

Virtualization : A Novice Approach

Amithchand Sheety¹ Mahesh Poola² Pradeep Bhat³ Dhiraj Mishra⁴

^{1,2,3,4} Padmabhushan Vasantdada Patil Pratishthan's College of Engineering, Eastern Express Highway, Near Everard Nagar, Sion-Chunabhatti, Mumbai-400 022, India.

Abstract— Virtualization provides many benefits – greater efficiency in CPU utilization, greener IT with less power consumption, better management through central environment control, more availability, reduced project timelines by eliminating hardware procurement, improved disaster recovery capability, more central control of the desktop, and improved outsourcing services. With these benefits, it is no wonder that virtualization has had a meteoric rise to the 2008 Top 10 IT Projects! This white paper presents a brief look at virtualization, its benefits and weaknesses, and today's "best practices" regarding virtualization.

I. INTRODUCTION

Virtualization, in computing, is a term that refers to the various techniques, methods or approaches of creating a virtual (rather than actual) version of something, such as a virtual hardware platform, operating system (OS), storage device, or network resources.

Virtualization addresses IT's most pressing challenge: the infrastructure sprawl that compels IT departments to channel 70% of their budget into maintenance, leaving scant resources for business-building innovation. The difficulty stems from the architecture of today's X86 computers: they're designed to run just one operating system and application at a time. As a result, even small datacenters have to deploy many servers, each operating at just 5% to 15% of capacity—highly inefficient by any standard.

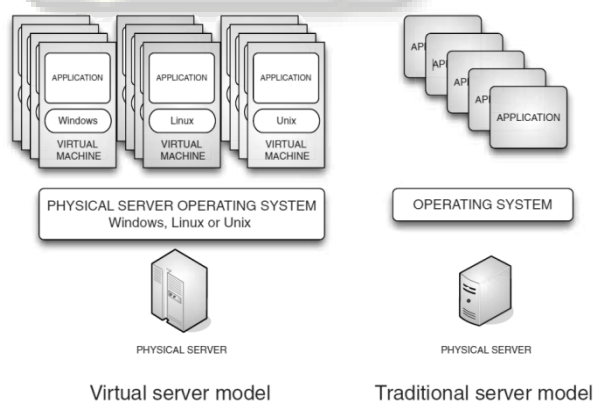


Fig. 1: Architecture of virtual computer

Virtualization solves the problem by enabling several operating systems and applications to run on one physical server or "host." Each self-contained "virtual machine" is isolated from the others, and uses as much of the host's computing resources as it requires.

II. REASON FOR VIRTUALIZATION

1) In the case of server consolidation, many small physical servers are replaced by one larger physical server to increase the utilization of costly hardware resources such

as CPU. Although hardware is consolidated, typically OS are not. Instead, each OS running on a physical server becomes converted to a distinct OS running inside a virtual machine. The large server can "host" many such "guest" virtual machines. This is known as Physical-to-Virtual (P2V) transformation.

- 2) Consolidating servers can also have the added benefit of reducing energy consumption. A typical server runs at 425W [4] and VMware estimates an average server consolidation ratio of 10:1.
- 3) A virtual machine can be more easily controlled and inspected from outside than a physical one, and its configuration is more flexible. This is very useful in kernel development and for teaching operating system courses.
- 4) A new virtual machine can be provisioned as needed without the need for an up-front hardware purchase.
- 5) A virtual machine can easily be relocated from one physical machine to another as needed. For example, a sales person going to a customer can copy a virtual machine with the demonstration software to his laptop, without the need to transport the physical computer. Likewise, an error inside a virtual machine does not harm the host system, so there is no risk of breaking down the OS on the laptop.
- 6) Because of the easy relocation, virtual machines can be used in disaster recovery scenarios.

III. EXAMPLES OF VIRTUALIZATION SCENARIOS

- 1) Running one or more applications that are not supported by the host OS: A virtual machine running the required guest OS could allow the desired applications to be run, without altering the host OS.
- 2) Evaluating an alternate operating system: The new OS could be run within a VM, without altering the host OS.
- 3) Server virtualization: Multiple virtual servers could be run on a single physical server, in order to more fully utilize the hardware resources of the physical server.
- 4) Duplicating specific environments: A virtual machine could, depending on the virtualization software used, be duplicated and installed on multiple hosts, or restored to a previously backed-up system state.
- 5) Creating a protected environment: if a guest OS running on a VM becomes damaged in a way that is difficult to repair, such as may occur when studying malware or installing badly behaved software, the VM may simply be discarded without harm to the host system, and a clean copy used next time.

