Chapter 1

Windows Server 2008 Architecture

Any server operating system version upgrade involves changes to nearly every subsystem, and Windows Server 2008 is no exception. There have been kernel changes to allow for better processor virtualization, driver model changes to make drivers more stable and secure, a completely new TCP/IP protocol stack for better performance, a new graphics engine, and very significant changes to the Windows Class Libraries that give access to the Windows .NET Framework and Network Class Library. To a user learning Windows Server 2008, these changes underpin all of the new procedures that differentiate Windows Server 2008 and Vista Service Pack 1 (SP1) from the previous versions of Windows Server 2003 and Windows XP. So in order to provide a framework for our discussions in future chapters, this first chapter explains the Windows Server 2008 architectural model, along with the changes that they portend.

There have been enormous changes in the Windows Server architecture for this release of Microsoft's flagship product. As Microsoft's desktop OS and Microsoft Office slow down in revenue growth, Windows Server has become particularly important to Microsoft's fortunes. Microsoft Windows Server 2008 continues Microsoft's push to integrate web-based services into its server product. There are now strong links to the .NET Framework, both programmatically as well as in the look and feel of applications that will take advantage of the architecture that is described in this chapter. While Microsoft emphasizes the new modules that affect what the user can see, there are plenty of architecture changes to basic systems such as the kernel, memory, services, and many more. In this chapter, you will read about the most significant architectural goals of this release and how you can get even more value out of your investment in Windows Server.

Understanding the System's Roots

Writing an operating system for a new computer system has always been something of a heroic effort. In 1981 Tracy Kidder won a Pulitzer Prize and an American Book Award for his book *The Soul of a New Machine*, in which he describes the creation of the 32-bit Eclipse MV/8000 minicomputer at Data General. The company Data General was competing with at the time was Digital Equipment Corporation (DEC), their computers were the PDP-11 series, and that operating system was VAX (short for Virtual Address Extension).

The early history of Windows NT as documented in G. Pascal Zachary's *Showstopper! The Race to Create Windows NT and the Next Generation at Microsoft* (1994) was similarly chaotic. NT started as a co-development project between IBM and Microsoft; it was to be version 3.0 of OS/2, both companies having worked on versions 1 and 2. Windows 3.0's success led Microsoft to develop OS/2 API into an extended Windows API, at which point the joint project fell apart. IBM would go on to release OS/2 3.0 as their "Warp" version, which although something of a success in the business world never caught on with the general public. Microsoft chose another direction.

When DEC canceled the PRISM project in 1988, Microsoft was able to hire away many of the team members working in DEC West in Seattle. Systems architect David Cutler, who had experience with Virtual Memory System (VMS) and Reduced Instruction Set Computer (RISC) architecture, became the head of the development team that rewrote the Microsoft version of OS/2. The concepts of a microkernel, protective mode, an executive program, device drivers, and hardware abstraction all arise from the work of Cutler's group and the heritage that they brought to Microsoft. The basic idea behind Windows Server was that Microsoft (being a software company) should create an operating system that was portable from one hardware platform to another with a straightforward recompile.

The original NT project targeted development of IA-32 (Intel Architecture), MIPS (Microprocessor without Interlocked Pipeline Stages), Alpha, PowerPC, SPARC, and Intel i860 and i960. Microsoft released the first version of NT on July 27, 1993. Indeed, the multiplatform heritage of Windows Server is supposedly the origin of the NT name, as initial development of OS/2 3.0 was for the i860 processor code named N10, or "N-Ten." It is speculated, but not confirmed, that Microsoft replaced the NT name (which they marketed under "New Technology") with the year designations because of a trademark issue with Nortel Networks, whose Northern Telecom owns the NT trademark.

Today the Windows operating systems in general, and the Windows Server systems specifically, owe much of their success and many of their shortcomings to the design decisions made by the original NT team for Windows NT. There isn't space here to expound on why many of the technical features in Windows are a little goofy (you can take the Windows Registry as an example), but any reader interested in software history from a technical viewpoint should take a look at Raymond Chen's *The Old New Thing: Practical Development Throughout the Evolution of Windows* (2006).

