ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

A technical review on comparison of Xen and KVM hypervisors: An analysis of virtualization technologies

Ms Jayshri Damodar Pagare [1], Dr. Nitin A Koli [2]

Research Scholar, Sant Gadge Baba Amravati University, Amravati, India [1]

Head, Computer Centre, Sant Gadge Baba Amravati University, Amravati, India [2]

Abstract: This paper presents the recent technical survey on comparison of Xen and KVM Hypervisors. Hypervisors are widely used in cloud environments and virtualization through the use of hypervisors has become widely used. This paper reviews in depth analysis of virtualization technologies experimented by researchers from feature comparison to performance analysis. This paper will be useful for researchers to work on appropriate hypervisors.

Keywords: Virtualization, Hypervisors.

I. INTRODUCTION

Virtualization, a mechanism to abstract the hardware and system resources from a given Operating System. It is typically performed within a Cloud environment across a large set of servers using a Hypervisor or Virtual Machine Monitor (VMM), which lies in between the hardware and the OS. From the hypervisor, one or more virtualized OSs can be started concurrently as seen in Figure1, leading to one of the key advantages of Cloud computing.
This, along with the advent of multi-core processors, allows for a consolidation of resources within any data center. From the hypervisor level, Cloud computing middleware is deployed atop the virtualization technologies to exploit this capability to its maximum potential while still maintaining a given QoS and utility to users.[1]

Virtualization is the process of decoupling hardware from the operating system on a physical machine. It is one of the main technologies used for improving resource efficiency in datacenters which allows the deployment of co-existing computing environments over the same hardware infrastructure. A *Virtual Machine (VM)* is the virtualized representation of a physical machine that is run and maintained on a host by a software virtual machine monitor or *hypervisor* [2]. The hypervisor implements the virtualization on the physical machine and can be one of two types. *Type 1* hypervisors are sometimes referred to as native hypervisors as they run on "bare metal," or directly on the host's hardware to control the hardware and to monitor guest operating-systems. *Type 2* hypervisors are hosted hypervisors, meaning they are software applications running within a conventional operating-system environment.

Xen [3] is an example of a Type 1 hypervisor. Xen provides full virtualization to partition the host machine into multiple VMs. Xen can also use a technique known as *paravirtualization*, where the operating system is aware of virtualization and works with the hypervisor, to improve

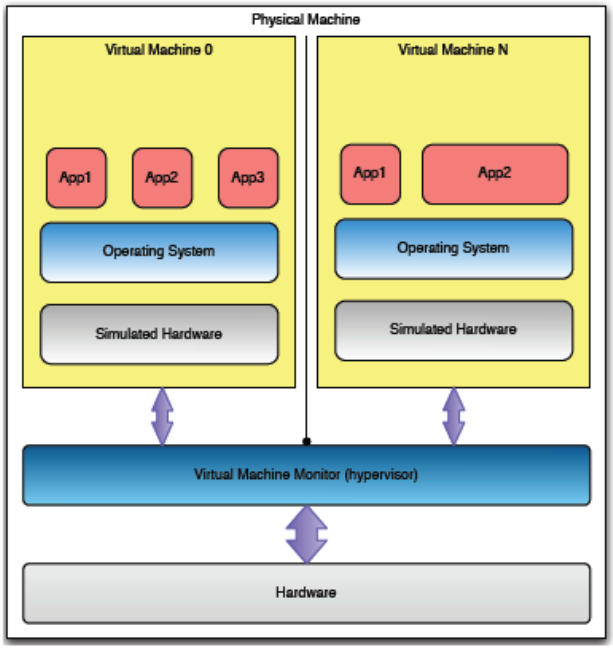

Fig. 1. Virtual Machine Abstraction[1]

the efficiency of operation. VMs provide many advantages to increase efficiency and decrease cost to data enters; however, VMs have significant security implications. A primary concern is ensuring the proper virtual environment is operating within a VM and whether the VM is configured properly. Another concern is virtual machines specific vulnerabilities that can be exploited to mount attacks specialized to subvert the built in defences of the guest operating systems.