IV. TRANSFORMING BUSINESS WITH VIRTUALIZATION

Improve the efficiency and availability of IT resources and applications through virtualization. Start by eliminating the old "one server, one application" model and run multiple virtual machines on each physical machine. Free your IT

admins from spending so much time managing servers rather than innovating. About 70% of a typical IT budget in a non-virtualized datacenter goes towards just maintaining the existing infrastructure, with little left for innovation.

An automated datacenter built on the production-proven VMware virtualization platform lets you respond to market dynamics faster and more efficiently than ever before. VMware vSphere delivers resources, applications—even servers—when and where they're needed. VMware customers typically save 50-70% on overall IT costs by consolidating their resource pools and delivering highly available machines with VMware vSphere.

- 1) Run multiple operating systems on a single computer including Windows, Linux and more.
- 2) Let your Mac run Windows creating a virtual PC environment for all your Windows applications.
- 3) Reduce capital costs by increasing energy efficiency and requiring less hardware while increasing your server to admin ratio.
- 4) Ensure your enterprise applications perform with the highest availability and performance.
- 5) Build up business continuity through improved disaster recovery solutions and deliver high availability throughout the data center.
- 6) Improve enterprise desktop management & control with faster deployment of desktops and fewer support calls due to application conflicts.

V. TYPES OF VIRTUALIZATION

A. Hardware Virtualization

Hardware virtualization or platform virtualization refers to the creation of a virtual machine that acts like a real computer with an operating system. Software executed on these virtual machines is separated from the underlying hardware resources. For example, a computer that is running Microsoft Windows may host a virtual machine that looks like a computer with the Ubuntu Linux operating system; Ubuntu-based software can be run on the virtual machine. [1][2]

In hardware virtualization, the host machine is the actual machine on which the virtualization takes place, and the guest machine is the virtual machine. The words host and guest are used to distinguish the software that runs on the physical machine from the software that runs on the virtual machine. The software or firmware that creates a virtual machine on the host hardware is called a hypervisor or Virtual Machine Manager.

- 1) Full virtualization: Almost complete simulation of the actual hardware to allow software, which typically consists of a guest operating system, to run unmodified.
- 2) Partial virtualization: Some but not the entire target environment is simulated. Some guest programs, therefore, may need modifications to run in this virtual environment.
- 3) Paravirtualization: A hardware environment is not simulated; however, the guest programs are executed in their own isolated domains, as if they are running on a separate system. Guest programs need to be specifically modified to run in this environment.

B. Desktop Virtualization

Desktop virtualization is the concept of separating the logical desktop from the physical machine.

One form of desktop virtualization, virtual desktop infrastructure (VDI), can be thought as a more advanced form of hardware virtualization. Rather than interacting with a host computer directly via a keyboard, mouse, and monitor, the user interacts with the host computer using another desktop computer or a mobile device by means of a network connection, such as a LAN, Wireless LAN or even the Internet. In addition, the host computer in this scenario becomes a server computer capable of hosting multiple virtual machines at the same time for multiple users.

VI. CAPABILITIES OF VIRTUALIZATION

A. Snapshotting

A snapshot is the state of a virtual machine, and, generally, its storage devices, at an exact point in time. Snapshots are "taken" by simply giving an order to do so at a given time, and can be "reverted" to on demand, with the effect that the VM appears (ideally) exactly as it did when the snapshot was taken.

The capability is, for example, useful as an extremely rapid backup technique, prior to a risky operation. It also provides the foundation for other advanced capabilities (discussed below).

Virtual machines frequently use virtual disks for storage. In a very simple case, for example, a 10 gigabyte hard disk is simulated with 10 gigabyte flat file. Any requests by the VM for a location on its physical disk (which does not "exist" as an actual physical object in and of itself) are transparently translated into an operation on the corresponding file (which does exist as part of an actual storage device).

Once such a translation layer is present, however, it is possible to intercept the operations and send them to different files, depending on various criteria. In a snapshotting application, every time a snapshot is taken, a new file is created, and used as an overlay. Whenever the VM does a write, the data is written to the topmost (current) overlay; whenever it does a read, each overlay is checked, working from the most recent back, until the most recent write to the requested location is found. In this manner, the entire stack of snapshots is, subjectively, a single coherent disk.

