performance analysis. This new mechanism can be beneficial in providing more advanced features like dynamic tracing. The key motivation for this work is to provide a mechanism for such dynamic modification at the hypervisor level. This mechanism can be used in providing more advanced features like runtime customization. It is also useful for supporting dynamic debugging and tracing services. To overcome the issues, the creation of a hypervisor-level dynamic instrumentation facility can be used. This new hypervisor mechanism will allow probes to be loaded at runtime, which can be used to add additional code to existing functions or provide a full replacement for the given function.

The process migration is used to transfer an active process between two machines and restoring the process from the point it left off on the selected destination node. The purpose is to provide for an enhanced degree of dynamic load distribution, fault resilience, eased system administration, and data access locality. Potential of process migration is great especially in a large network of personal workstations [6]. In a typical working environment, there generally exists a large pool of idle workstations whose power is unharnessed if the potential of process migration is not tapped. Providing for a reliable and efficient migration module that helps handle inconsistencies in load distribution and allows for an effective usage of the system resource potential is highly warranted.

Hyper Safe, a lightweight approach that endows existing bare-metal hypervisors with a unique self-protection capability to provide lifetime control flow integrity. Specifically, technique are used. The non by passable memory lockdown and restricted pointer indexing. In non-by passable memory lockdown reliably protects the hypervisor's code and static data from being compromised even in the presence of exploitable memory corruption bugs (e.g., buffer overflows), therefore successfully providing hypervisor code integrity. The restricted pointer indexing introduces one layer of indirection to convert the control data into pointer indexes. These pointer indexes are restricted such that the corresponding call/return targets strictly follow the hypervisor control flow graph, hence expanding protection to control-flow integrity [7].

Server consolidation enables multiple servers, each of which is encapsulated in a virtual machine to concurrently run on a single physical machine and to be dynamically relocated among different physical machines [8]. It achieves high resource utilization by alleviating waste of underutilized resources, thereby magnifying the number of servers a data center can accommodate.

A new post copy strategy for live VM migration, previously studied only in the context of process migration. At a high-level, post-copy migration defers the memory transfer phase until after the VM's CPU state has already been transferred to the target and resumed there. Post-copy first transmits all processor state to the target, starts the VM at the target, and then actively pushes the VM's memory pages from source to target. Concurrently, any memory pages that are faulted on by the VM at target, and not yet pushed, are demand-paged over the network from source. Post-copy thus ensures that each memory page is transferred at most once, thus avoiding the duplicate transmission overhead of pre-copy [9]. Effectiveness of post-copy depends on the ability to minimize the number of network-bound page-faults (or network faults), by pushing the pages from source before they are faulted upon by the VM at target. To reduce network faults, supplement the active push component of post-copy with adaptive prepaging. Prepaging is a term of optimizing memory-constrained disk-based paging systems. It traditionally refers to a more proactive form of pre-fetching from storage devices (such as hard disks) in which the memory subsystem can try to hide the latency of high-locality page faults by intelligently sequencing the pre-fetched pages. Modern virtual memory subsystems do not typically employ prepaging due increasing DRAM capacities. Although post-copy doesn't deal with disk-based paging, the prepaging algorithms themselves can still play a helpful role in reducing the number of network faults in post-copy. Prepaging adapts the sequence of actively pushed pages by using network faults as hints to predict the VM's page access locality at the target and actively push the pages in the neighbourhood of a network fault before they are accessed by the VM and hence proposed and compare a number of prepaging strategies for post-copy, which Call bubbling, that reduce the number of network faults to varying degrees. The Microwiper, a practical approach to efficiently propagate memory in live migration of virtual machines. Two main strategies are used in Microwiper. First, we propose ordered propagation to transfer dirty pages in the order of their rewriting rates. Second, a transfer throttle is designed to avoid memory hot spot in data transferring. It estimates available network bandwidth in sending pages for traffic control. After the accumulated rewriting rate exceeds the estimated bandwidth, the next iteration is started immediately [10]. We build a prototype Microwiper system by retrofitting the live migration in Xen hypervisor. Experimental results show that compared with pre-copy based migration in Xen, Microwiper can not only reduce dirtied pages, but also shorten service downtime and total migration time. Specifically, Microwiper reduces downtime and transferred pages by more than 50%..