As all virtual machines have a standard interface and appear identical to the software running within it, we consider all attacks that can take advantage of that interface. Examples of such attacks include an adversary copying a virtual machine to maliciously run multiple copies of licensed software on multiple computers since their environments are identical, thus breaking the

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

licensing agreement. Another problem of abstracting the real hardware interface from the guest VM is that it has no binding to physical hardware, thus disallowing it to determine its physical location and being able to decide whether the host environment is safe. By not knowing what else is operating on the platform, the guest VM cannot determine the trust level of the host platform. If the authenticity of the VM is not known and the trust of host environment cannot be determined, the guest VM cannot be trusted.

II. BACKGROUND OF HYPERVISORS[4]

A. Xen hypervisor

Xen is an open source hypervisor originally developed at the University of Cambridge and now distributed by Citrix Systems, Inc. The first public release of Xen occurred in 2003. It is designed for various hardware platforms, especially x86, and supports a wide range of guest operating systems, including Windows, Linux, Solaris and versions of the BSD family.

Xen architecture consists of a hypervisor, a host OS and a number of guests. In Xen terminology, the host OS is referred to as the Domain 0 (dom0) and the guest OS is referred to as Domain U (domU). Dom0 is created automatically when the system boots. Although it runs on top of the hypervisor itself, and has virtual CPUs (vCPUs) and virtual memory, as any other guest, Dom0 is privileged. It provides device drivers for the I/O devices, it runs user space services and management tools, and is permitted to use the privileged control interface to hypervisor. It also performs all other OS related functions. Like a DomU, Dom0 can be any operating system in the Linux ,Solaris, and BSD family. As we see, Xen separates the hypervisor execution from Dom0. In this way, the series of tasks that are not related to processing the virtualized guests performed by Dom0 do not influence the hypervisor, ensuring maximum performance.

Xen employs para-virtualization from the very beginning. Through para-virtualization, Xen can achieve very high performance, but it has the disadvantage of supporting Linux only; and that Linux has to have a modified kernel and bootloader, and a fixed layout with two partitions, one for hard disk and one for swap.

Xen also implements support for hardware-assisted virtualization. In this configuration, it does not require modifying the guest OS, which make it possible to host Windows guests.

B. KVM hypervisor

KVM is a hardware-assisted virtualization developed by Qumranet, Inc and was merged with upstream mainline Linux kernel in 2007, giving the Linux kernel native virtualization capabilities. KVM make use of the virtualization extensions Intel VT-x and AMD-V. In 2008, Red Hat, Inc acquired Qumranet.

KVM is a kernel module to the Linux kernel, which provides the core virtualization infrastructure and turns a Linuxhost into a hypervisor. Scheduling of processes and memory is handled through the kernel itself. Device emulation is handle by a modified version of QEMU . The guest is actually executed in the user space of the host and it looks like a regular process to the underlying host kernel.

A normal Linux process has two modes of execution: kernel mode and user mode. KVM adds a third one: guest mode. When a guest process is executing non-I/O guest code, it will run in guest mode. All the guests running under KVM are just regular linux processes on the host. Each and every virtual CPU of your KVM guests is implemented using a Linux thread. The Linux scheduler is responsible for scheduling a virtual CPU, as it is a normal thread. This brings the advantage that you can set priorities and affinity for these processes using normal adjusting commands, and use all the common linux utilities related to processes. Also, control groups can be created and used to limit resources that each guest can consumeon a host. In KVM, guest physical memory is just a chunk of host virtual memory. So it can be swapped, shared, backed by large pages, backed by a disk file, and is also NUMA aware.

KVM supports I/O para-virtualization using virtio subsystem. Virtio is a virtualization standard for device (network, disk, etc.) drivers where the guest's device driver is aware of running in a virtual environment, and communicates directly with the hypervisor. This enables the guests to get high performance network and disk operations. Virtio is different,but architecturally similar to Xen para-virtualized device drivers.