The RAM memory of the system can be managed in a similar way (though in the simplest systems, snapshots are disk-only, and the VM must be restarted).

Generally, referencing a snapshot means to reference that snapshot, and all prior snapshots on which it is based, down to the initial state when the VM was created.

To revert to a prior snapshot simply means to restart (or resume, if a memory state, processor state, and peripheral state snapshots are available in addition to disk states) the machine using only the overlays available up to a specific exact point in time (when the snapshot was taken, which resulted in new overlay files, rendering the ones that had been in use an instant before read-only), plus a new set of overlays to hold the current running state of the machine.

B. Teleportation

The snapshots described above can be moved to another host machine with its own hypervisor; when the VM is temporarily stopped, snapshotted, moved, and then resumed on the new host, this is known as migration. If the older snapshots are kept in sync regularly, this operation can be quite fast, and allow the VM to provide uninterrupted service while its prior physical host is, for example, taken down for physical maintenance.

C. Failover

Similar to teleportation above, failover allows the VM to continue operations if the host fails. However, in this case, the VM continues operation from the last-known coherent state, rather than the current state, based on whatever materials the backup server was last provided with.

D. Challenges

An often overlooked issue with virtualization is the complexities of licensing. For example, a server running a Linux OS attempting to offer a virtualized Windows Server must still satisfy licensing requirements. Therefore the benefits of on-demand virtualization and flexibility of virtualization is hampered by closed-source, proprietary systems. Some vendors of proprietary software have attempted to update their licensing schemes to address virtualization, but the flexibility vs. license cost issues are opposing requirements.

Virtualized desktop results in dependence on centralized servers (for computing and SAN storage) and the network (and higher-bandwidth requirements). This leaves the end users vulnerable to server, SAN, and network outages or capacity limits. The user may be able to continue operating locally through an outage, but becomes dead if the user logs off or reboots the machine. This is in contrast to thick-clients, where the user can continue to operate locally until the connectivity can be restored.

VII. VIRTUAL MACHINE

How Virtualization Works..?



Fig. 2: multiple OS running using virtual machine.

The heart of virtualization is the “virtual machine” (VM), a tightly isolated software container with an operating system and application inside. Because each VM is completely separate and independent, many of them can run simultaneously on a single computer. A thin layer of software called a hypervisor decouples the VMs from the

host, and dynamically allocates computing resources to each VM as needed.

A virtual machine is a tightly isolated software container that can run its own operating systems and applications as if it were a physical computer. A virtual machine behaves exactly like a physical computer and contains its own virtual (i.e., software-based) CPU, RAM hard disk and network interface card (NIC).

An operating system can't tell the difference between a virtual machine and a physical machine, nor can applications or other computers on a network. Even the virtual machine thinks it is a “real” computer. Nevertheless, a virtual machine is composed entirely of software and contains no hardware components whatsoever. As a result, virtual machines offer a number of distinct advantages over physical hardware.



Fig. 3: limiting the hardware configuration of the guest OS.

VIII. TECHNIQUES OF VIRTUALIZATION

A. Emulation of the underlying raw hardware (native execution)

This approach is described as full virtualization of the hardware, and can be implemented using a Type 1 or Type 2 hypervisor. (A Type 1 hypervisor runs directly on the hardware; a Type 2 hypervisor runs on another operating system, such as Linux). Each virtual machine can run any operating system supported by the underlying hardware. Users can thus run two or more different "guest" operating systems simultaneously, in separate "private" virtual computers.

The pioneer system using this concept was IBM's CP-40, the first (1967) version of IBM's CP/CMS (1967–1972) and the precursor to IBM's VM family (1972–present). With the VM architecture, most users run a relatively simple interactive computing single-user operating system, CMS, as a "guest" on top of the VM control program (VM-CP). This approach kept the CMS design simple, as if it were running alone; the control program quietly provides multitasking and resource management services "behind the scenes". In addition to CMS, in the early stage especially communication tasks were performed by multitasking VMs (RSCS, GCS, TCP/IP, UNIX), and users can run any of the other IBM operating systems, such as MVS, even a new CP itself or now z/OS. Even the simple CMS could be run in a threaded environment (LISTSERV,

TRICKLE).z/VM is the current version of VM, and is used to support hundreds or thousands of virtual machines on a given mainframe. Some installations use Linux for z Series to run Web servers, where Linux runs as the operating system within many virtual machines.

