Virtualization of IT Infrastructure for Small and Medium Businesses

Ghannam Aljabari
Palestine Polytechnic University, Hebron, Palestine, Email: galjabari@ppu.edu

***Abstract*—Small and medium businesses (SMBs) are adopting virtualization to improve server utilization and consolidation. However, virtualization can also be used to create a virtualized environment for the whole IT infrastructure. This paper presents a review of the various virtualization techniques and open source implementations. These techniques enable SMBs of all sizes to create a viralized environment for their IT systems and their own infrastructure. In this context, SMBs can move forward with virtualization beyond just servers and also virtualize the entire network infrastructure that were traditionally hardware bound. As a result, SMBs can reduce their IT budgets and the time spent on routine IT administration tasks.**

***Index Terms*—virtualization; hypervisor; virtual infrastructure; virtual network; virtual storage; virtual WLAN; virtual data center**

I. Introduction

Virtualization is quickly becoming the platform of choice for users and businesses that want to reduce power and hardware cost and be able to increase resource sharing and utilization. Today, system virtualization is very popular for servers and desktops. With system virtualization, several virtual machines (VMs) can run simultaneously on a commodity hardware.

With the adoption of system virtualization, a new layer of network virtualization is emerging that provides inter- and intra- VM connectivity and has many of the same functionality provided by the physical networking hardware. This new layer also provides many of the features that are not available in physical networks such as software flexibility and *live migration* which provides the ability to move a running VM between physical machines with no interruption to service. In addition, multiple virtual networks can be constructed within a single physical machine or across multiple physical machines. However, the way of deploying and managing virtual networks is different from physical networks.

The network virtualization layer primarily consists of virtual interfaces, used by individual VMs, and virtual switches, which connect the VMs to each other. System virtualization can also be used to create virtual routers, virtual firewalls, and other virtual appliances for Internet connectivity. As a result, virtualization provides an integrated software environments to emulate the entire network infrastructure for production deployment, or development and testing purposes, i.e., virtualization allows the testing of different IT systems prior to production, which reduces real risks of failure.

By means of virtualization, small and medium businesses (SMBs) that have already implemented system virtualization can also virtualize the entire network infrastructure that were traditionally hardware bound. SMBs that have not yet applied virtualization to their IT systems and network infrastructure can also reduce the IT budget and the downtime, so it can significantly improve SMBs' availability and business continuity.

This paper shows how open source virtualization techniques that are available today can be used to create a virtualized environment for the whole IT infrastructure. The virtual infrastructure environment is designed for SMBs to provide several different IT services in parallel and on the same hardware infrastructure, mainly to reduce costs and increase utilization.

II. System Virtualization

System virtualization enables multiple VMs to run concurrently on the same physical machine with different operating systems (OSs) and applications [1]–[3]. Thus making the OSs and applications unaware of the underlying hardware, yet viewing computing resources as shared resource pools available via virtualization. The term *guest* is usually used to refer to the VM while the *host* is used to refer to the hosting environment.

There are several techniques to system virutalization and several software implementations, both open source and commercial. However, open source solutions provide free alternatives for commercial virtualization software like VMware products and Microsoft Hyper-V. Virtualization software abstracts the underlying hardware by creating VMs, which represent virtualized resources such as CPUs, physical memory, network interfaces, and I/O devices [1], [4]. The virtualization software is known as a Virtual Machine Monitor (VMM) or *hypervisor* [1].

There are two models for the hypervisor, *hosted* and *native*. In the hosted model, a hypervisor runs as an application on top of an OS and supports a broad range of hardware configurations. In contrast, a native (bare-metal) hypervisor runs directly on the hardware to control the hardware and to manage guest OSs. Since it has direct access to the hardware resources rather than going through an OS, a bare-metal hypervisor is more efficient than a hosted model and delivers greater scalability, robustness and performance [3].

