Part 1

Hyper-V

Chapter 1

Understanding Microsoft's Hypervisor

Just about every business today is either evaluating or implementing server *virtualization*, or partitioning a physical computer into multiple virtual computers. With hardware systems becoming so powerful, many applications do not require all the available horsepower that comes on a commodity server today. As a result, many companies are viewing server virtualization as a way to either save money by consolidating several underutilized systems onto larger, more efficiently utilized servers or create a dynamic data center that allows movement of virtual machines from one host to another as the needs dictate.

Consolidation will enable companies to save money on physical hardware purchases, possibly some licensing costs, and definitely power and cooling. From a development and test environment, it also speeds up the ability to set up test environments and restore to earlier points to rerun tests. These scenarios promise cost savings of various amounts. Other companies are looking at how virtualization will make it easier and faster to change their infrastructure as their business environment changes.

Windows Server 2008 provides everything needed to support server virtualization as an integrated feature of the operating system — Windows Server 2008 Hyper-V. Hyper-V is Microsoft's next-generation hypervisor-based server virtualization technology, following its earlier Virtual Server product.

Before getting into the details of various aspects of Windows Server 2008 Hyper-V, it will be helpful to understand a little of the history of virtualization and the architecture of Hyper-V. As with any software product, there are new versions and capabilities in the works. In fact, we started writing about the version 1 product and needed to add content to cover version 2. This is a very dynamic time for virtualization products.

In this chapter, you will learn about:

- Microsoft's history in virtualization
- Monolithic versus microkernelized virtualization architectures
- Hardware requirements

Virtualization History

Today, full server and hypervisor virtualization are being implemented or investigated by nearly every company. Based on this recent interest, you would guess that this is a new technology. But it is not. In the early 1970s, IBM released their first commercial version of full operating system environment virtualization on their IBM System/370 and named it VM/370. They had been running the precursor to VM/370 in their labs, and they even had another unsupported product they distributed to customers in the late '60s. VM/370 was an implementation of what is known as full virtualization, or a complete virtualization of the underlying hardware in software enabling the execution of all the software that could run on the physical hardware. If you want to read an interesting article about some of the early work done in virtualization, read the paper titled ''Formal requirements for virtualizable third generation architectures,'' published in 1974 in *Communications of the ACM*, the Association for Computing Machinery journal. ACM is a professional computing organization formed over 60 years ago. You can see that the virtualization roots go deep. If IBM was selling this technology over 30 years ago, why is it only really taking off now?

Other examples of full virtualization exist. For many years, device manufactures have been developing their software on virtualized instances of their devices. For example, take out your cell phone. Do you think that someone is actually writing code and debugging it directly on that device? Obviously, that does not seem likely. The phone manufacturers virtualize that operating system environment on another platform that has human interface devices such as keyboards and monitors and that can run powerful compilers. Think of all the computer chips that are used in computers and graphics cards and all those other devices. Do you think that the engineers who design those chips are actually creating physical chips as prototypes? No, they have massive CAD programs that allow them to put the circuits together inside a computer and then execute the logic of that virtual chip in the memory of another computer. Once they have built the virtual environment of that chip, they will transfer that design into a physical prototype, but most of the development work is done in a virtual manner.

So what makes the current versions of operating system virtualization so intriguing? IBM had a very good product with VM/370 — they still do, though it is now officially called z/VM. However, it runs only on IBM hardware and clones of IBM hardware, such as Fujitsu systems. This product works very effectively and is still in use today, but that is not where most of the servers are today.

x86

More recently, virtualization has moved out of the specialized use cases and into the general marketplace. IBM has a great product, but the number of IBM mainframe systems pales in comparison to the number of systems running the Intel x86 architecture (x86 refers to both the 32-bit and 64-bit processors from both Intel and AMD that are based on the original x86 instruction set). Millions of x86 systems in the world today are running Microsoft Windows and various Linux distributions. Those hardware platforms have continued to get more and more powerful as Intel and AMD continue to produce faster and more powerful chips. In fact, the Intel and AMD systems today are far more powerful than the IBM systems on which the original VM/370 was developed. The availability of excess capacity in the commodity systems of today has created the interest in virtualization.

When organizations first started running business applications on the x86 architecture, neither the hardware nor the operating systems were as robust and stable as the IBM mainframe and other minicomputer operating systems. As a result, it was common practice to install a single application on a server instead of running multiple applications on a single system as

organizations do on IBM mainframe and other minicomputer systems. And, because the x86 architecture systems cost so much less than the mainframe and minicomputers, organizations did not see a major problem with this.

As the x86 operating systems and hardware systems became more powerful and robust, organizations wanted to start running more applications on a single host. However, some limitations in the development and runtime environments on these systems could potentially lead to application conflicts. To mitigate some of these potential issues, many of the independent software vendors wanted to support their applications on stand-alone systems, that is to say, on systems that were running only their application. Application development has improved to the point that it is possible to run many applications on the same system, and some organizations do that. But there may still be reasons for keeping the application systems separate. That is a completely different discussion that we will not go into here. Suffice it to say that there are millions of systems in use that run a single application.

When you have a powerful system and it is running a single application, the system is most likely not running at a very high capacity. In fact, many surveys have shown that it is very common for the majority of x86 architecture machines to be running at under 15 percent of the available capacity of that machine, some even under 5 percent. That is a lot of wasted potential. Combine that with the cost of housing these machines in expensive data centers and the cost of the energy needed to power and cool these environments, and it is easy to see why organizations are looking at ways to lower costs and make better use of the resources they already have.

So why not virtualize several of these individual hosts that are running single applications and put them on a single host that is capable of running virtual operating system environments? After all, if an organization can consolidate two servers to one server, they can save half the costs associated with hardware acquisition, energy consumption, and cooling. But why stop at 2:1 consolidation? In fact, many companies achieve consolidation ratios of 10:1, and even higher ratios are being looked at for the server environment. Ratios of 25:1 and higher are being attained when consolidating desktop systems onto servers. That is exactly why several software companies are providing software products that enable companies to virtualize physical x86 systems to consolidate their physical infrastructure.

Today's Virtualization Market

Connectix was founded in 1988 and was one of the early leaders of virtualization. It developed a Virtual PC product that enabled a Macintosh system to run a Windows desktop operating system. They then enhanced that with a product that was supported on the Windows desktop operating system so that one could run multiple Windows operating system environments on a Windows PC. Connectix was purchased by Microsoft in 2003, and Microsoft continued enhancing the Virtual PC product. Microsoft made several significant architectural changes to the Virtual PC product, releasing the Virtual Server 2005 product designed for running server operating systems. Most recently, Microsoft has added a Type-1 hypervisor product, Hyper-V, to its lineup. Types of hypervisors are explained in the section ''Virtualization Architectures.''

Another major player in this space is VMware, a company formed in 1998. They also started with a desktop-based product and developed server virtualization products. They became the market leader as they enhanced their emulation product, VMware Server, into a Type-1 hypervisor solution for the x86 architecture.

There are several other players in this market space with various implementations of hardware system virtualization technologies, such as Citrix with its Xen hypervisor and

various implementations of KVM from Red Hat, Novell, Sun, and Oracle. But the combination of Microsoft and VMware comprise over three-fourths of the market. Since Microsoft has entered the marketplace, it is looking more and more like the various server virtualization products will become a commodity market because companies like Microsoft and Citrix do not charge for their hypervisor products. This will force the virtualization vendors to differentiate themselves based on the added value they bring to their engines.