System Overview

Windows Server is an example of a preemptive, multitasked, multithreaded operating system. The term *preemptive* means that the Windows operating system can switch the order of tasks held in memory based on a set of scheduling policies and priorities, a process commonly referred to as context switching. Windows can perform a context switch because it recognizes that a process has ended, that a process is blocked by a shared resource, or that the process has been switched to another processor. This behavior (which is a software version of a system interrupt) is performed by the kernel, and is different from a system in which tasks are scheduled through time sharing or through a system of hardware interrupts. There are many advantages to preemption, but the most important are that preemption uses resources more efficiently and that it can remove many instances of processes that are either CPU bound or input/output (I/O) bound, waiting for other processes to finish.

A process or task is the execution of program or service. You can see processes executing in Windows Server 2008 or Vista in the Task Manager, where the consumption of memory and CPU is given for each. Figure 1.1 shows the Process page of the Task Manager.

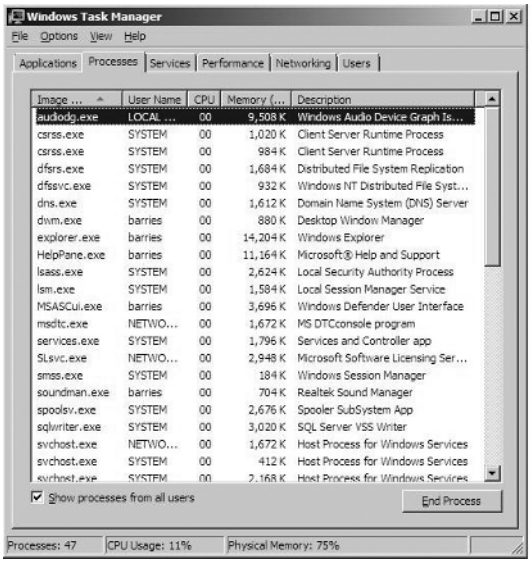
Although the Task Manager got an upgrade in Vista so that it shows more information and actually describes what each process is, it does not offer the most complete description you can get on your system. That honor belongs to the service- and process-related commands you can enter into PowerShell, which is discussed in Chapter 8, "PowerShell."

You can get to the Task Manager by doing any of the following:

- Right click the Taskbar and select the Task Manager from the context menu.
- Click an empty area of the desktop and press Ctrl+Shift+Esc.
- Click the Start menu, enter **TASKMGR** into the Search text box, then press the Enter key.

- Press Windows+R to open the Run dialog box, then enter **TASKMGR** and press Enter.
- Press Ctrl+Alt+Del and select Start Task Manager from the menu on the Login/Logoff screen.

FIGURE 1.1
The Task Manager has been upgraded to give more detailed information.



Windows gives you so many different ways to get to the Task Manager because the Task Manager allows you to kill errant processes and bring a server back to life. The information that the Task Manager presents to you is one of the best tools you have in improving the performance of Windows Server.

The Windows operating system examines processes, and decomposes them into the fundamental units of execution, which are called threads. Threads are assigned priorities and then added to the scheduling of the execution queue. Through context switching, all processes get access to the processor according to their priority, something that is often referred to as time slicing. When a process allows it, Windows can decompose the process into multiple threads, optimizing the processing and improving the process's performance (known as multithreading). Windows is capable of managing the event queue for a single process or to multiple processors through a Symmetric Multi-Processor (SMP) scheme. With time slicing, it appears that the system is doing multiple tasks at the same time, or multitasking, but that is only the net result of individual steps that are executing too quickly to be differentiated. The effect a user sees is akin to viewing a movie.

Windows 2000/NT Architecture

In its original conception, Windows NT was built as a modular system with two distinct layers. What the user sees and works with was called the user mode, and all hardware I/O was contained in modules that are part of the kernel mode. The kernel mode isolates programmatic control of hardware from the user mode, thus making the system both more stable and more secure. These two different layers were designed to execute in separate memory addresses, further isolating the User mode from direct hardware control. Figure 1.2 shows a schematic of the Windows 2000 operating system.