A key challenge for system virtualization is the handling of privileged instructions to virtualize the CPU on x86 architecture [3]. Privileged instructions include all those that change the allocation of the shared resources such as halt the machine, set the timer, set the program counter, and I/O related

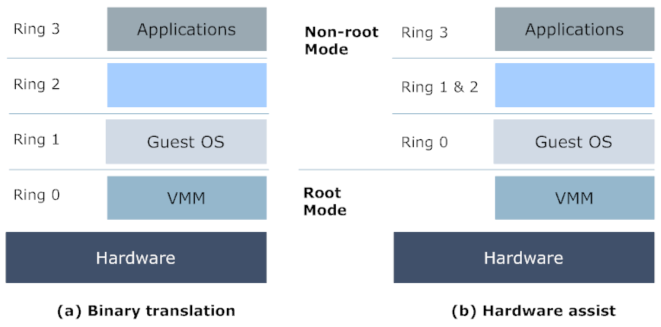

Fig. 1. Full virtualization techniques

instructions. The x86 architecture offers four levels of privilege known as *rings*, numbered from 0 to 3. While applications typically run in Ring 3 (user mode), the operating system needs to have direct access to hardware and must execute its privileged instructions in Ring 0 (kernel mode). In user mode, only non-privileged instructions can be executed. However, if a privileged instruction is executed in user mode, an interrupt is generated and control is passed to an interrupt handling routine, which is part of the OS [5].

Today, two common alternative approaches exist to resolve this challenge:

A. Full Virtualization

In this approach, VMs and guest OSs run on top of virtual hardware provided by the VMM. However, the VMM has to provide the VM with an image of an entire system, including virtual BIOS, virtual memory, and virtual devices to allow the guest OS to run without modification. As a result, the guest OS or application is not aware of the virtual environment [1], [3].

The main advantage of full virtualization approach is that it supports any platform and also provides a complete isolation of different applications, which helps make this approach highly secure. However, this approach has poor performance in trying to emulate a complete set of hardware in software [1], [3].

Virtualbox [6], an open source virtualization software, relies on a technique called *binary translation* to provide a fully virtualized machine. The binary translation technique, as shown in Fig. 1, allows the VMM to run in Ring 0 for isolation and performance, while moving the guest OS to a user level ring with greater privilege than applications in Ring 3 but less privilege than the VMM in Ring 0. The VMM translates all guest OS instructions in the memory and caches the results for future use, while user level instructions are executed directly on the CPU [3].

KVM [7], which stands for Kernel-based Virtual Machine, is a bare-metal hypervisor that takes advantage of *hardware-assist* features on x86 architecture such as Intel VT and AMD-V to improve the performance of guest OSs. The first generation of hardware assist features was added to processors in 2006, so that KVM hypervisor supports only newer x86 hardware systems [8].

Using KVM, several fully VMs can be created and operated in Linux environments, since KVM adds VMM capabilities to the Linux kernel. By adding virtualization support to Linux kernel, the virtual environment can benefit from all the ongoing work on the Linux kernel itself. Thus, researchers can focus their efforts on optimizing Linux and KVM for the VM environment not replicating OS functions within the hypervisor [9].

When using hardware assist features, additional operating modes, root and non-root mode, are added to CPU architecture to virtualize privileged instructions. Both of these modes support the four privilege rings just like the CPU architecture without virtualization features. As depicted in Fig. 1, the VMM operates in root mode and has access to real hardware, while the guest OS operates in non-root mode and its access to hardware is under complete control of the VMM [5], [8].

While these hardware assist features reduce the overhead for virtualizing the CPU, an additional amount of resources are expended by the hypervisor in handling memory virtualization. Because the guest OS cannot directly access the physical host memory, the hypervisor must provide a virtualized memory implementation in which the hypervisor provides mapping between the physical memory and the virtual memory. This is often implemented using shadow page tables within the hypervisor.

Intel developed the Extended Page Table (EPT) feature and AMD developed the Rapid Virtualization Indexing (RVI) feature. These features provide a virtualized memory management unit (MMU) in hardware that delivers significant performance improvements compared to the software only implementation. The memory virtualization features are incorporated into the recent generation of Intel and AMD processors.