III. LIVE MIGRATION

Live Migration allows a server administrator to move a running VM or application between different physical machines without disconnecting the client or application. The use of virtual machine (VM) migration technology for data centers management has attracted significant attention in the recent years. The capability of migrating live virtual ma-chine among distinct physical hosts provides a significant new benefit for multiple VM-based environments in many key scenarios:

Minimize downtime: In migration terminology, downtime is the time duration during which service is unavailable to the client due to unavailability of a running instance of a virtual machine.

Minimize total migration time: Total migration time is the time period between start of migration and end of it when the virtual machine is fully migrated and starts running on the new host.

Minimize resource contention**:** Time concerned between active services and migrating virtual machine.

IV. ARCHITECTURAL DESIGN

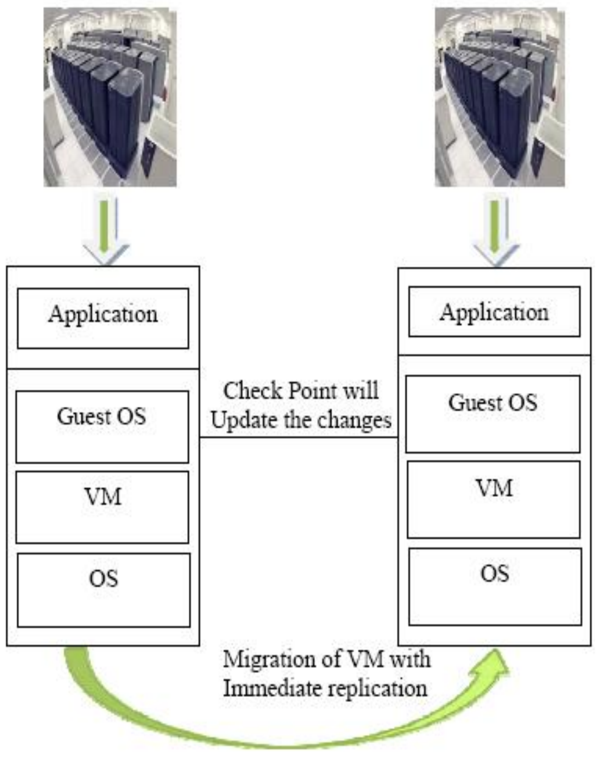
Application
Guest OS
VM
OS
Check Point will Update the changes
Application
Guest OS
VM
OS
Migration of VM with Immediate replication

Fig. 1 : Architectural diagram

V. FUNCTIONAL MODULE DESIGN

The functional modules of this paper are listed as follows:

- Physical Server Communication Implementation
- Live Raw Migration
- Establishment of Networks with Systems
- Disaster Recovery Site
- Synchronization checks

A: Physical Server Communication Implementation:

Step 1: Physical server communication is the implementation of source server for the disaster site.

Step 2: It will have an operating system with the virtual machine configuration.

Step 3: The operating system level cluster will be configured.

B: Live Raw Migration:

Step 1: Live virtual migration from the physical server to the Destination server.

Step 2: Raw migration of virtual machine

Step 3: Operating system synchronization

C: Establishment of Networks with Systems:

Step 1: Physical server cluster implementation

Step 2: DRS cluster verification

Step 3: Physical server and disaster recovery site cluster configuration.

D: Disaster Recovery Site:

Step 1: Disaster recovery site configuration for optimized uptime for the servers

Step 2: Replication of the operating system from physical server.

E: Synchronization checks:

Step 1: Cluster implementation checks

Step 2: Synchronization of operating system

Step 3: Mutual communication between the servers.