Microsoft's Server Virtualization Products

Microsoft offers three different products meant for virtualizing server operating system environments.

- Microsoft Virtual Server 2005 (hosted virtualization)
- Microsoft Windows Server 2008 Hyper-V (hypervisor)
- Microsoft Hyper-V Server 2008 (hypervisor)

Microsoft Virtual Server 2005

Microsoft Virtual Server 2005 is a hybrid or hosted virtualization product. It is written as a software service to emulate a specific hardware environment on which desktop or server operating system environments run. Virtual Server 2005 runs as a service on the Windows operating system, which in turn is running on the physical hardware. It was a groundbreaking release from Microsoft as it got them into the server operating system virtualization environment.

Virtual Server 2005 was Microsoft's first product as a virtual machine manager for the server operating system environments. They had obtained the Virtual PC product when they purchased Connectix in February of 2003. Virtual Server uses many of the components that Virtual PC has, except that Virtual Server was designed specifically to address the needs for virtualization as a service instead of as a desktop utility. Virtual PC and Virtual Server are good products that are still in use by thousands of companies around the globe. They work on both 32-bit and 64-bit versions of the Windows operating system prior to Windows 7 and Windows Server 2008 R2, but they support only 32-bit guest operating system environments.

Though the products are quite effective at what they do, there are some constraints in the software implementation. First, the execution environment is defined in software. This restricts operating systems and applications to the capabilities of the emulated hardware. For example, the software-emulated hardware is a motherboard that contains a single processor. That particular software implementation of a specific hardware constraint limits any virtual machine running on Virtual Server to a single processor.

Also, as new hardware is introduced and deployed, it may not be fully exploited. An example here is that the software-emulated network card is a DEC 21140, a four-port NIC. This was picked because it was a very common NIC, and nearly every x86 operating system has a driver for the DEC 21140. But those drivers did not support newer technologies that are being implemented in most organizations, such as VLAN tagging.

Lastly, the software-emulated motherboard was a 32-bit motherboard. In order to move into the 64-bit world, an entirely new software emulator would need to be written that would emulate an x86 64-bit motherboard. Rather than write an entirely new emulation package, which would still be limited due to the very nature of emulation, Microsoft decided to proceed down the proven path of the Type-1 hypervisor.

Microsoft Windows Server 2008 Hyper-V

Windows Server 2008 Hyper-V is what is sometimes known as a Type 1 hypervisor because it runs directly on the hosting hardware platform. This helps minimize the virtualization overhead. Though more efficient than a hybrid hypervisor, every form of virtualization has some level of overhead. It is much easier to minimize this in a Type 1 hypervisor because it executes on the hardware instead of being one level removed.

Hyper-V is a role of the Windows Server 2008 operating system. Microsoft has been driving toward making the operating system very modular with well-defined programming interfaces for interaction among the components. Windows Server 2008 is a major step in delivering on the goal to create roles in the operating system that can be added or removed or changed as needed with minimal to no effect on other roles. The release of Windows Server 2008 came in February of 2008 with 17 defined roles, one of which is Hyper-V. When Windows Server 2008 was released, Hyper-V was still in a beta state of development. In other words, features were mostly complete, but removal of bugs and additional performance work needed to be done on it. When these last tweaks were completed, it was released to the public in July of 2008 as an update available through Windows Update. This is a good example of Microsoft's goal of being able to change a role of the operating system without impacting the other roles.

Hyper-V can be installed as the only role on Windows Server 2008, or it can be installed in conjunction with any of the other 16 roles on Windows Server 2008. We will get into this in later chapters, but for production purposes, it is recommended that the Hyper-V role be the only role installed.

When the Hyper-V role is installed on Windows Server 2008, it inserts the hypervisor binary code between the Windows operating system and the physical hardware during the boot process. Installing the Hyper-V role also sets the `hypervisorimagelaunchtypeboot` Boot Configuration Database (BCD) setting to auto. If, for whatever reason, you want to disable the hypervisor, simply use the `bcdedit` command to change the parameter to disabled. Windows starts to boot as it normally does, but when the Hyper-V role is installed, the boot process inserts the `hvboot.sys` driver into the process. This launches the hypervisor while the rest of the Windows boot process continues.

Remember that if well-defined functions can be moved into hardware, performance can be improved. Both Intel and AMD have built technologies into their chips that make virtualization perform better. Hyper-V exploits some of these capabilities to ensure optimal performance. In fact, it requires that the 64-bit chip on which Hyper-V is installed have either the Intel-VT or AMD-V hardware-assisted virtualization technology. More information on this can be found in the section on hardware requirements.

Microsoft Hyper-V Server 2008

Though Windows Server 2008 Hyper-V can handle nearly any situation where x86/x64 virtualization is required, there are some instances where people do not necessarily want all the capabilities of having the hypervisor as a role of the operating system. That is one of the reasons why Microsoft released Microsoft Hyper-V Server in September of 2008.

Hyper-V Server is not a Windows Server operating system environment, but it is built on core Windows technologies. The easiest way to think of it is as a Windows Server Core installation with only the Hyper-V role installed. This is not entirely accurate because Hyper-V Server does not contain the full Windows Server operating system, but it is easier to comprehend if you think of it this way. To understand this, you need to know a bit about how the modularization of Windows is coming into play here.

During installation, you can choose between two Windows Server 2008 installation options. One option is what everyone is familiar with — the entire operating system with all its capabilities available. A second option is to select a Windows Server Core installation. The Core installation under Windows Server 2008 removes half of the available roles, and it removes the graphical user interface. All the core services like networking, security, file services, and RPC are still available.

Microsoft Hyper-V Server 2008 takes this a step further. It uses the Windows kernel and the Hyper-V role, but no other roles are there — the required components to support them are not even part of the distribution. Using the Windows kernel ensures that all the device drivers supported by the full Windows platform are available for use by Hyper-V Server. And, because the GUI is not available, all management of the host is done with remote management tools — the same remote management tools that are used for managing a Windows Server Core or Full installation. There is a command-line capability, and much can be done with these commands, but it is expected that this system will generally be managed with the remote graphical management tools.

The architecture of the virtualization stack in Windows Server 2008 Hyper-V and Microsoft Hyper-V Server 2008 is the same. It is simply packaged differently for different markets.

Virtualization Architectures

There are three primary forms of system virtualization. Figure 1.1 illustrates these three architectures.

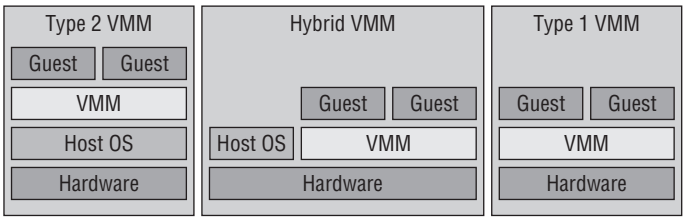


FIGURE 1.1 Virtualization architectures

The Type 2 architecture is implemented with things like Java Virtual Machines or Microsoft Common Language Runtime environment. It provides a runtime environment in which commands or processes can be run, regardless of the underlying operating system. There is a dependency on the virtual machine manager to understand the underlying operating system, but the language used to create the process has no dependencies on the operating system. Type 2 focuses on process virtualization, not server virtualization. Therefore, we will not spend any more time discussing this form of virtualization.