III. GENERAL DIFFERENCES BETWEEN XEN AND KVM[5]

Xen consists of a thin software layer above the hardware, which mostly controls CPU and memory access (see Figure2a). It implements its own CPU scheduling and memory management. There is a special, trusted virtual machine, called Domain-0, running control software the whole system is administrated through. I/O tasks are delegated to Domain-0. This means that all other virtual machines (or "unprivileged domains", Domain-U) communicate with peripheral devices indirectly, through Domain-0.

KVM has a somewhat different architecture (Figure 2b). It coexists with the host OS, which is an ordinary operating system (GNU/Linux in this case) and provides a set of system calls (ioctl-s) for creating and managing virtual machines from userspace.

Every virtual machine is seen by the host OS as an ordinary process and is treated accordingly. Note that KVM, as a kernel extension, is not a complete solution itself: it is only an extension to the standard Linux kernel that provides the infrastructure for

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

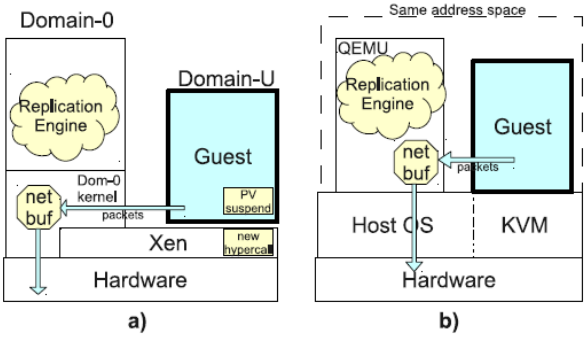

Fig. 2. Comparison of architectures: Remus (a) and Romulus (b)[5]

running and managing virtual machines. There is a tool in userspace (a modified version of QEMU emulator) that takes advantage of this infrastructure to provide a complete virtualization solution (although the combination of the two is most commonly called just KVM, for simplicity).

An important difference between Xen and KVM is in the virtualization technique used. Namely, making an operating system run in a virtualized environment is not a straightforward task. In typical architectures, such as x86, most instructions can be executed natively in a virtualized environment, but there are some problematic instructions that need special treatment. A possible approach to solve this problem in software is called *paravirtualization*, which means changing the source code of an operating system in order to make it virtualization-aware. Since 2006 there have appeared commercial processors capable of solving this problem in hardware, by introducing special modes of execution for use with virtualization. This approach is called *full virtualization*. KVM supports full virtualization, with a limited support for paravirtualization in form of device drivers . In fact, this is the main reason for its simplicity: KVM only makes hardware virtualization extensions accessible from userspace through a set of system calls. Xen supports both paravirtualization and full virtualization, but the former is used whenever possible because it usually gives better performance.

IV. THE FEATURE COMPARISON

Virtualization technology provides a way to share computing resources among VMs by using hardware/software partitioning, emulation, time-sharing, and dynamic resource sharing. Traditionally, the operating system (OS) controls the hardware resources, but virtualization technology adds a new layer between the OS and hardware. A virtualization layer provides infrastructural support so that multiple VMs (or guestOS) can be created and kept independent of and isolated from each other. Often, a virtualization layer is called a hypervisor or virtual machine monitor (VMM). While virtualization has long been used in mainframe systems , VMware has been the pioneer in bringing virtualization to commodity x86 platforms, followed by Xen and a variety of other virtualization platforms .[6]

Figure 3 shows three different approaches to virtualization: para-virtualization (PV), full virtualization (FV), and hardware-assisted virtualization (HVM). Paravirtualization requires modification to the guest OS, essentially teaching the OS how to make requests to the hypervisor when it needs access to restricted resources. This simplifies the level of hardware abstraction that must be provided, but version control between the hypervisor and paravirtualized OS is difficult since they are controlled by different organizations. Full virtualization supports running unmodified guests through binary translation.