KVM officially became part of the mainline Linux kernel as of version 2.6.20. KVM hypervisor consists of two main components: a set of kernel modules providing the core virtualization infrastructure such as CPU and memory management, and a userspace program that provides device emulation for I/O hardware devices, currently through a modified version of QEMU (Quick Emulator). QEMU provides an emulated BIOS, PCI bus, USB bus and a standard set of devices such as IDE and SCSI disk controllers, network cards, etc.

KVM hypervisor can also be used with SPICE (Simple Protocol for Independent Computing Environments) [10] to provide a complete solution for *virtual desktop infrastructure*. SPICE is an open source remote computing software, providing client access to remote VM display and devices (e.g., keyboard, mouse, audio). SPICE achieves a user experience similar to an interaction with a local machine, while trying to offload most of the intensive CPU and GPU tasks to the client.

B. Paravirtualization

Paravirtualization or OS assisted virtulaiuzation presents each VM with an abstraction of the hardware that is similar but not identical to the underlying physical hardware [2]. This approach requires modifications to the guest OSs that are running on the VMs in order to replace privileged instructions with hypercalls that communicate directly with the VMM. As a result, the guest OSs are aware that they are executing on a VM, allowing for near-native performance [1].

Xen [11] is an open source virtualization software based on paravirtualization approach. Xen hypervisor runs directly on hardware, allowing the host machine to run multiple modified guest OSs concurrently. Modifying the guest OS is not feasible for non-open source platforms such as Microsoft Windows [4]. As a result, such OSs are not supported in a paravirtualization environment. Recently, unmodified guest OSs are also supported by Xen. In this mode, Xen provides fully abstracted VM with hardware support (Intel VT and AMD-V) referred to as *hardware virtual machine* (HVM).

Xen architecture includes three components: the VMM, the privileged domain guest referred to as Domain0 or Dom0, and unprivileged domain guests referred to as DomainU or DomU. Dom0 has a unique privilege to access the hardware through secure interfaces and to manage all aspects of DomU such as starting, stopping and I/O requests [4]. For many years, Linux has been used in Dom0 as a management OS on top of the VMM and there was a Linux patch to transform the Linux kernel into this Dom0. With the release of Linux kernel 3.0, Xen hypervisor has become part of the mainline Linux kernel, allowing domain guests to run without the need to apply the patch to the kernel.

III. Network Virtualization

In recent years, network virtualization is moving toward virtualized environments. By allowing multiple logical networks to co-exist on a shared physical infrastructure, network virtualization provides flexibility and manageability [12]. Network virtualization often combines hardware and software resources to deploy virtual networks for different topologies. Over the years, the term *virtual network* has been used to describe different types of network virtualization such as VLAN (Virtual Local Area Network) and VPN (Virtual Private Network). Today, this layer is providing connectivity to tens of VMs per physical machine. However, networking in virtual environments impose a set of challenges that are not available in physical networks such as scaling and isolation [13].

Modern OSs provide the ability to create virtual network interfaces that are supported entirely in software. From the OS's point of view, these interfaces behave similar to physical network interfaces. However, the virtual interface does not send the packets into a wire, but makes them available to userspace programs running on the system. Virtual network interfaces are commonly referred to as TAP and TUN interfaces in Linux OS. TAP interfaces operate with Layer 2 packets, while TUN interfaces can handle Layer 3 packets. VMs use TAP interface to create a network bridge with the physical network interface [14].

Full virtualization approach provides virtual network interfaces (NIC) by emulating legacy Ethernet devices for simplicity. The virtual network interfaces appear to the VM as virtualized hardware devices within the hypervisor. In this technique, no modification is required for the guest OS. However, there is a significant performance overhead due to the context switching between VM and hypervisor. In the paravirtualization approach, the para-virtualized driver is used in the guest OS to achieve high I/O performance. But, this method requires modifying the guest OS and having a special driver to expose some details of the hardware [15].