Hybrid and Type 1 architectures deal with hardware virtualization. In the hybrid (also sometimes called full or hosted) environment, the virtual machine manager runs alongside the operating system. It is installed as an application on the operating system and emulates an entire physical environment. This enables virtualizing the whole operating system environment in the virtual machine. Microsoft's Virtual PC and Virtual Server and VMware's Workstation and Server products are examples of hybrid architectures.

The Type 1 architecture is what we are considering with Hyper-V. In this case, the hypervisor runs directly on the hardware without the need for an intervening operating system. This is the most efficient way to virtualize an environment because it has the least amount of interference between the hardware and the guest machines.

Since hybrid and Type 1 hypervisors provide similar capabilities, one operating system environment running one or more additional operating system environments, we need to look a little bit at the different implementations. First, remember that we said IBM's VM/370 is a full virtualization product. This means that the entire hardware environment is emulated in software. Connectix and VMware built their initial products this way. Microsoft's Hyper-V is implemented as a Type 1 hypervisor, which is defined in the ''Hyper-V Architecture'' section later in this chapter.

From the brief history given above, it is easy to see that most of the historical virtualization has been done in the full or hosted virtualization method. That is the method where the entire hardware system is emulated in software. This method works and provides a great deal of flexibility in the operating system environments supported by providing an emulated hardware environment with devices that are supported by many operating systems. But that flexibility comes at the cost of performance. Every access to hardware is emulated in software, requiring the execution of many software instructions to emulate what would be normally handled by the hardware.

The Type 1 hypervisor improves that environment so that the operating systems executing in the partitions have more direct access to the physical resources of the host on which they are running.

Hyper-V Architecture

If you really want to understand how a product works, it is often helpful to understand the architecture of that product. And understanding the architecture is easier if regularly used terms are defined.

Hypervisor We like the definition of hypervisor that Microsoft uses in its device driver kit on MSDN (`http://msdn.microsoft.com/en-us/library/bb969710.aspx`).

> *''The hypervisor is a layer of software that runs above the physical hardware and below one or more operating systems. The hypervisor's main purpose is to provide isolated execution environments called partitions. The hypervisor provides each partition with the partition's own set of hardware resources (for example, memory, devices, and CPU cycles). The hypervisor must control and arbitrate access to the underlying hardware.*
>
> *''The hypervisor is a single binary that contains several components (for example, scheduler, partition management, and virtual processor management).''*

Guest The operating system environments that run in the partitions are referred to as guests. They are also often called virtual machines, or VMs. These terms are used interchangeably in this book. One of the goals of the hypervisor is to be agnostic to what sort of operating system environment is running in the guest.

Parent Partition In general all partitions created by the hypervisor are equal. However, you will see as we get deeper into the specifics of Hyper-V that the parent partition (sometimes called the root partition) acts as the owner of all the hardware resources. The ownership of physical memory and logical cores presents a special case. When a child partition is created, the parent partition allocates physical memory to the child and then the child partition manages it. Similarly, virtual cores are allocated to the child partitions and then scheduled by the operating system running in the child partitions. This is different from the architecture of VMware ESX, as in that architecture the hypervisor owns the hardware resources. This difference is

explained in more detail later in this chapter when the differences between monolithic and microkernelized hypervisors are discussed.

Because the parent partition in Hyper-V owns all the hardware resources, it also handles other system functions generally thought as being part of an operating system. These include things like booting the system, creating and managing other partitions, Plug and Play recognition, hardware error reporting, and so on. This is different from ESX, which handles all these functions in the hypervisor.

Hardware Virtualization Hardware virtualization is the act of providing multiple logical instances of physical hardware for use by the operating system environments running in the partitions. For example, on a system with only two cores, it may be possible to run three or four virtual machines, each with two cores.

Emulation Emulation is the process by which a virtualized device mimics a real physical hardware device so that guests can use the typical drivers for that hardware device. This means that a well-known hardware device, like the DEC 21140 network card, can use the device driver that is included in nearly every operating system. Emulated devices are less efficient than synthetic devices, but emulated devices provide support for operating systems that do not have integration components installed.

VMBus The VMBus is a high-speed memory bus that was developed specifically for Hyper-V. Any I/O traffic that passes to/from a child partition to the parent partition traverses the VMBus. This special kernel-mode driver is installed when the Hyper-V role is installed. Requests for access to physical devices, such as disks and network cards, are transmitted over the VMBus to achieve the highest possible performance.

Synthetic Device Synthetic devices are purely virtualized devices with no physical hardware counterparts. They function only in the context of virtual machines running under Hyper-V. Drivers for synthetic devices are included with the Integration Components for the guest operating system. The synthetic device drivers use the VMBus to communicate with the virtualized device software in the root partition.

Emulated or Legacy Device Hyper-V provides the ability to run operating systems that were written to run on physical hardware and have no knowledge of what virtualization is. This applies to older operating systems, such as Windows NT and Windows 98. These are known as legacy operating systems. Hyper-V provides emulated or legacy hardware devices. A given device's functions are emulated entirely in software in order that the legacy operating systems can access whatever the physical device is on the host computer. For example, the legacy NIC is a software-emulated DEC 21140 network interface card. By providing this legacy network interface, legacy operating system environments can still operate under Hyper-V even though the host environment might have a totally different physical device.

Integration Components Integration Components are a set of services and drivers that improve the integration and performance between the physical and virtual machines. These components enable the guest operating systems to use the higher-performing synthetic devices instead of emulated devices. This reduces the overhead required for the emulation of devices. Integration Components make use of the VMBus directly, thereby bypassing any emulation of a physical hardware device. Performance of synthetic devices with Integration Components approaches the performance of a physical device.

Integration Components provide the following capabilities to the supported operating systems:

- Synthetic devices (IDE, SCSI, NIC, video, mouse)
- OS shutdown
- Time synchronization
- Data exchange
- Heartbeat
- Volume Shadow Copy Services

Table 1.1 shows the operating systems with varying levels of Integration Component support. This can change, so it always makes sense to check sources at Microsoft. Microsoft publishes this information in the Hyper-V deployment document, but it is not in a tabular format. Be sure to refer to that document (`http://go.microsoft.com/fwlink/?LinkID=124368`) for the latest information.

Virtual Processors Each child partition has one or more virtual processors, sometimes called cores or logical processors, associated with it. A virtual processor is a virtualized instance of an x86 or x64 processor complete with user-level and system-level registers.

Hyper-V does not use hard processor affinities, so a virtual processor may move from one physical processor to another, depending on how the individual thread gets scheduled. Hyper-V schedules virtual processors according to specified scheduling policies and constraints to try to maintain locality for better performance, but there may be situations that move a virtual processor from one physical core to another.

Address Translation Any virtual memory system provides each application with a zero-based virtual address space. It then has a page table in memory that is used to map the virtual addresses to the physical addresses in the host.

A hypervisor introduces a second level of complexity into this. Because it allocates chunks of physical memory to each virtual machine, it needs to provide a physical memory virtualization facility to allow each partition to have a zero-based contiguous physical address space. Virtual processors support all the paging features and memory access modes that are supported in the physical environment so that the virtualized operating system runs the same in the virtual environment as it would in a physical environment.

To put this into practice, the hypervisor needs to implement two levels of address translation. The first level is what comes ''out of the box'' with the guest operating system environment. This is done via standard page tables maintained by the guest. Again, because we want the guest to run unmodified, this works exactly the same way as it would if the guest were installed on a physical host, except that the guest is writing to virtual memory instead of physical memory.