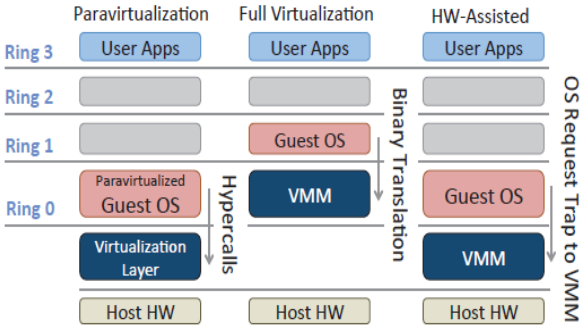

Fig. 3. Different Approaches to Providing the Virtualization Layer[6]

These techniques can incur large overheads since instructions that manipulate protected resources must be intercepted and rewritten. As a result, Intel and AMD have begun adding virtualization support to hardware so that the hypervisor can more efficiently delegate access to restricted resources .Some hypervisors support several of these techniques; hypervisors using hardware-assisted virtualization as this promises to offer the greatest performance and flexibility. Our target hypervisors are KVM and Xen. Both of these hypervisors use different architectures.

Xen and KVM use open-source modifications of the Linux kernel, Xen was initially based on the paravirtualization technique, but it now supports HVM as well .However, it still retains a separate management domain (dom0) which controls VMs and can negotiate access to custom block and network drivers. KVM runs as a kernel module, which means it uses most of the features of the linux. kernel operating system itself. For example, rather than providing its own CPU scheduler for VMs, KVM treats each VM as a process and uses the default Linux scheduler to allocate resources to them.

A variety of software and operating system aspects can affect the performance of hypervisors and VMs. In particular,how the hypervisor schedules resources such as CPU,memory, disk, and network are critical factors. Each of these resources requires different techniques to virtualize, leading to performance differences in each hypervisor depending on the types of activities being performed. Table I summarizes performance-related elements for both these hypervisor

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 12, December 2014

TABLE I. Feature Comparison (* DEFAULT OR USED IN THIS PAPER)[6]

Features	KVM	XEN
Base OS	Linux(+QEMU)	Linux(+QEMU)
Latest Release Version	2.6.32-279	4.1.2
Architecture	Bare metal	Hosted
Supported virtualization technologies	Para-virtualization,full virtuaization,hardware-assisted virtualization	Para-virtualization,full virtuaization,hardware-assisted virtualization
Network Management features	FIFO based scheduling	FIFO based scheduling
CPU Scheduling Features	Linux schedulers (Completely Fair Queuing Scheduler*, round-robin, fair queuing, proportionally fair scheduling, maximum throughput, weighted fair queuing)	SEDF (Simple Earliest Deadline First), Credit*
Memory Address Translation Mechanism	Shadow page table, Hardware-Assisted Pagetable*	Direct Pagetable (PV mode), Shadow Pagetable (HVM mode), Hardware-Assisted Pagetable*

V. PERFORMANCE COMPARISON

Xianghua et al.[7] have performed series of tests that stress a variety of system sources including CPU, memory, process creation, disk I/O and network I/O, here host one Netperf as server and create four the same guest os in their own virtual machine as client. After establishing some baseline data, they run a variety of different stress tests in virtual machines and report the impact on Netperf performance in all four virtual machines. The results shown in the Table II.

A. CPU Intensive Test

The first test stressed CPU usage with a tight loop containing both integer and floating point operations. Both virtualization systems performed well on this test even the misbehaving VMs. On both platforms that the CPU load on the misbehaving ever does rise to nearly 100%.[7]

B. Memory Bomb

Test began with a stress test which simply loops constantly allocating and touching memory. After this test, In the Xen and Kvm case, the misbehaving VM did not report results, but all others continued to report nearly 100% good results as before.[7]

C. Fork Bomb

Next ran a program that loops, creating new child processes. As in the memory consumption test, under Xen, Kvm, the misbehaving virtual machine presented no results[7].