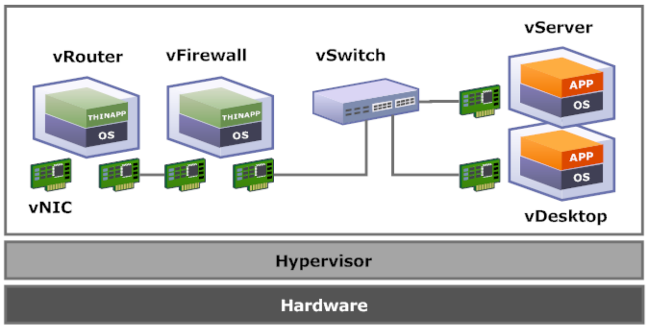

Fig. 2. Virtual infrastructure components

Most of the virtualization approaches also provide some form of virtual networking. For example, VMware has a distributed switch for virtual machine networking [16]. Linux-based virtualization platforms, including Xen and KVM, generally use network bridging or Virtual Distributed Ethernet (VDE) switch. Network bridge acts like an Ethernet hub; passing all traffic. While, VDE provides Layer 2 switching, including spanning-tree protocol and VLAN support [17].

Open vSwitch [18] is an open source software switch that provides connectivity between the VMs and the physical interfaces. It implements standard Layer 2 and Layer 3 switching with advanced features such as traffic monitoring (e.g. NetFlow), port mirroring (e.g. SPAN), basic ACL (Access Control List) and QoS (Quality of Service) policies. The Open vSwitch consists of two components: a fast kernel module and lightweight userspace program. The kernel module implements the forwarding engine, while the userspace program implements forwarding logic and configuration interfaces. Open vSwitch supports multiple Linux-based virtualization software, including Xen and KVM [13].

Quagga [19] is an open source routing software that provides implementations of TCP/IP based routing protocols such as OSPF, RIP, and BGP. In addition to traditional IPv4 routing protocols, Quagga also supports IPv6 routing protocols. XORP (Extensible Open Router Platform) [20] is another open source routing software that support most of IPv4 and IPv6 routing protocols.

Vyatta software [21] incorporates open source routing and security projects such as Quagga, IPtables, OpenVPN and many others into a network OS for x86 hardware platforms. Vyatta also can be delivered as VMs, providing routing, firewalling, VPN, and more for virtual environments. Thus, Vyatta network OS complements virtual networking components by delivering the virtual router, virtual firewall, and virtual VPN gateway. A typical virtual infrastructure is shown in Fig. 2.

IV. Storage Virtualization

Storage virtualization is the application of virtulization to storage services or devices for the purpose of aggregating functions or devices, hiding complexity, or adding new capabilities to lower level storage devices. Today, several techniques are employed to virtualize storage functions, which include physical storage, RAID groups, logical unit numbers (LUNs), storage zone, volume management, file systems and database objects. Hence, storage virtualization provides an abstraction layer for physical storage resources [22].

Most of the work in storage virtualization in recent years has focused on *block virtualization*. With block virtualization, several physical disks are presented as a single logical device [22]. There are two common methods to deliver virtual storage pools with block virtualization or aggregation: *host-based* and *network-based*.

Host-based method uses the volume manager included on the host OS to deliver storage virtualiztion. The volume manager provides a common way for managing and allocating storage space on physical devices. Most OSs have a basic volume manager such as LVM (Logical Volume Manager) in Linux and LDM (Logical Disk Manager) in Windows. Third-party volume manager products are also available such as VMware VMFS. The advantage of this method that it consolidates storage resources made from heterogeneous storage systems.

Network-based virtualization is the most common method for delivering virtual storage. It depends on a *storage area network* (SAN) to connect and manage storage systems using Fibre Channel (FC) switch or iSCSI storage networking protocol. This method has the advantage of providing a single management interface for all virtulized storage.

File system virtualization is another type of storage virtualization, which is used to manage shared network access to files in the file system. The simplest form of file system virtualization is the concept of a *network attached storage* (NAS) such as NFS and CIFS.

GlusterFS [23] is an open source cluster file system, distributed across multiple systems and aggregates the total storage into a single storage pool. A GlusterFS cluster exposes this pool as an NFS or CIFS mount point. The benefit of this model is that the underlying storage becomes fully virtualized and can be distributed as widely as required. GlusterFS has a client and a server component. The server stores the data, and the client connect to servers with a custom protocol over TCP/IP to access the data. The important feature of GlusterFS is that it is a pure software solution, able to run on commodity storage hardware.