A second level of address translation is provided by the hypervisor without knowledge of the guest. This allows the hypervisor to virtualize physical memory, mapping guest virtual addresses to system physical addresses. The guest physical address space is defined at the time the partition is created.

TABLE 1.1: Hyper-V Integration Component support

SERVER OS	SYNTHETIC IDE	SYNTHETIC SCSI	SYNTHETIC NETWORK	SYNTHETIC VIDEO	SYNTHETIC MOUSE	OS SHUTDOWN	TIME SYNC	DATA EXCHANGE	HEARTBEAT	VSS SUPPORT
Windows Server 2008 R2 x64	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Windows Server 2008 x64	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Windows Server 2008 x86	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Windows Server 2003 SP2 x64	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Windows Server 2003 SP2 x86	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Windows 2000 Server SP4	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No VSS support

SUSE Linux Enterprise Server 10 x64	Yes	Yes	Yes	No	No	No	No	No	No	No VSS support
SUSE Linux Enterprise Server 10 x86	Yes	Yes	Yes	No	No	No	No	No	No	No VSS support
Client OS										
Windows 7 x64	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Windows 7 x86	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Windows Vista SP1 x64	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Windows Vista SP1 x86	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Windows XP SP2/SP3 x86	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No VSS support
Windows XP SP2 x64	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No VSS support

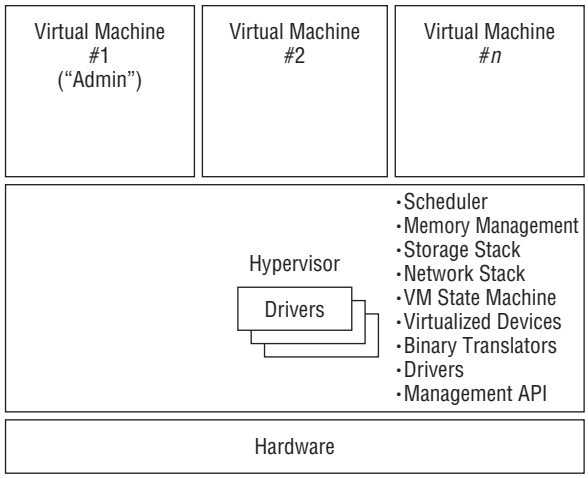
Monolithic versus Microkernelized

Monolithic and microkernelized are the two primary approaches to creating hypervisors. The difference is in the functions that are considered part of the hypervisor. A monolithic hypervisor contains many more functions than does a microkernelized hypervisor.

Monolithic

This implementation is less complex than writing a complete operating system because it is not likely to provide all the device drivers that a full operating system would have, but it is still a quite complex implementation. Figure 1.2 is a simplistic graphical representation of the monolithic architecture.

Figure 1.2
Monolithic hypervisor



With this architecture, notice that the various hardware device drivers are part of the hypervisor, as well as many other components. This is needed because the monolithic hypervisor is really an operating system. It handles all the functions that one would generally expect in an operating system, such as scheduling, memory management, file systems, the driver stacks for all supported hardware, management interfaces, and so on. What are missing that you would find in a general-purpose operating system are those components for handling regular program execution.

Microkernelized

Microsoft's Hyper-V uses what Microsoft calls a microkernelized hypervisor. The terminology comes from the work that Carnegie Mellon University scientists performed in the late '80s and early '90s when they developed the concept of an operating system kernel that contained just the bare minimum of functions that needed to be performed at the highest privileged execution mode of the hardware. They called their kernel the Mach kernel. Figure 1.3 shows a simplistic graphical representation of the microkernelized architecture.

Notice that in the microkernelized architecture, only the functions that are absolutely required to share the hardware among the virtual machines are contained in the hypervisor. The scheduler provides the shared access to the physical cores or CPUs of the hosting machine, and the memory manager guarantees that no two virtual machines try to access the same physical memory. The other required functions of an operating system are found in the parent

partition. With Windows Server 2008 Hyper-V, the parent partition runs the Windows Server 2008 operating system. This is how Microsoft gets Hyper-V to support all the systems and hardware devices that are supported by the Windows operating system.

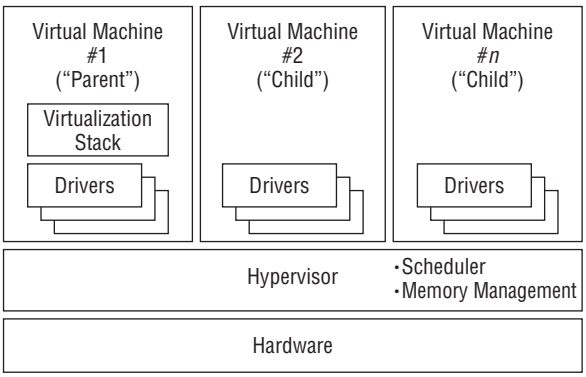


FIGURE 1.3
Microkernelized hypervisor

To understand why this is important, we need to look at how a typical operating system runs on x86 hardware. Execution of different functions is performed at different levels of the processor called rings, depending on how much access to the hardware is provided. The x86 hardware provides Rings 0–3. See Figure 1.4 for a diagram of the way Windows uses these rings.

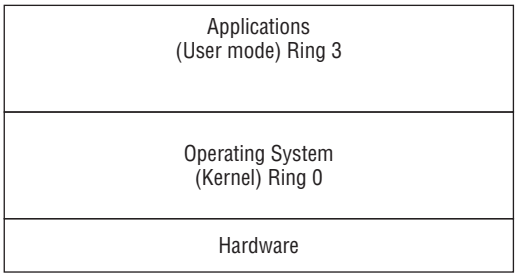


FIGURE 1.4
Security rings

Generally, the lower ring values designate a higher level of system access privileges. Operating systems running on the x86 hardware architecture have used Ring 0 to designate the kernel of the operating system — the portion that requires direct access to the hardware. User code, or applications, is designated as Ring 3. User code has to talk to the operating system to gain access to the physical hardware.

This is important from a security perspective because we do not want a general-purpose user to have direct access to a resource that is shared with other users. That creates a major security hole in a system and major potential for instability. It also means that code that runs in Ring 0 must be very careful in how it executes in order to ensure the stability of the system. Most of the time when a system crashes, it is due to a problem in a device driver. Device drivers run in Ring 0.

This works quite well when there is only a single operating system running on a system, as it can arbitrate requests from multiple applications. But in a virtual environment, there are likely to be many guest operating systems. There needs to be a way to maintain control of all platform resources in order to ensure smooth operations. In order to maintain this control

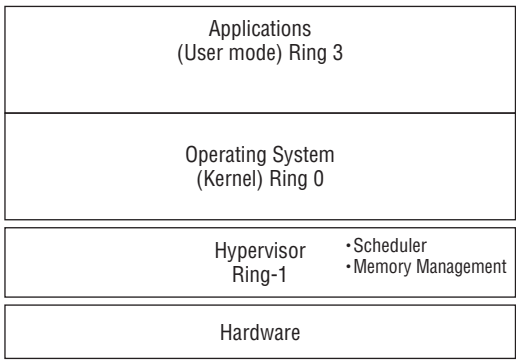
while multiple operating systems are running on a single physical server, the control should be placed at Ring 0, the highest privilege level on the CPU. However, all of today's operating systems are written to have their kernels run at Ring 0.