D. Disk Intensive Test

For a disk intensive stress test, they use IOzone , next ran threads of IOzone each running an alternating read and write pattern (iozone -i 0 -i 1 -r 4 -s 64M -t 10).The results of this test were quite amazing. On Xen, the situation was mixed. The misbehaving VM saw a degradation of 15% and the other three VMs were also impacted, showing an average degradation of 2-3%. With Xen was proposed hardware access model, a specialized device driver VM could be written to ensure quality of service guarantees for each client VM .On KVM, good performance maintained on the three well-behaved VMs was close to 100%. However, the misbehaving VM saw a degradation of 26%[7].

E. Server Transmits Data

For the server transmitting test, next started 4 threads which each constantly sent 60K sized packets over UDP to external receivers. For this test, the results were once again mixed. For Xen, the well-behaved VMs show almost no degradation and the misbehaving VMs shows a slight but repeatable degradation of less than 1%. Under KVM, the well-behaved VMs show a slight but repeatable degradation of less than 1%, but the misbehaving VM presented no results[7].

F. Server Receives Data

Finally, for the server transmitting test,next started 4 threads which each constantly read 60K packets over UDP from external receivers. The results for this test were similar to the server transmit test on Xen. On Xen, there was a slight but repeatable degradation the misbehaving VM, but all the wellbehaving VMs were unaffected. For Kvm all four VMs retained 100% good response. There is no substantial degradation on the misbehaving VM that was seen in the sender transmit case[7].

TABLE II. SUMMARY OF STRESS TEST RESULTS[7]

	XEN		KVM	
	Normal	Stress	Normal	Stress
CPU	0	0.02	0	0.03
MEMORY	0	DNR	0	DNR
FORK	0.02	DNR	0	DNR
DISK Intensive	2.61	15.67	0.01	26.2
NETWORK Sender	0.02	0.43	0.14	DNR
NETWORK Receiver	0.03	0.12	0	0

Percent of degradation in good response rate. For each test, the percent degradation for either the bad or misbehaving VM is shown, as well as, the average degradation across the three good or well-behaving VMs .DNR indicates the Netperf client reported only an error and no actual results because of the unresponsiveness of the server it was testing.

VI. CONCLUSION

Here we conducted a survey of virtualization technologies namely XEN ,KVM. By taking Results of various researchers tests for both the virtualization technologies ,we conclude that there is no perfect hypervisor, and both the hypervisor handles different workload which is best suited for them.

REFERENCES

[1] Younge, A.J.; Henschel, R.; Brown, J.T.; von Laszewski, G.; Qiu, J.; Fox, G.C., "Analysis of Virtualization Technologies for High Performance Computing Environments," *Cloud Computing (CLOUD), 2011 IEEE International Conference on* , vol., no., pp.9,16, 4-9 July 2011

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

[2] S. Campbell, and M. Jeronimo, *Applied Virtualization Technology*, Hillsboro,OR: Intel Press 2006.

[3] P. Barham, B. Dragovic, K. Fraser, "Xen and the Art of Virtualization," *ACMSIGOPS Operating System Review*,vol.37,no.5,pp.164-177,2003.

[4] Soriga, S.G.; Barbulescu, M., "A comparison of the performance and scalability of Xen and KVM hypervisors," *Networking in Education and Research,2013 RoEduNet International Conference 12th Edition* , vol., no., pp.1,6, 26-28 Sept. 2013

[5] Petrovic, D.; Schiper, A., "Implementing Virtual Machine Replication: A Case Study Using Xen and KVM," *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on* , vol., no., pp.73,80, 26-29 March 2012

[6] Jinho Hwang; Sai Zeng; Wu, F.Y.; Wood, T., "A component-based performance comparison of four hypervisors," *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on* , vol., no., pp.269,276, 27-31 May 2013

[7] Xianghua Xu; Feng Zhou; Jian Wan; Yucheng Jiang, "Quantifying Performance Properties of Virtual Machine," *Information Science and Engineering, 2008. ISISE '08. International Symposium on* , vol.1, no., pp.24,28, 20-22 Dec. 2008