An Introduction to Virtualization

by Sean Campbell and Michael Jeronimo

Virtualization is one of the more significant technologies to impact computing in the last few years. With roots extending back several decades, today its resurgence in popularity has many industry analysts predicting that its use will grow expansively in companies over the next several years. Promising benefits such as consolidation of infrastructure, lower costs, greater security, ease of management, better employee productivity, and more, it's easy to see why virtualization is poised to change the landscape of computing.

But what exactly is virtualization? The term is used abundantly, and often confusingly, throughout the computing industry. You'll quickly discover after sifting through the literature that virtualization can take on different shades of meaning depending on the type of solution or strategy being discussed and whether the reference applies to memory, hardware, storage, operating systems, or the like.

Virtualization Defined

Virtualization refers in this article to the process of decoupling the hardware from the operating system on a physical machine. It turns what used to be considered purely hardware into software. Put simply, you can think of virtualization as essentially a computer within a computer, implemented in software. This is true all the way down to the emulation of certain types of devices, such as sound cards, CPUs, memory, and physical storage. An instance of an operating system running in a virtualized environment is known as a virtual machine. Virtualization technologies allow multiple virtual machines, with heterogeneous operating systems to run side by side and in isolation on the same physical machine. By emulating a complete hardware system, from processor to network card, each virtual machine can share a common set of hardware unaware that this hardware may also be being used by another virtual machine at the same time. The operating system running in the virtual machine sees a consistent, normalized set of hardware regardless of the actual physical hardware components. Technologies such as Intel®

Virtualization Technology (Intel® VT), which will be reviewed later in this article, significantly improves and enhances virtualization from the perspective of the vendors that produce these solutions.

With a working definition of virtualization on the table, here's a quick mention of some of the other types of virtualization technology available today. For example, computer memory virtualization is software that allows a program to address a much larger amount of memory than is actually available. To accomplish this, you would generally swap units of address space back and forth as needed between a storage device and virtual memory. In computer storage management, virtualization is the pooling of physical storage from multiple network storage devices into what appears to be a single storage device that is managed from a central console. In an environment using network virtualization, the virtual machine implements virtual network adapters on a system with a host network adapter. But again in the context of this book virtualization refers to the process of utilizing virtual machines.

Terminology

Individual vendors often choose terminology that suits their marketing needs to describe their products. Like the nuances of the virtualization technologies, it's easy to get confused over the different terms used to describe features or components. Hopefully as virtualization technology continues to evolve and as more players enter the marketplace, a common set of terminology will emerge. But for now, here is a list of terms and corresponding definitions.

Host Machine

A host machine is the physical machine running the virtualization software. It contains the physical resources, such as memory, hard disk space, and CPU, and other resources, such as network access, that the virtual machines utilize.

Virtual Machine

The virtual machine is the virtualized representation of a physical machine that is run and maintained by the virtualization software. Each virtual machine, implemented as a single file or a small collection of files in a single folder on the host system, behaves as if it is running on an individual, physical, non-virtualized PC.

Virtualization Software

Virtualization software is a generic term denoting software that allows a user to run virtual machines on a host machine.

Virtual Disk

The term refers to the virtual machine’s physical representation on the disk of the host machine. A virtual disk comprises either a singl e file or a collection of related files. It appears to the virtual mac hine as a physical hard disk. One of the benefits of using virtual machine architecture is its portability whereby you can move virtual disk files from one physical machine to another with limited impact on the files. Subsequent chapters illustrate various ways in which this can be a significant benefit across a wide var iety of areas.

Virtual Machine Additions

Virtual machine additions increase the performance of the guest ope rating system when compared to running without the additions, provide access to USB devices and other specialized devices, and, in some cases, to higher video resolutions than without the additions, thus offering an i mproved user interface experience within a virtual machine. The additions also allow the use of customizations such as shared folders, drag -and-drop copy and paste between the host and virtual machines and between virtual machines, and other enhancements.

One particularly useful enhancement is the ability of the mouse pointer’s focus to naturally move from the virtual machine window to the host machine’s active application windows without having to phys ically adjust it each time the window changes. This allows you to interact with the virtualized operating system as if it were nothing more than a nother application window, such as a word processing program running on the host machine.