It is possible for software solutions to mitigate this. One method would be to insert a virtual machine manager that runs in Ring 0 to take control of the CPU whenever the guest operating system executes an instruction that could potentially cause conflict. The other method would be to modify the guest operating system prior to running it in a virtual environment. Both of these have been implemented, but each has shortcomings. Inserting a management component to trap instructions causes execution overhead. Modifying the operating system means you need to have a different version of the operating system for a virtual environment than is used in a physical environment.

AMD and Intel came to the rescue on this. They built hardware virtualization technology into their chips that allows specially written software to run at a level below the traditional Ring 0. Though the initial Intel architecture for the x86 family of processors did not have any levels beyond Rings 0 through 3, by placing certain components into some of the x86 64-bit family of processors, this critical control for sharing of resources can now be handled at a Ring-1 level. This ensures that the guest operating systems require no modification whatsoever in order to run, and they run very efficiently. This is known as the AMD-V and Intel-VT technologies.

Microsoft wrote its Hyper-V hypervisor to execute in this new ring. But since it executes at an even higher privilege level than the kernel of the operating system, Microsoft made sure that only those components that absolutely need to run at that level are running there. Figure 1.5 shows how this looks.

FIGURE 1.5
Security rings



This implementation requires that Hyper-V be very tight code. No third-party code exists in the hypervisor. The code has been very carefully inspected by Microsoft and by outside security firms that Microsoft has brought in to review the code. This is to ensure that the code is as secure as possible and has a very, very low probability of breaking.

This implementation leaves the third-party code that comes in the form of device drivers in the kernel of the parent partition. In other words, the device drivers that are used by a Hyper-V system are the exact same device drivers that are used by a Windows Server 2008 operating system. Hardware vendors do not have to write a different set of drivers for Hyper-V, and they continue to write device drivers using the same tools and operating environments they are all familiar with. With the monolithic hypervisor, either the owner of the hypervisor needs to write new device drivers or the kernel needs to be opened to device driver writers.

The microkernelized architecture helps achieve three goals:

Isolation and Security Hyper-V provides a high degree of isolation between partitions. Data that is in one partition cannot be viewed or manipulated by any other process except through well-defined communication paths, such as TCP/IP. In the next section on the parent partition, you will learn how the VMBus provides a path between the parent and child partitions. This is a special path that is used only for I/O to and from the physical devices owned by the parent partition.

Efficient Virtualization Hyper-V supports efficient virtualization of processors, memory, and devices. The ''virtualization tax,'' or overhead required to virtualize an operating system environment, is minimized.

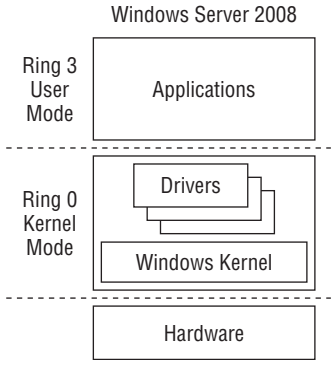
Scalability Hyper-V must provide scalability to large numbers of processors. The first release of Hyper-V supported 16 host cores. A patch increased that number to 24 host cores. Hyper-V R2 supports 64 host cores. As the number of cores in commodity servers increases, Microsoft will likely keep pace.

Parent Partition

Hyper-V provides isolation between the various instances of the operating systems by means of partitioning. A partition is a logical unit of isolation in which operating systems run. Hyper-V requires a parent, or root, partition running Windows Server 2008 x64 Edition. (As noted earlier, Microsoft Hyper-V Server 2008 is not a Windows Server, but it does run the Windows kernel.) The virtualization stack runs in the parent partition and has direct access to the hardware devices in order to share them among the child partitions. The root partition communicates with the hypervisor to create the child partitions that host the guest operating systems.

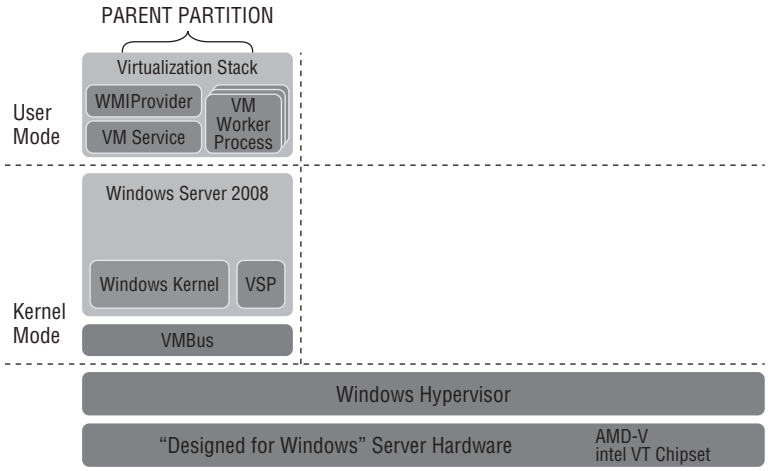
The parent partition is very important in the Hyper-V implementation. Until the Hyper-V role is installed on a system, there is no parent partition, and the Windows kernel operates directly on the hardware. See Figure 1.6 for a picture of what this looks like.

FIGURE 1.6
Operating system



Installing the Hyper-V role on Windows Server 2008 causes the hypervisor to be installed between the physical hardware and the Windows kernel at system boot time. This immediately changes the installation of Windows Server 2008 into this special parent partition. This partition still owns all the physical resources of the system, but it now needs to provide additional services to the other partitions, known as child partitions. See Figure 1.7 for a detailed diagram of the way this environment now looks.

FIGURE 1.7
Parent partition



You will notice that there are now quite a few new pieces in the picture. Let's again start at the bottom and work our way up.

VMBus The VMBus is a high-speed memory communication mechanism used for interpartition communication and device enumeration on systems with multiple active virtualized partitions. If the host is running only the parent partition, this is not used. But when other child partitions are running, this is the means of communication between those child partitions that have the Hyper-V Integration Services installed.

Child partitions do not have direct access to the physical resources of the host platform. They are presented with virtual views, sometimes called virtual or synthetic devices. Synthetic devices take advantage of special Integration Components for storage, networking, graphics, and input subsystems. Integration Component I/O is a specialized virtualization-aware implementation of high-level communication protocols (such as SCSI) that utilize the VMBus directly, bypassing any device emulation layer.

Operating systems running in the child partitions make requests to the virtual devices. These requests are redirected via the VMBus to the devices in the parent partition, which handles the actual requests. This makes the communication more efficient but requires a guest that is hypervisor and VMBus aware via the Integration Components. Hyper-V I/O and a hypervisor-aware kernel are provided via installation of Hyper-V Integration Services.

The VMBus provides the means of communication between the virtual service provider, or VSP, in the parent partition and the virtual service client, or VSC, in the child partitions.

VSP The virtual service provider provides support for the synthetic devices, instantiated as VSCs, that are in use in the child partitions. The VSP is how the child partition gains access to the physical device on the host system. A synthetic device is the way to provide a guest an abstracted version of a well-defined device interface that can be mapped via the VSP to the physical device that is on the host system. This is the mechanism in the microkernelized hypervisor that ensures support for the full range of hardware platforms available to Windows Server 2008 instead of limiting the number based on specific device drivers included in a monolithic hypervisor.

Windows Server 2008 Kernel The beauty of the microkernelized hypervisor is that the operating system kernel remains unchanged. The kernel, VMBus, device drivers, and VSP run in Ring 0 in the same manner on the hardware that the kernel and device drivers run in a nonvirtualized environment.

VM Worker Process A VM worker process is created for each child partition that is created. This process is used to provide virtual machine management services from the parent partition to the guest operating system running in a child partition. The Virtual Machine Management Service spawns a separate worker process for each running virtual machine.