V. Wireless LAN Virtualization

Wireless LAN virtualization is considered as an alternative approach for deploying multiple virtual wireless LANs with different security mechanisms. Wireless LAN virtualization enables several virtual WLAN interfaces to co-exist on a common shared physical device. All virtual interfaces operate concurrently without considering the physical nature of the wireless LAN medium as well as physical management functions. Each virtual interface abstracts a single wireless LAN device and has its own wireless network and its own unique MAC address [24].

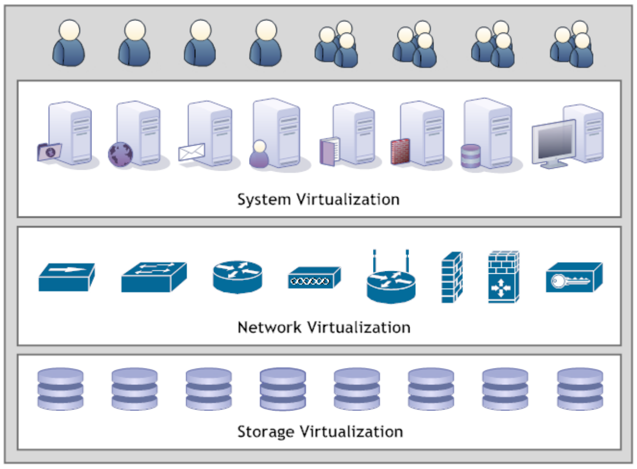

Fig. 3. Virtual data center

A virtual WLAN interface can be configured to operate as an access point (AP) and also as a station (STA) device. A virtual AP is bound to a virtual interface and each virtual AP independently keeps the configuration and service of the wireless LAN. In this way, several virtual APs can be configured on top of only one physical wireless LAN device.

In general, the virtual AP consists of two parts: control plane and forwarding plane. The control plane is concerned with the information that defines the functionality of the AP such as the SSID (Service Set Identifier), and RF (Radio Frequency) channel. While the forwarding plane defines the part of the AP, that forwards packets to its destination [25].

hostapd [26] is an open source software for controlling wireless LAN authentication and association. It implements IEEE 802.11 AP management and provide support for several security mechanisms such as WPA, IEEE 802.11i, and IEEE 802.1X. The current version of hostapd support Linux OS and OpenBSD. The functionality of wireless AP or router is emulated by integrating hostapd with a VDE switch to enable MAC forwarding similar to physical AP, and then can be assigned to a virtual router to enable IP forwarding and routing [24].

VI. Data Center Virtualization

Virtual data center (VDS), as shown in Fig. 3, is a fully-isolated virtual infrastructure environment where a user or a group of users can create and manage VMs, virtual networks and storage pools. In addition, a set of ACLs is defined to allow different role management for shared infrastructure resources.

Virtualization of data center is the underlying technology for *cloud computing*. Cloud computing provides on-demand network access to a shared pool of computing resources like servers, storage and networking. With cloud computing, IT providers can deliver the computing infrastructure as a service

(IaaS). Thus, eliminating the need for cloud consumers to deploy and manage the hardware infrastructure [27].

OpenNebula [28] is an open source management software for data center virtualization. OpenNebula provides a virtual infrastructure environments for IT enterprises to build their own *private cloud* using their internal infrastructures. The OpenNebula architecture includes several components specialized in different aspects of virtual infrastructure management such as image and storage technologies, virtual networking, and the underlying hypervisor for creating and managing VMs. OpenNebula provides different interfaces (APIs) that can be used with the underlying hypervisor, including KVM and Xen. OpenNebula also supports a *hybrid cloud* model by using cloud drivers to interface with *public clouds* or commercial cloud providers [29].

OpenStack [30] is an open source platform for building private and public IaaS clouds. It includes three core software projects to orchestrate a cloud: OpenStack Compute, OpenStack Object Storage, and OpenStack Image Service. These projects are designed for large-scale deployments of virtual machines, virtual netoworks, and virtual disk images. Open Stack provides drivers and APIs required to interact with underlying hypervisors, including KVM and Xen.