Full virtualization is particularly helpful in operating system development, when experimental new code can be run at the same time as older, more stable, versions, each in a separate virtual machine. The process can even be recursive: IBM debugged new versions of its virtual machine operating system, VM, in a virtual machine running under an older version of VM, and even used this technique to simulate new hardware.

The standard x86 processor architecture as used in the modern PCs does not actually meet the Popek and Goldberg virtualization requirements. Notably, there is no execution mode where all sensitive machine instructions always trap, which would allow per-instruction virtualization.

Despite these limitations, several software packages have managed to provide virtualization on the x86 architecture, even though dynamic recompilation of privileged code, as first implemented by VMware, incurs some performance overhead as compared to a VM running on a natively virtualizable architecture such as the IBM System/370 or Motorola MC68020. By now, several other software packages such as Virtual PC, VirtualBox, Parallels Workstation and Virtual Iron manage to implement virtualization on x86 hardware.

Intel and AMD have introduced features to their x86 processors to enable virtualization in hardware.

As well as virtualization of the resources of a single machine, multiple independent nodes in a cluster can be combined and accessed as a single virtual NUMA machine.

B. Emulation of a non-native system

Virtual machines can also perform the role of an emulator, allowing software applications and operating systems written for computer processor architecture to be run.

Some virtual machines emulate hardware that only exists as a detailed specification. For example:

- 1) One of the first was the p-code machine specification, which allowed programmers to write Pascal programs that would run on any computer running virtual machine software that correctly implemented the specification.
- 2) The specification of the Java virtual machine.
- 3) The Common Language Infrastructure virtual machine at the heart of the Microsoft .NET initiative.
- 4) Open Firmware allows plug-in hardware to include boot-time diagnostics, configuration code, and device drivers that will run on any kind of CPU.

This technique allows diverse computers to run any software written to that specification; only the virtual machine software itself must be written separately for each type of computer on which it runs.

C. Operating system-level virtualization

Operating system-level virtualization is a server virtualization technology which virtualizes servers on an

operating system (kernel) layer. It can be thought of as partitioning: a single physical server is sliced into multiple small partitions (otherwise called virtual environments (VE), virtual private servers (VPS), guests, zones, etc.); each such partition looks and feels like a real server, from the point of view of its users.

For example, Solaris Zones supports multiple guest OS running under the same OS (such as Solaris 10). All guest OS have to use the same kernel level and cannot run as different OS versions. Solaris native Zones also requires that the host OS be a version of Solaris; other OS from other manufacturers are not supported. [Citation needed], however you need to use Solaris Branded zones to use another OS as zones.

Another example is System Workload Partitions (WPARs), introduced in the IBM AIX 6.1 operating system. System WPARs are software partitions running under one instance of the global AIX OS environment.

The operating system level architecture has low overhead that helps to maximize efficient use of server resources. The virtualization introduces only a negligible overhead and allows running hundreds of virtual private servers on a single physical server. In contrast, approaches such as full virtualization (like VMware) and Paravirtualization (like Xen or UML) cannot achieve such level of density, due to overhead of running multiple kernels. From the other side, operating system-level virtualization does not allow running different operating systems (i.e. different kernels), although different libraries, distributions etc. are possible.

IX. VIRTUAL MACHINE CHARACTERISTICS

In general, VMware virtual machines possess four key characteristics that benefit the user:

- **Compatibility:** Virtual machines are compatible with all standard x86 computers
- **Isolation:** Virtual machines are isolated from each other as if physically separated
- **Encapsulation:** Virtual machines encapsulate a complete computing environment
- **Hardware independence:** Virtual machines run independently of underlying hardware

A. Compatibility

Just like a physical computer, a virtual machine hosts its own guest operating system and applications, and has all the components found in a physical computer (motherboard, VGA card, network card controller, etc). As a result, virtual machines are completely compatible with all standards x86 operating systems, applications and device drivers, so you can use a virtual machine to run all the same software that you would run on a physical x86 computer.

B. 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 superior to applications running in a traditional, non-virtualized system.

C. 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. For example, you can move and copy a virtual machine from one location to another just like any other software file, or save a virtual machine on any standard data storage medium, from a pocket-sized USB flash memory card to an enterprise storage area networks (SANs).

D. Hardware Independence

Virtual machines are completely independent from their underlying physical hardware. For example, you can configure a virtual machine with virtual components (e.g., 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).

When coupled with the properties of encapsulation and compatibility, hardware independence gives you the freedom to move a virtual machine from one type of x 86 computers to another without making any changes to the device drivers, operating system, or applications. Hardware independence also means that you can run a heterogeneous mixture of operating systems and applications on a single physical computer.