Shared Folders

Most virtual machine implementations support the use of shared folders. After the installation of virtual machine additions, shared folders enables the virtual machine to access data on the host. Through a series of under -the-cover drive mappings the virtual machine can open up files and fol ders on the physical host machine. You then can transfer these files from the physical machine to a virtual machine using a standard mechanism such as a mapped drive.

Shared folders can access installation files for programs, data files, or other files that you need to copy and load into the virtual machine. With shared folders you don’t have to copy data files into each virtual m achine. Instead, all of your virtual machines access the sa me files through a shared folder that targets a single endpoint on the physical host m achine.

Virtual Machine Monitor (VMM)

A virtual machine monitor is the software solution that implements virtualization to run in conjunction with the host operating system. The vi rtual machine monitor virtualizes certain hardware resources, such as the CPU, memory, and physical disk, and creates emulated devices for virtual machines running on the host machine. An overview of emulated devices is presented later in this chapter. For now, it is important to understand that the virtual machine monitor determines how resources should be allocated, virtualized, and presented to the virtual machines running on the host computer. Many software solutions that exist today utilize this method of virtualization. Figure 1 illustrates the concept of the virtual machine monitor.

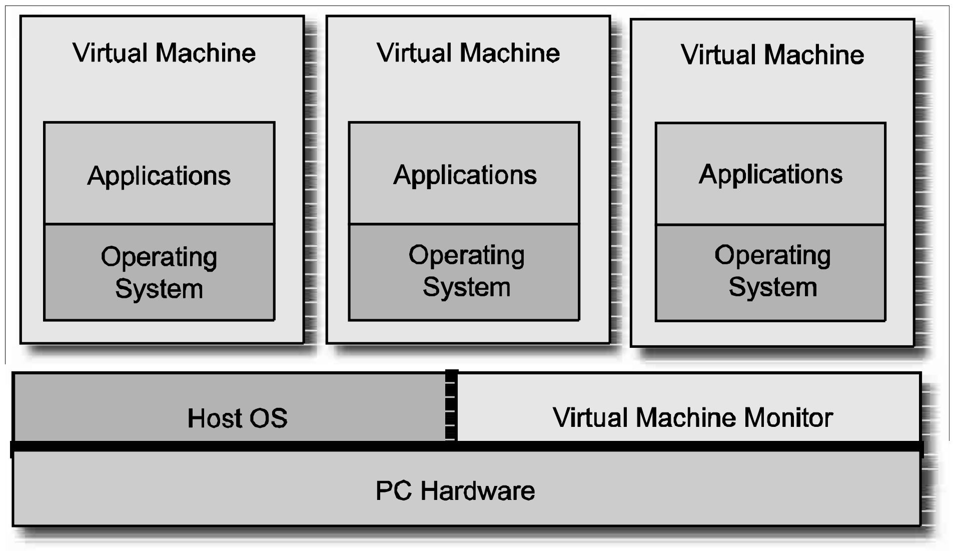

Figure 1 Virtual Machine Monitor Architecture

Hypervisor

In contrast to the virtual machine monitor, a hypervisor runs directly on the physical hardware. The hypervisor runs directly on the hardware without any intervening help from the host operating system to provide access to hardware resources. The hypervisor is directly responsible for hosting and managing virtual machines running on the host machine. However, the implementation of the hypervisor and its overall benefits vary widely across vendors.