VI. EXPERIMENTAL SETUP

The experiments are performed on two machines of same configuration, Intel i3 processor and 3 GB RAM, connected via a fast Ethernet. We have used Fedora 16 as the guest OS. The host kernel is the modified version of Xen. The virtual machine is migrated within reduced downtime and the total data transferred is increased within the reduced amount of time.

VII. PERFORMANCE EVALUATION

Fig 2 shows that compared with normal pre-copy migration, the Copy on Write checkpointing has greater performance improvement on total data transferred. It also reduces network traffic.

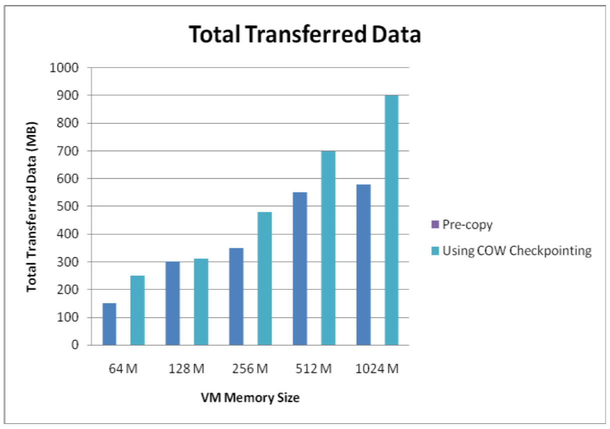

Fig 2: Total transferred data of Pre-copy and COW Checkpointing for VM memory size

VIII. CONCLUSION

Thus this project provides quick migration of physical memory image, network connections and other applications with reduced downtime. The total migration time and total data transferred is also calculated, which shows that network traffic of virtual migration have reduced and the performance is increased. The checkpoint validations and resuming activities in virtual machine migration is used to recover long running desktop application.

IX. REFERENCES

[1] Hai Jin, Li Deng, Song Wu, Xuanhua Shi, Xiaodong Pan "Live Virtual Machine Migration with Adaptive Memory Compression" IEEE International conference on Cluster Computing and Workshops, 2009.

[2] Gao Xiaopeng, Wang Sumei,Chen Xianqin " VNSS: A Network Security Sandbox For Virtual Computing Environment" International conference on networking and security,2010.

[3] Gao Xiaopeng, Wang Sumei,Chen Xianqin " VNSS: A Network Security Sandbox For Virtual Computing Environment" International conference on networking and security,2010.

[4] Tsuyoshi Ota, Shusuke Okamoto "Using I/O schedulers to reduce I/O load in virtualization environments" Workshops of International Conference on Advanced Information Networking and Applications, 2011.

[5] Thomas Naughton, Geoffroy Vall´ee, Ferrol Aderholdt "Loadable Hypervisor Modules" Proceedings of the 43rd Hawaii International Conference on System Sciences, 2010.

[6] Nalini Vasudevan, Prasanna Venkatesh “Design and Implementation of a Process Migration System for the Linux Environment”

[7] Zhi Wang, Xuxian Jiang “HyperSafe: A Lightweight Approach to Provide Lifetime Hypervisor Control-Flow Integrity” IEEE Symposium on Security and Privacy, 2010.

[8] Hwanju Kim, Heeseung Jo, and Joonwon Lee XHive “Efficient Cooperative Caching for Virtual Machines” IEEE Transactions on Computers, vol. 60, no. 1, January 2011.

[9] Michael R. Hines, Umesh Deshpande, and Kartik Gopalan “Post-Copy Live Migration of Virtual Machines” ACM SIGOPS Operating Systems, Vol 43, 2009.

[10] Yuyang Du Hongliang Yu, Guangyu Shi, Jian Chen, Weimin Zheng “Microwiper: Efficient Memory Propagation in Live Migration of Virtual Machines” 39th International Conference on Parallel Processing,2010.

◈◈◈