VM Service The Virtual Machine Management Service is what creates the child partitions in which the guest operating system environments execute. It creates a separate process for each created child partition. Management tools communicate with this service to obtain information about the child processes.

WMI Provider The VM Service exposes a set of Windows Management Instrumentation (WMI) APIs for managing and controlling virtual machines. This provides a common management interface to both physical and virtual machines, meaning you do not need to learn a different set of tools in order to manage the virtual machines.

Child Partitions

Child partitions contain the various operating system environments. Hyper-V accommodates three different types of partitions:

- Windows partitions with Integration Components
- Linux partitions with Integration Components
- OS partitions without Integration Components

The use of Integration Components is an optimization technique for a guest operating system to make it aware of virtual machine environments and to tune its behavior when it is running in a virtual environment. Integration Components help to reduce the overhead of certain operating system functions such as memory management or disk access.

There are Integration Components for several components that can be installed into operating systems via a menu option or command line. These Integration Components do not require changes to the operating system source code. Here is a list of the various services provided by Integration Components:

- Synthetic devices (IDE, SCSI, NIC, video, mouse)
- OS shutdown
- Time synchronization
- Data exchange
- Heartbeat
- Volume Shadow Copy Services

Figure 1.8 shows what a child partition running a Windows operating system environment looks like when it has Integration Components installed.

As you can see, the child partition does not have the various service providers and management components that exist in the parent partition, but there are pieces that act in parallel with

components in the parent partition. Let's start at the bottom here and work our way up again, just as we did for the parent partition.

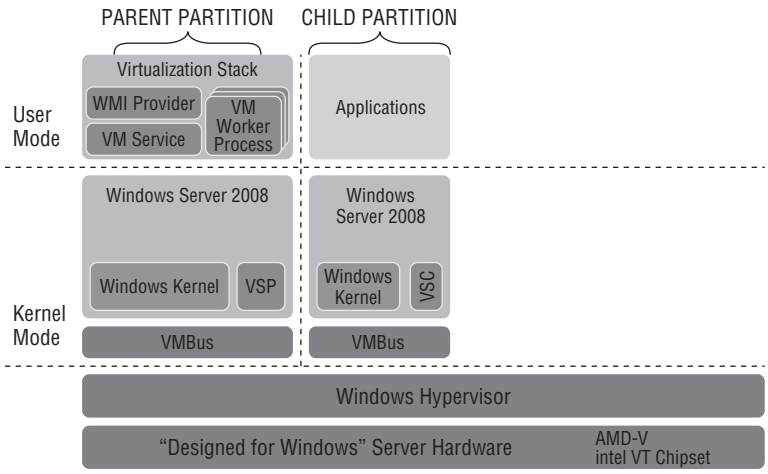


FIGURE 1.8
Integration Component Windows child partition

VMBus Any time Integration Components are used in an operating system environment, the VMBus will be used to provide the interpartition communication at a very high speed. The VSC communicates via the VMBus to talk to the VSP.

VSC The virtual service client is the synthetic device instance that resides in a child partition. Installation of Integration Services or Integration Components (Microsoft uses both terms in their documentation) installs the VSCs, which then communicate over the VMBus to the VSP in the parent partition.

Windows Kernel This is the off-the-shelf Windows kernel that is installed when the operating system is installed on physical hardware or in a virtual machine. No special code or special drivers are inserted into the Windows kernel to enable it to run in a virtual environment. The device drivers that are used are the device drivers that exist in the parent partition. The installation of the Integration Components installs the synthetic device drivers (VSC) for those operating systems for which Integration Components are available. Starting with Windows Server 2008 R2 and Windows 7, Microsoft includes the Integration Components in the distribution, so it is no longer necessary to install them separately.

Notice, too, that with the Hyper-V architecture, the kernel continues to run at Ring 0. Nothing is changed in the virtual environment from a security and isolation standpoint when compared to the physical environment.

Applications This is what you are really interested in. Sure, it is fun to virtualize an operating system, but operating systems are there to control the running of the applications. The applications are the useful parts of the computer environment. Applications run in user mode, or Ring 3, just as they do in a physical environment. Applications make the same sort of requests to the operating system as they would in a physical environment. In other words, the applications will generally not even know they are running in a virtual environment. The exception to this would be when applications make calls for specific devices that may not be supported, such as USB dongles or fax boards.

One of the primary reasons for the lack of support of some of these types of devices is that when Microsoft was architecting this environment, they were looking at it from the standpoint of what a server environment needs. When you look at specialized devices such as a dongle, those are primarily used on desktop applications. Other devices, such as fax boards, are not supported because they are installed on such a small percentage of servers.

Some devices, such as smart cards, may be able to be passed through to the virtual operating system environment by making use of a remote desktop connection using the remote desktop protocol. Microsoft has built into all its operating systems the ability to remotely manage via the remote desktop protocol. Built into the client that uses this is the ability to share local (desktop) devices with the host to which a connection is being made. So, if you have a requirement in your environment to use smart cards on your servers, you can share your smart card via the remote desktop connection.

If you have a need for a USB or serial device for your application, there are companies that market network adapters for these devices. One such company is FabulaTech (`www.fabulatech.com`). I have not used any of these devices myself, but I have talked with others who have had success using them with Hyper-V.

Now let's look at another type of child partition — the Xen-enabled Linux kernel. Xen is another hypervisor designed for the Linux community that is quite similar in architecture to Hyper-V. This Linux child partition running in Hyper-V is very similar to a Windows child partition with Integration Components.

Figure 1.9 shows an architectural view of a child partition running a Xen-enabled version of the Linux kernel.

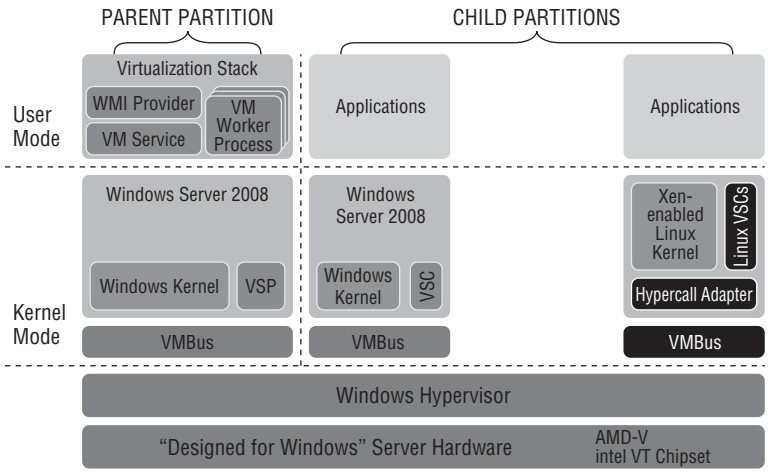


FIGURE 1.9 Xen-enabled Linux child partition

Notice that this is very similar to what a Windows child partition with Integration Components looks like. We have already covered the VMBus, so now we will examine the features that are unique to Xen-enabled partitions.

Hypercall Adapter Nearly every capability Microsoft puts into its operating systems has an application programming interface (API) for communicating with that service. The APIs for Hyper-V are referred to as *hypercalls*. These are native in the Windows Server 2008 environment. When they are ported to another environment, such as the Xen-enabled Linux kernel

environment, an equivalent package of tools needs to be built that talk to the defined interfaces of the API. The hypercall adapter is that section of code for use by our partners to enable non-Windows child partitions to access the hypervisor.