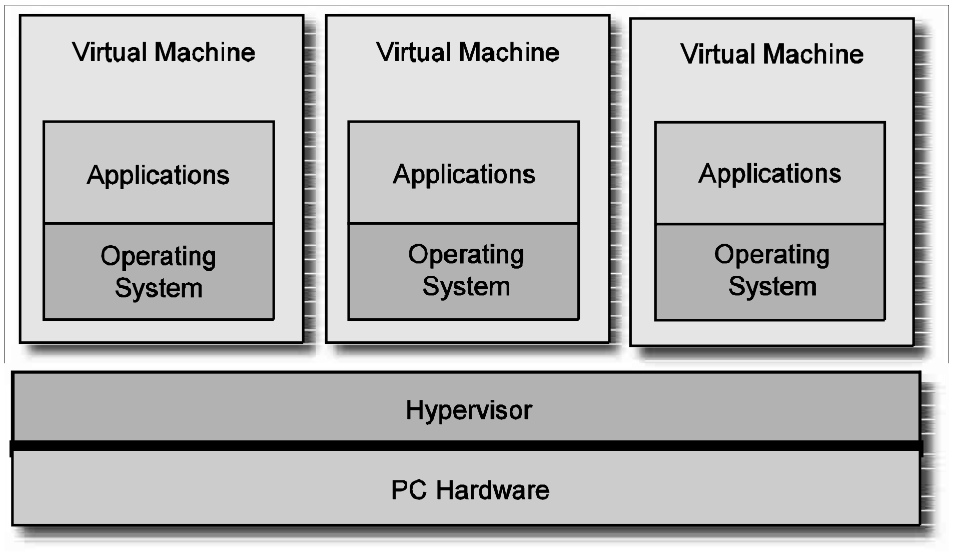

Figure 2 Hypervisor Architecture Overview

Paravirtualization

Paravirtuali zation involves modifying the operating system before it can be allowed to run in the virtualized environment as a virtual machine. Thus its use requires an open source operating system whose source is publicly avai lable.

Virtual Machine Isolation

While not strictly a technical term, the concept of virtual machine isol ation is important to understand. Virtual machines are essentially isolated from one another in the same way that two physical machines would be on the same network. A virtual machine's running operating system has no knowledge of other virtual machines running on the same machine. In some cases, the operating system itself has no way of knowing that it is running in a virtual ized environment either.

History of Virtualization

Before we place a foot firmly into the realm of virtualizat ion technologies that exist today, it's worthwhile to take a step back into history to e xplore the origin of virtualization within the mainframe environment. This is important because virtualization in its current incarnation is not a completely new technology and has roots in some past efforts.

From the 1950s to the 1990s

The concept of virtual memory dates to the late 1950s when a group at the University of Manchester introduced automatic page replacement in the Atlas system, a transistorized mainframe co mputer. The principle of paging as a method to store and transmit data up and down the memory hierarchy already existed but the Atlas was the first to automate the process, thereby providing the first working prototype of virtual me mory.

The term virtual machine dates to the 1960s. One of the earliest vi rtual machine systems comes from IBM. Around 1967, IBM introduced the System/360 model 67, its first major system with virtual me mory. Integral to the model 67 was the concept of a self -virtualizing processor instruction set, perfected in later models. The model 67 used a very early operating system called CP-67, which evolved into the virtual machine (VM) operating systems. VM allowed users to run several operating systems on a single processor machine. Essentially VM and the mainframe hardware cooperated so that multiple instances of any operating system, each with protected access to the full instruction set, could concurrently coexist.

In the mid 1960s IBM also pioneered the M44/44X p roject, exploring the emerging concept of time sharing. At the core of the system archite cture was a set of virtual machines, one for each user. The main machine was an IBM 7044 (M44 for short) and each virtual machine was an e xperimental image of the 7044 (44X for short). This work eventually led to the widely-used VM/timesharing systems, including IBM's well-known VM/370.

The concept of hardware virtualization also emerged during this time, allowing the virtua l machine monitor to run virtual machines in an isolated and protected environment. Because the virtual machine mon itor is transparent to the software running in the virtual machine, the software thinks that it has exclusive control of the hardware. The co ncept was perfected over time so that eventually virtual machine monitors could function with only small performance and resource overhead.

By the mid 1970s, virtualization was well accepted by users of various operating systems. The use of virtualization during these decades solved important problems. For example, the emergence of virtual storage in large-scale operating systems gave programs the illusion that they could address far more main storage (memory) t han the machine actually contained. Virtual storage expanded system capacity and made pr ogramming less complex and much more productive.

Also, unlike virtual resources, real system resources were extremely expensive. Virtual machines presented an efficient way to gain the maximum benefit from what was then a sizable investment in a co mpany's data center.

Although hardware -level virtual machines were popular in both the research and commercial marketplace during the 1960s and 1970s, they essentially disappeared during the 1980s and 1990s. The need for virtualization, in general, declined when low-cost minicomputers and personal computers came on the market.