VII. Conclusion

In this paper, we present the various virtualization techniques that let SMBs build their own IT infrastructure on a commodity hardware without the need to modify their IT systems or services. Also, we present open source solutions and technologies that can be used as building blocks to create and manage virtual IT infrastructure. The benefits of IT infrastructure virtualization for SMBs, make the implementation of VITI (Virtualization of IT Infrastructure) project an excellent platform for the development and testing of different virtual IT services.

In the future, we plan to execute VITI project to study the impact of virtualization on IT infrastructures and services including email, file sharing and security systems. In addition, the project provides a complete open source platform for SMBs to build their own IT as a service. Using this platform allows SMBs to focus on their businesses rather than managing their IT infrastructure and services .

References

[1] J. Sahoo, S. Mohapatra, and R. Lath, “Virtualization: A survey on concepts, taxonomy and associated security issues,” in *Second International Conference on Computer and Network Technology (ICCNT)*, 2010, pp. 222–226.

[2] P. Barham and et al., “Xen and the art of virtualization,” in *ACM Symposium on Operating Systems Principles (OSSP*, 2003, pp. 164–177.

[3] “Understanding full virtualization, paravirtualization, and hardware assist,” VMware, 2007, http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf.

[4] T. Abels, P. Dhawan, and B. Chandrasekaran, “An overview of xen virtualization,” http://www.dell.com/downloads/global/power/ps3q05-20050191-Abels.pdf.

[5] D. Menasc, “Virtualization: Concepts, applications, and performance modeling,” in *International Computer Measurement Group (CMG) Conference*, 2005, pp. 407–414.

[6] https://www.virtualbox.org.

[7] http://www.linux-kvm.org.

[8] A. Kivity, “Kvm: The linux virtual machine monitor,” in *Ottawa Linux Symposium (OLS)*, 2007, pp. 225–230.

[9] I. Habib, “Virtualization with kvm,” in *Linux Journal*, vol. 2008, no. 166, 2008, http://www.linuxjournal.com/article/9764.

[10] http://spice-space.org.

[11] http://xen.org.

[12] N. M. N. K. Chowdhury and R. Boutaba, “Network virtualization: State of the art and research challenges,” in *IEEE Communications Magazine*, 2009, pp. 20–26.

[13] B. Pfaff and et al., “Extending networking into the virtualization layer,” in *8th ACM Workshop on Hot Topics in Networks (HotNets-VIII)*, 2009.

[14] H. Coskun, I. Schieferdecker, and Y. Al-Hazmi, “Virtual wlan: Going beyond virtual access points,” in *Electronic Communications of the EASST*, vol. 17, 2009.

[15] L. Xia and et al., “Virtual wifi: Bring virtualization from wired to wireless,” 2011.

[16] “Vmware vnetwork distributed switch,” VMware, http://www.vmware.com/files/pdf/VMware-vNetwork-Distributed-Switch-DS-EN.pdf.

[17] R. Davoli, “Vde: Virtual distributed ethernet,” in *First International Conference on TRIDENTCOM*, 2005.

[18] http://openvswitch.org.

[19] http://www.quagga.net.

[20] http://www.xorp.org.

[21] http://www.vyatta.com.

[22] “Storage virtualization,” SNIA, http://www.snia.org/sites/default/files/sniavirt.pdf.

[23] http://www.gluster.org.

[24] G. Aljabari and E. Eren, “Virtualization of wireless lan infrastructures,” in *The 6th IEEE International Conference on IDAACS*, 2011.

[25] T. Hamaguchi, T. Komata, T. Nagai, and H. Shigeno, “A framework of better deployment for wlan access point using virtualization technique,” in *IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, 2010, p. 968973.

[26] http://hostap.epitest.fi/hostapd.

[27] P. Mell and T. Grance, “The nist definition of cloud computing,” *National Institute of Standards and Technology*, vol. 53, no. 6, 2009.

[28] http://opennebula.org.

[29] B. Sotomayor and et al., “Virtual infrastructure management in private and hybrid clouds,” *IEEE Internet Computing*, vol. 13, no. 5, 2009.

[30] http://openstack.org.