Linux VSCs Since the Xen architecture is quite similar to the Hyper-V architecture, it is possible to build a suite of VSCs for any distribution of Linux that has the kernel changes required to enable that distribution to run on the Xen hypervisor. These VSCs operate in exactly the same manner as the VSCs for the Windows integration components.

As of this writing, the Linux VSCs are available for SUSE Linux Enterprise Server 10 for both 32-bit and 64-bit. Work is under way on the Red Hat distribution. More changes are likely to occur as time goes on, so be sure to check `www.microsoft.com/windowsserver2008/en/us/hyperv-supported-guest-os.aspx` for the latest information about which distributions are supported.

Microsoft recently submitted code to the Linux kernel under GPL V2. This code provides the Integration Components with the synthetic drivers for upcoming versions of the Linux kernel. When incorporated into the kernel of any Linux distribution, this will provide VSCs for that Linux distribution.

Okay, now we will take a look at the last type of child partition. Figure 1.10 shows the structure of an operating system that has no knowledge of virtualization. These operating systems are sometimes called legacy operating systems, particularly when talking about older versions of Windows such as Windows NT Server or Windows 98.

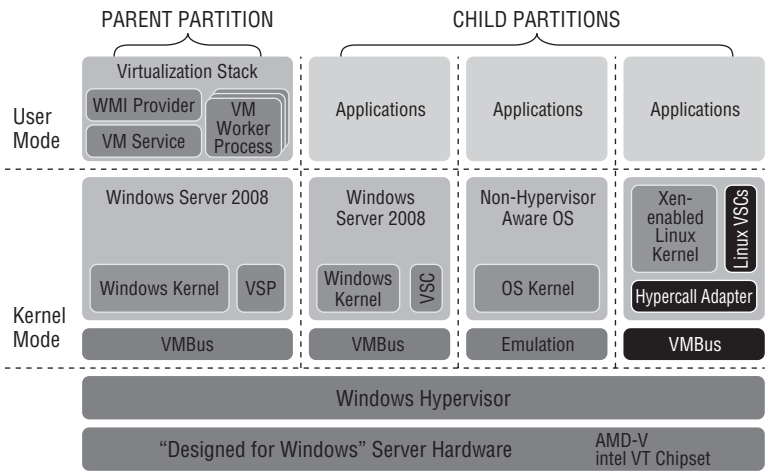


FIGURE 1.10 OS child partition without Integration Components

Child partitions without integration components do not have access to the VMBus or VSCs. They are called legacy because they do not have support for the synthetic devices to improve performance or minimize the virtualization tax. Hardware components are emulated in software, like the hybrid hypervisor explained earlier. The emulation layer provides the translation from the device drivers in the operating system to the parent partition. As noted earlier, software emulation of hardware devices is expensive in terms of performance. What may take a few instructions in hardware may take thousands of instructions in software. As a result, you

will likely see systems operating at a lower level of performance than their brothers and sisters running with Integration Components.

Hyper-V Technical Specifications

One of the beauties of making Hyper-V a role of the Windows Server 2008 operating system is that it can make use of all the hardware and software innovations that are continually being developed for the host platform. To start with, Hyper-V is a role in Windows Server 2008 Standard Edition, Enterprise Edition, and Datacenter Edition. Depending on the specific need, the proper edition of the operating system can be deployed and still be used to provide virtualization capabilities.

Let's look at the specifics of what is required to run Hyper-V. Then we will look at the capabilities that are available.

Hardware Requirements

To install and use the Hyper-V role, you will need the following:

- A Designed for Windows x64–based processor. Hyper-V is available in x64–based versions of Windows Server 2008 — specifically, the x64–based versions of Windows Server 2008 Standard, Windows Server 2008 Enterprise, and Windows Server 2008 Datacenter Editions.
- Hardware-assisted virtualization. This is available in processors that include a virtualization option; specifically, Intel VT or AMD Virtualization (AMD-V). Hardware virtualization is defined in the BIOS and needs to be enabled, as it is typically off by default.
- Hardware Data Execution Prevention (DEP) must be available and be enabled. Specifically, you must enable Intel XD bit (execute disable bit) or AMD NX bit (no execute bit). As with hardware virtualization, DEP is enabled in the BIOS.
- Enough memory (RAM) to run the Windows kernel (at least 1 GB, but 2 GB is better) and to run one or more virtual machines.

Designed for Windows

Hyper-V requires an x64 system, a 64-bit hardware platform that is built with the x86 instruction set. Supported systems are limited to those systems that carry the Designed for Windows logo in the Windows catalog. Go to `www.windowsservercatalog.com` and select the server category. From the Server page, you will find a category specifically for Hyper-V–capable systems. This gives the customer the broadest possible selection of supported host systems. This also ensures that any storage and network card that are supported on those host systems can be used by the virtual machines. This is significantly different from a software-emulated environment, which limits the execution environment to the hardware that is emulated by software. It is also different from some other virtualization vendors that limit the selection of host, storage, and network to those particular systems and devices for which they have written their own device drivers.

Hardware Virtualization

Both the major x86/x64 chip vendors, Intel and AMD, have worked to include features in their chips to provide hardware-assisted virtualization. These features offload to hardware some of the things that would have been done in software to streamline execution and boost performance. Intel's implementation is called Intel-VT (for Virtualization Technology); see

www.intel.com/technology/virtualization. AMD's implementation is called AMD-V (for Virtualization); see www.amd.com/us-en/0,,3715_15781_15785,00.html.

Hyper-V requires either an Intel-VT or AMD-V chip. Several years ago Intel and HP developed another 64-bit chip called Itanium, and its architecture is often referred to as IA64. Hyper-V is not supported on IA64 platforms. It is only for the x64 platform.

Data Execution Prevention

Microsoft is very serious about building security into its products. Intel and AMD are also concerned about building whatever capabilities they can into their chips to enhance security. Both chip vendors have implemented a feature called Data Execution Prevention. In a nutshell, this feature prevents the execution of instructions in areas of a program that is defined as containing only data. This is a common way that hackers can cause problems in code with things such as buffer overflows. Hyper-V requires either Intel-XD or AMD-NX enabled on the chip.

Real World Scenario

Sherman, Set the WABAC Machine to 1973

When I was an assembly code developer, which was well before the days of the Internet and script-kiddies, I would regularly define a block of data as my patch area. Because it took so long to compile code back then, it was generally much quicker to find the offending instruction, change it to a branch instruction to branch to the patch area, put in the corrected code, and then branch back to the regular code by using binary patch methods. (Believe it or not, this was even sometimes done with punch cards when punch cards were used to start and control execution of the program.) In fact, this was such a common procedure that COBOL even had an ALTER statement that could be built into the code and could be executed if certain conditions were met. Hackers learned to quickly exploit the ability to execute instructions that existed in areas defined as data. A prime example of this sort of exploit is called buffer overrun.

Early exploits, and ease of doing this, showed the need for a stricter definition of what was data and what was code in a program. As that got better defined, it became easier to implement capabilities in the hardware to enforce the data execution protection. Therefore, Intel implements what it calls XD (for execute disable). AMD implements what it calls NX (for no execute).

Hyper-V Capabilities

There are many ways to define features or capabilities. Many of the other chapters in this book about Hyper-V will explain various operational features and capabilities and how to use them. In this section, we will look at the technical features, generally thought of as how one can configure virtual machines. First, we will look at what sort of system is supported for the host. Then we will look at the options available for configuring virtual machines.