Although not the focus of this article, another type of virtual machine, Sun Microsystems' Java Virtual Machine (JVM) and Microsoft's Common Language Runtime (CLR), deserve a place on the historical timeline and are worth mentioning here. The key thing to understand though is that these machines do not present a virtual hardware platform. But due to the potential confusion between this type of virtual machine and the virtual machines covered in this article a brief overview is in order to clear up these differences. These virtual machines emerged during the 1990s and extended the use of virtual machines into other areas, such as software development. Referred to as simulated or abstracted machines, they are implemented in software on top of a real hardware platform and operating system. Their beauty lies in their portability. In the case of JVM, compiled Java programs can run on *compatible* Java virtual machines regardless of the type of machine underneath the implementation.

Figure 1.3 outlines the relationship between a JVM or the CLR and the host operating system.

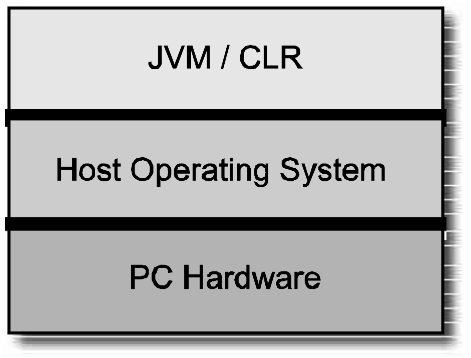

Figure 3 Runtime Virtual Machines

The Reemergence of Virtualization

The 1990s saw an explosion in the number of servers used throughout the enterprise. However, while their numbers continued to grow, many were underutilized in the workplace. Placing more than one application on a single server often was not a viable option even though that one application might use only a fraction of the server's available resources. Server proliferation presented deployment, update, and support challenges as well as issues with security and disaster recovery. Organizations soon realized that while waste and costs were escalating, productivity and efficiency were plummeting. The question became, "how do we consolidate our servers?" The answer was to use virtualization technology.

While the past several years have seen the re-emergence of virtualization, vendors have faced significant complications in developing the software to allow others to virtualize operating systems and applications. The advent of

Intel VT has removed or significantly reduced some of these complications. Intel recognized the re-emergence of virtualization and began working with VMM developers, implementing hardware assists in Intel processors and chipsets, and driving specifications to improve virtualiz ation in the future.

Challenges with the IA-32 Architecture and Software-Only Virtualization Solutions

So far the landscape of virtualization seems to be trouble free. But there is a crucial problem and th at concerns the original IA -32 architecture. It wasn't designed for virtualization. Intel processors were designed primarily to run a single instance of the operating system. So on systems that use Intel architecture, virtualization is presently a software -only solution. Here is a look at the problem and the various approaches used to solve the pro blem before the benefit of using Intel VT became available.

IA-32 Architecture and Privilege Levels

Intel processors provide protection based on various rings or privilege levels, numbered 0, 1, 2, and 3. The privilege level, 0 being the highest, determines what actions a specific process can perform. For example, memory mapping can be executed only in privilege level 0. In contrast, end-user applications run in pr ivilege level 3. Software running in a lower-numbered privilege level can exercise control over software ru nning at a higher-numbered privilege level. Most IA-32 software uses only privilege le vels 0 and 3.

Some of an operating system's components must run at privilege level 0 in order to have unlimited access to the underlying CPU. Similarly, in a virtualized system the virtual machine monitor (VMM) must ring in privilege level 0. The VMM must also create the illusion to the guest operating system that it, too, is running in ring 0. But the VMM cannot a llow a guest operating system such control because doing so might mo dify the VMM's code and data or give the guest operating system access to privileged instru ctions.

Before the availability of virtualization software, privilege levels would have been of little concern. To get around the conflict with priv ilege levels, the virtualization software relocates the guest operating sy stem to another ring—a technique known as *ring deprivileging*. Deprivileging is accomplished using one of two models. If the system uses the ring 0/1/3 model, the virtualization software deprivileges the guest operating system to privilege level 1. This allows the guest operating system to properly control its applications by locating the m in privilege level 3. In the 0/3/3 model the guest operating system is moved to privilege level 3 where it runs at the same privilege level as its applications. With either model, the VMM has privilege level 0 all to itself.