X. VIRTUAL MACHINE SOFTWARE

A. DOSBox

DOSBox is emulator software that emulates (loosely: “simulates”) an IBM PC compatible computer running the older operating system, MS-DOS. Many IBM PC compatible graphics and sound cards are also emulated. This means that original MS-DOS programs are provided an environment in which they can run correctly on many modern computers running a variety of operating systems. DOSBox is especially intended for use with old PC games. DOSBox is free software written primarily in C++ and distributed under the GNU General Public License.

B. Java Virtual Machine

A Java virtual machine (JVM) is a virtual machine that can execute Java byte code. It is the code execution component of the Java software platform. Sun Microsystems has stated that there are over 5.5 billion JVM-enabled devices.

A Java virtual machine is a program which executes certain other programs, namely those containing Java byte code instructions. JVMs are most often implemented to run on an existing operating system, but can also be implemented to run directly on hardware. A JVM provides an environment in which Java byte code can be executed, enabling such features as automated exception handling, which provides root-cause debugging information for every software error (exception), independent of the

source code. A JVM is distributed along with a set of standard class libraries that implement the Java application programming interface (API). These libraries, bundled together with the JVM, form the Java Runtime Environment (JRE).

C. VIRTUALBOX

Oracle VM VirtualBox (formerly Sun VirtualBox, Sun xVM VirtualBox and innotek VirtualBox) is an x86 virtualization software package, created by software company Innotek GmbH, purchased by Sun Microsystems, and now developed by Oracle Corporation as part of its family of virtualization products. Oracle VM VirtualBox is installed on an existing host operating system as an application; this host application allows additional guest operating systems, each known as a Guest OS, to be loaded and run, each with its own virtual environment.

D. VMWARE

VMware, Inc. (NYSE: VMW) is a company providing virtualization software, [1] [2][3] founded in 1998 and based in Palo Alto, California, USA. The company was acquired by EMC Corporation in 2004, and operates as a separate software subsidiary.

VMware's desktop software runs on Microsoft Windows, Linux, and Mac OS X, while VMware's enterprise software hypervisors for servers, VMware ESX and VMware ESXi, and are bare-metal embedded hypervisors that run directly on server hardware without requiring an additional underlying operating system. [4]

XI. CONCLUSIONS

The power and success of the Virtual Machine concept comes from the ability of users to access and utilize functions and devices that are simply combinations of instruction sets. The ability to provide a virtual solution to the real limitations of modern computer systems is a very powerful tool that is continuing to extend the abilities of modern computer systems.

The uses and benefits of virtualization are so varied and numerous, they are impossible to ignore. IT managers, web hosts, data center administrators, even home users, all stand to benefit from one or more of the available virtualization technologies. Industry will continue to adopt virtualization for many reasons: collections of inefficient servers can be replaced with fewer machines; software can be tested while isolated in harmless virtual partitions; and data centers can gracefully (and virtually) conform to shifting work models, new technologies and changing corporate priorities.

REFERENCES

- [1] Smith, James; Nair, Ravi (2005). "The Architecture of Virtual Machines". *Computer* (IEEE Computer Society) 38 (5): 32–38. doi:10.1109/MC.2005.173.
- [2] Matthew Chapman and Gernot Heiser. vNUMA: A virtual shared-memory multiprocessor. Proceedings of the 2009 USENIX Annual Technical Conference, San Diego, CA, USA, June, 2009.

- [3] James E. Smith, Ravi Nair, Virtual Machines: Versatile Platforms For Systems And Processes, Morgan Kaufmann, May 2005, ISBN 1-55860-910-5
- [4] Craig, Iain D. Virtual Machines. Springer, 2006, ISBN 1-85233-969-1
- [5] Turban, E; King, D; Lee, J; Viehland, D (2008). "Chapter 19: Building E-Commerce Applications and Infrastructure". Electronic Commerce A Managerial Perspective. Prentice-Hall. pp. 27.
- [6] Baburajan, Rajani, "The Rising Cloud Storage Market Opportunity Strengthens Vendors," infoTECH, August 24, 2011. It.tmcnet.com. 2011-08-24.
- [7] Oestreich, Ken, "Converged Infrastructure," CTO Forum, November 15, 2010. Thectoforum.com.
- [8] Virtualization in education". IBM. October 2007. Retrieved 6 July 2010.