Host System

Microsoft has two different ways of implementing the Hyper-V hypervisor. There are slightly different hardware requirements for the host systems.

Windows Server 2008 Hyper-V

- Designed for Windows x64 host
  - AMD-V or Intel-VT hardware virtualization in the chip
  - Data Execution Prevention
- Recommended processor speed: 2 GHz or faster
- Windows Server 2008 Standard, Enterprise, or Datacenter Edition
- Up to 16 processors or cores
  - KB956710 increases this to 24 cores only for those systems that have Intel's 6-core chip.
  - The maximum number of cores supported by Hyper-V is the same for each edition of Windows Server 2008, but the maximum number of physical processors varies according to edition:
    - Standard Edition — up to 4 processors
    - Enterprise Edition — up to 8 processors
    - Datacenter Edition — up to 64 processors
  - Windows Server 2008 Hyper-V R2 increases this number to 64 cores.
- Up to 1 TB memory for Windows Server 2008 Enterprise and Datacenter Editions. Standard Edition supports a maximum of 32 GB of memory.
- At least one network interface card (NIC)
  - Recommend one NIC for host server management
  - Recommend at least one NIC for virtual machine communication
  - Recommend at least one NIC dedicated to iSCSI if using iSCSI storage
  - Recommend at least one NIC for cluster communication if the systems are configured in a failover cluster
- Storage to host the parent partition, virtual machines, and associated data
  - Direct-attached storage: Serial Advanced Technology Attachment (SATA), external Serial Advanced Technology Attachment (eSATA), Parallel Advanced Technology Attachment (PATA), Serial Attached SCSI (SAS), SCSI, USB, and IEEE 1394 (sometimes known as FireWire)
  - Storage area networks (SANs): Internet SCSI (iSCSI), Fibre Channel, and SAS technologies

Microsoft Hyper-V Server 2008

The hardware configuration options for a Hyper-V Server host are very similar to those of Windows Server 2008 Hyper-V in that the host supports the Windows-certified hardware components. Since this is designed for a more restrictive set of uses, there are some limitations when compared to Windows Server 2008 Hyper-V.

- A host with up to 32 GB of physical memory
- A host with up to four physical processors (maximum of 24 logical processors)
- Guest with a maximum of 31 GB of memory

Because this is not a Windows operating system (it is just the Windows kernel), it does not support features of the operating system such as failover clustering. In order to ensure that it fits within a corporate environment, management agents such as antivirus and System Center Operations Manager can be installed. The hardware limitations are the same limitations that exist for Windows Server 2008 Standard Edition.

Microsoft Hyper-V Server 2008 R2 significantly increases the capabilities.

- A host with up to 1 TB of physical memory
- A host with up to eight physical processors (maximum of 64 logical processors)
- Guests with a maximum of 64 GB of memory
- Support for Failover Clusters and Live Migration

Supported Virtual Machines

A point that often confuses people is the word *support*. The term *supported* can easily be misconstrued. Microsoft defines *supported* to mean that it either owns the code or has a support agreement with another organization that does have access to the code so that errors can be fixed at the source level. For example, when Microsoft says that it does not support other operating systems, it means that it does not have a way to gain access to the code in order to change it. That becomes pretty obvious when you consider any product that is not owned by Microsoft. Since Microsoft does not own the code, it cannot make changes to that code if a problem occurs. This does *not* imply that simply because it is not supported, it does not work. So, the list of supported operating systems that follows includes those operating system environments to which Microsoft can make changes or have changes made.

- Guest operating system support
  - Windows Server 2008 and 2008 R2 x64/x86
  - Windows Server 2003 SP2 x64/x86
  - Windows Server 2000 SP4 x86
  - Windows Web Server 2008 and 2008 R2 x64/x86
  - Windows HPC Server 2008 and 2008 R2 x64
  - SUSE Linux Enterprise Server 10 and 11 x64/x86
  - Red Hat Enterprise Linus 5.2 and 5.3 x64/x86
  - Windows 7 x64/x86
  - Windows Vista SP1 x64/x86 (not Home editions)
  - Windows XP Professional SP2/SP3 x64/x86

    - Always check `www.microsoft.com/windowsserver2008/en/us/hyperv-supported-guest-os.aspx` for the latest details. This list will change as more operating system environments are tested and added.
- Nonsupported operating system environments
    - Hyper-V has the ability to run other x64 and x86 operating system environments. Though some of these other operating system environments may run fine, Microsoft has not done any testing on these environments, so they do not fall under the support statement defined above.
- 1-, 2-, or 4-processor virtual machine
- Up to 64 GB of memory for each guest, depending on OS
- Up to 12 network interface cards (NIC)
    - Up to 8 synthetic
    - Up to 4 legacy
    - Static or dynamic MAC addresses
    - VLAN support
- Disks
    - IDE
    - SCSI
    - iSCSI
    - Virtual hard drives (VHD) up to 2040 GB in size
    - Pass-through disks (direct connection to disk connected to the host) — size limited by operating system. Windows currently has a limit of a 256 TB volume.
- USB devices
    - Mouse
    - Keyboard
    - Disks and NICs connected to the host (more detail on this in the chapter on storage and networking)
    - Hyper-V does not provide direct access to any USB devices, other than the keyboard and mouse. Some nonsupported USB devices, such as a smart card reader, can be used in a virtual machine by using the USB device-recognition capabilities of the remote desktop.
- Documented Windows Management Instrumentation (WMI) interfaces for scripting and management. For more information on the WMI interface, see `http://go.microsoft.com/fwlink/?LinkID=108564`.
- Snapshots of at-rest or running virtual machines. Up to 50 snapshots per VM can be taken.

- Virtual CD/DVD drive
  - A virtual machine has one virtual CD/DVD by default.
  - Virtual machines can be configured with up to three CD/DVD drives, connected to IDE controllers. (Virtual machines support up to four IDE devices, but one device must be the startup disk.)
  - A virtual CD/DVD drive can access CD or DVD physical media or ISO files. However, only one virtual machine can be configured to access the physical CD/DVD drive of the host system at a time.
- Virtual COM port — Each virtual machine is configured with two virtual serial (COM) ports that can be attached to a named pipe to communicate with a local or remote physical computer. This is software-only access — no access to the physical COM ports on the host.
- Virtual floppy drive — this is a software-only floppy drive used for accessing `.vfd` (virtual floppy drive) files.

The capabilities shown above are the basic capabilities that come with the Hyper-V. In subsequent chapters you will learn about more advanced capabilities such as Live Migration, snapshots, Cluster Shared Volumes, backup of running virtual machines, built-in high availability, hot add/remove of storage, and more. As you learn about these capabilities, think of how they can be applied in your particular environment to help you implement a dynamic virtual environment.

The Bottom Line

Microsoft's history in virtualization Microsoft has introduced multiple products for virtualizing the x86 architecture.

> **Master It** List the three products that Microsoft has released that provide the capability to virtualize the x86 or x64 architecture.

Monolithic versus microkernelized virtualization architectures Monolithic and microkernelized are the two primary approaches to creating hypervisors. The difference is the amount of code between the virtual machine and the physical hardware on which it is running.

> **Master It** List the components that exist in the monolithic hypervisor. List the components that exist in the microkernelized hypervisor.

Hardware requirements Not all systems are capable of running Hyper-V.

> **Master It** List the minimum requirements of the host system for installing and running Hyper-V.