Unfortunately, deprivileging creates a new set of virtualization challenges. The VMM must constantly monitor the activities of the guest operating systems to trap attempts to access the hardware and certain system calls. It must execute these calls itself and emulate the results. For example, when software runs at a privilege level other than the one for which it was written, as in the case with the guest operating system, a problem referred to as *ring aliasing* can arise. Certain instruction calls authorized for use outside privilege level 0 can return a value that co ntains the current privilege level. The guest operating system is able to read the return value and determine that it is not running at privilege level 0. A conflict within the guest operating system could develop. However, since the call is a valid operation for an application running at privilege levels greater than 0, the VMM is unable to detect and provide the proper fix for this oper ation.

Another problem arises when the guest operating system, thinking it has control of the state of the CPU, makes a valid request for the state of the CPU. The CPU state returned is the true state of the CPU controlled by the VMM, not the simulated CPU state of the guest operating system. These values are in conflict and could cause exec ution failure.

The VMM that is in charge of the CPU must switch the context of the guest operating system process. A guest operating system is not gene rally written to support context switching and may store important data in hidden locations. When the VMM attempts to save the context, this i nformation can be lost. Restoring the complete context of the guest ope rating system would not be possible and the guest operating system would produce an execution failure. There are numerous other scenarios with advers e impacts.

Addressing the Virtualization Challenges

To address the virtualization challenges, designers of virtual machine monitors have developed two approaches: Paravirtualization and binary translation.

Paravirtualization

Briefly discussed earlier, this solution requires changes to the source code of the guest operati ng system, especially the kernel, so that it can be run on the specific VMM. Paravirtualization can be used only with operating systems that can be modified, such as Linux.

Binary Translation (or Patching)

With this approach the VMM makes changes to the binaries of the guest operating system as it is loaded into the virtual machine. This on -the-fly solution extends the range of operating systems that can be su pported as the operating system does not need to be modified to support this a pproach but comes with higher performance overhead than VMMs that use paravirtualization. This approach also requires a greater effort in some ways on the part of the designer of the VMM.

Intel® Virtualization Technology (Intel® VT)—Solving the Privilege Problem

Intel Virtualization Technology, a series of ha rdware-based processor and chipset innovations, delivers support to address some of the problems with software-only solutions. It enables VMMs to run off-the-shelf operating systems and appl ications and allows guest software to run at its intended privilege level, thereby eliminating the need for paravirtualization and b inary translation. Intel VT includes VT-x support for IA-32 processor virtualization and VT-i support for the Itanium® arch itecture. Here is a high-level look at the extensions to the IA-32 architecture.

Virtual Machine Extensions (VMX) Operations

VT-x augments the current IA-32 architecture with a new mode of CPU operation: VMX, which stands for virtual machine extensions. The VMM runs in VMX root operating level, which is fully privileged. Guest opera ting systems run in VMX non-root operating level. The key point is that both forms of operation support all four ring levels. The guest operating systems run within their expected ring levels and each thinks it controls the CPU; that is, the entire machine. The guest operating system in co nstrained, however, not by privilege level, but because it runs in VMX non-root operating level.

Two transitions are associated with VMX. These commands assoc iated with these transitions pass control back and f orth between the VMM and the guest operation systems:

- VM entry—VMM- to-guest transition, which enters VMX non-root operations
- VM exit—guest-to-VMM transition, which enters VMX root operations.

With the VM entry command, the guest operating system can exe cute VMX non-root operations. When the guest operating system passes co ntrol back to the VMM with the VM exit command, the VMM returns executing its privileged VMX root operations again. The virtual machine control structure is a new data structure that ma nages VM entries and VM exits.

Virtual Machine Benefits

Reducing hardware and software needs, improving performance and scalabil ity, and reducing downtime are key factors in managing costs in today's co mpanies. Virtual machines provide the means for companies to achieve these goals. Here is a brief overview of the benefits you can e xpect to gain using virtual machines. These benefi ts will be covered in depth later in this book in richer scenarios as well as in the context of other sc enarios.

- Virtual machines allow more efficient use of resources by consolidating multiple operating environments on underutilized servers onto a smaller number of virtualized servers.
- Virtual machines make the manageability of systems easier. For example, you do not need to shut down servers to add more memory or upgrade a CPU.
- The complexity of overall administration is reduced because each virtual machine's software environment is independent from the underlying physical server environment.
- The environment of a virtual machine is completely isolated from the host machine and the environments of other virtual machines so you can build out highly-secure environments that are tailored to your specifications. For example, you can configure a different security setting for each virtual machine. Also, any attempt by a user to interfere with the system would be foiled because one virtual environment cannot access another unless the virtualization stack allows this. Otherwise, it restricts access entirely.
- You can migrate old operating systems for which it is difficult to obtain appropriate underlying hardware for a physical machine. Along these same lines, you can run old software that has not been, or cannot be, ported to newer platforms.
- You can run multiple, different operating systems from different vendors simultaneously on a single piece of hardware.
- Because virtual machines are encapsulated into files you can easily save and copy a virtual machine. You can quickly move fully configured systems from one physical server to another.
- Virtualization allows you to deliver a pre-configured environment for internal or external deployment scenarios.
- Virtual machines allow for powerful debugging and performance monitoring. Operating systems can be debugged without losing productivity and without having to set up a more complicated debugging environment.
- The virtual machine provides a compatible abstraction so that all software written for it will run on it. For example, a hardware-level virtual machine will run all the software, operating systems, and applications written for the hardware. Similarly, an operating system–level virtual machine will run applications for that particular operating system, and a high-level virtual machine will run programs written in the high-level language.
- Because virtual machines can isolate what they run, they can provide fault and error containment. You can insert faults proactively into software to study its subsequent behavior. You can save the state, examine it, modify it, reload it, and so on. In addition to this type of

isolation, the virtualization layer can execute performance isolation so that resources consumed by one virtual machine do not necessarily affect the performance of other virtual m achines.

Multi-Core Technologies and Virtualization Technologies

One of the primary applications of virtualization technology involves running more than one operating system at the same time on one physical machine. Multiple operating systems are, in particular , necessary in development and testing situations where engineers must develop software simultaneously on different operating systems. They are also very common in IT scenarios where legacy operating systems need to run side by side with more modern systems. However with virtualization technology, an installed operating system such as Microsoft† Windows† is not designed to share hardware resources, such as processor, memory, disk space, network, and video, with other operating systems running at the same ti me on the same physical machine. To sidestep this constraint, the user had to, prior to the advent of virtualization, dual-boot (or tri-boot, and so on) the machine between the different operating systems such as Wi ndows XP and Linux†.

Dual booting gives the user the flexibility of using multiple operating systems but at the significant disadvantage of having to shut down one operating system completely before using another. In order to share core data files and documents, the user must store them in a l ocation available to each operating system regardless of which one is currently booted and in active use, which further reduces productivity and increases comple xity. While this is viable in some contexts it slows down the process of i nteracting with the host machine and in some contexts is simply not a viable solution as will be outlined in later chapters. In addition proces sing power must be wholly applied to the execution of one operating sy stem or other and cannot be easily split across all the operating systems you might want to run concurrently on the same machine. Virtualization makes it possible to remove all of these limitations.

By contrast, each virtualized operating system takes a portion of available resources such as CPU, memory, and physical di sk and uses them for its own user-specified tasks. However, sharing the same phys ical resources that previously would have been dedicated to one physical machine comes at a cost. The host machine that is running these virtua lized operating systems must have more resources than were previously allocated to a single m achine.

A possible solution to this dilemma may lie in the emergence of increased processing power. With today’s emphasis on multiple core arch itecture and Hyper-Threading Technology, these proce ssors can be best utilized when placed in an environment where virtualization is in heavy use. The additional core(s) these processors provide can be dedicated to individual virtualized operating systems to allow the optimum scale out of resources. Additio nal benefits such as separating defined user tasks into given virtualized operating systems can allow for more secure or hardened dedicated virtualized operating systems all operating on the same piece of physical hardware. The use of virtualization makes it possible to take full advantage of new processor architectures and proce ssors that go from dual -core to quad-core, eight-core, and beyond.

Hardware Utilization—Possible Performance Impacts

Virtualizing your infrastructure or even a small number of machines can have enormous benefits, but it can also affect the performance of your server, workstation, or mobile machine hardware even with adva nces such as multi-core processors. It is important to understand some of the trad eoffs that occur at the hardware level with virtualization. This section outlines them on a component-by-component basis.

Physical RAM, CPU, hard disk space, a nd networking all play a role in determining whether a host machine is prepared to run a virtual m achine-based application. Properly preparing your host machines prior to running virtual machines on them will help you achieve better stability, scalability and long-term performance for your virtual machines. When selecting a host, you'll need to ensure that it meets the virtual machine application's minimum hardware requirements and further that enough resources, parti cularly memory, are available for the number of virtual machines you want to run simultaneously on the host.

Here is a breakdown of the various hardware components that are the usual bottlenecks and what can be done to prevent them.

CPU

The CPU is one of the more significant bottlenecks in the system when running multiple virtual machines. All of the operating systems that are running on the host in a virtual machine are competing for access to the CPU. An effective solution to this problem is to use a mul ti-processor or, better, a multi-core machine where you can dedicate a core or more to a virtual machine. The technology to assign a given core to a virtual m achine image is not yet fully provided by current virtualization vendors but is expected to be available in the near future. In the absence of a multi -core processor, the next best step is to find the fastest processor available to meet your needs.

Memory

Memory also can be a significant bottleneck but its effect can be mit igated, in part, by selecting the best vendor for your virtualization solution because various vendors handle memory utilization differently.

Regardless of the vendor you chose, you must have a significant amount of memory—one that is roughly equivalent to the amount you would have assigned to each machine if they were to run as a physical machine. For example, to run Windows XP Professional on a virtual m achine, you might allocate 256 megabytes (MB) of memory. This is on top of the 256 MB recommended for the host computer, assuming Windows XP is the host.

This can mean in many cases that a base machine configuration comes out to approximately 1–2 gigabytes (GB) of memory or perhaps many more gigabytes for a server -based virtualization solution.

You can easily change memory configuration for a guest operating system that is virtualized. Typically this change is done from within the virtualization software itself and requires only a shutdown and restart c ycle of the virtual machine to take effect. Contrast this process with the requirement to manually install memory on each physical machine and you can see one of the benefits of virtualization technology.

Physical Disk

When it comes to virtualization, overall disk space utilization for each virtual machine isn't as great a concern as is the intelligent utilization of each physical drive. An additional important point to consider is the rotational speed of the drive in use. Because you may utilize multiple virtual machines on a single drive the rotational speed of the drive can have a dramatic affect on performance with greater drive speeds. For the best performance across most of the virtualization products today, consider implementing multiple disk drives and using the fastest drive possible, in terms of its rotation speed, for each drive.

One way to boost performance of a virtualized solution beyond just having a faster drive is to ensure that the host machine and its associated operating system have a dedicated physical hard drive, and that all virtual machines or potentially each virtual machine has a separate physical hard disk allocated to it.

Network

Network utilization can also present bottleneck issues, similar to those with memory. Even though the virtual machine doesn't add any significant amount of network latency into the equation, the host machine must have the capacity to service the network needs of all of the running virtual machines on the host machine. However as with memory you still need to apply the appropriate amount of network bandwidth and network resources that you would have if the machines were running on separate physical hardware.

You might need to upgrade your network card if you are running multiple virtual machines in an IT environment and all machines are experiencing heavy concurrent network traffic. But in most desktop virtualization scenarios you will find that the network is not the problem. Most likely the culprit is the CPU, disk, or memory.

Conclusion

Virtualization technology, while not new, is growing at a significant rate in its use on servers and desktop machines and long ago lost its connection to mainframe systems alone. While challenges do exist, such as the unification of terminology, the development of even more robust software solutions, and the implementation of greater device virtualization support, virtualization is still poised to make a significant impact on the landscape of computing over the next few years.

For more information about virtualization and Intel VT, please refer to the book *Applied Virtualization Technology* by Sean Campbell and Michael Jeronimo.

About the Authors

Sean Campbell has been a consultant working with Microsoft and Intel® technologies for more than a decade, specializing in custom application development and solutions for emerging and mainstream technologies.

Over his 20-year software career, **Michael Jeronimo** has been a developer, architect, and a staff software architect for Intel Corporation, where he developed concepts and projects for Intel's Digital Home effort and developed Internet security technology.