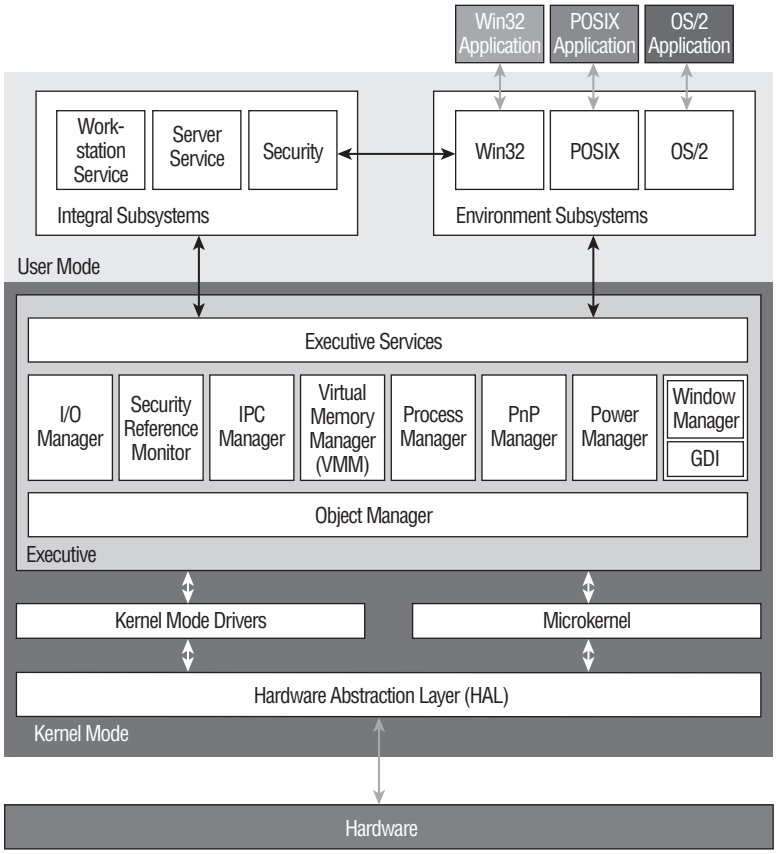


FIGURE 1.2
The heritage of the Windows operating system architecture is evident in this diagram of Windows 2000/XP.

The kernel mode contains several major subsystems that are fundamental to its operation. The Hardware Abstraction Layer (HAL) is software that is specifically compiled for the hardware platform that Windows runs on, and is designed to make the operating system portable. When Windows is ported to a different type of computer, the HAL can be recompiled for that platform's instruction set. The HAL thus preserves the code, command set, and APIs of the remaining kernel mode components. The HAL's hardware code controls I/O interfaces, interrupts, and processor access.

The microkernel contains a minimal operating system: upon startup, it loads essential services for address space management, thread management services, and Inter-Process Communication (IPC). IPC is a set of methods for exchanging data between threads in one or more processes. In an operating system with a true microkernel, the size of that microkernel is very small. Real-time operating systems (RTOSs) that are used in embedded systems, such as vending machines or in cars, contain a microkernel. The Windows operating system is more appropriately referred to as a hybrid kernel.

Executive Services communicate with the user mode subsystems, converting program commands into object code in the Object Manager. Executive Services is where you find low-level security routines, interface and process management, graphics display routines, power management, and other vital services that make Windows Windows. These subsystems are organized into a set

of "Managers," examples of which are the I/O Manager, Memory Manager, Cache Manager, Object Manager, and so forth.

In Windows 2000 (and Windows NT before that), the user mode contained two subsystems: those with integral services and an environmental subsystem. The environmental subsystem is a set of operating system routines that allow Windows to run different operating systems. The Win32 subsystem can run 32-bit Windows applications, and through Virtual DOS Machines (VDMs), it could also run 16-bit applications such as MS-DOS and Windows 3.1 in emulation. Environmental modules are extensible, as you can see with the module that supports OS/2 and another that supports the POSIX standard of Unix.

The integral subsystem in the user mode provides services to the environmental subsystem. High-level security such as resource access and the Active Directory is implemented in this security subsystem. The other two subsystems are networking services. Workstation is an API that communicates with the network redirector and communicates with other systems. Server is the API to a service that allows that local computer to provide services to other systems on the network.

With this brief overview of the Windows 2000 architecture, let's turn our attention to the changes that Windows Server 2008 and Vista SP1 have ushered in.

Architecture for Windows Server 2008

Microsoft first unveiled Windows Server 2008's architecture at the Microsoft Professional Developers Conference in 2003. Figure 1.3 shows the original system block diagram that detailed the major projects under the Windows Server 2008 umbrella.

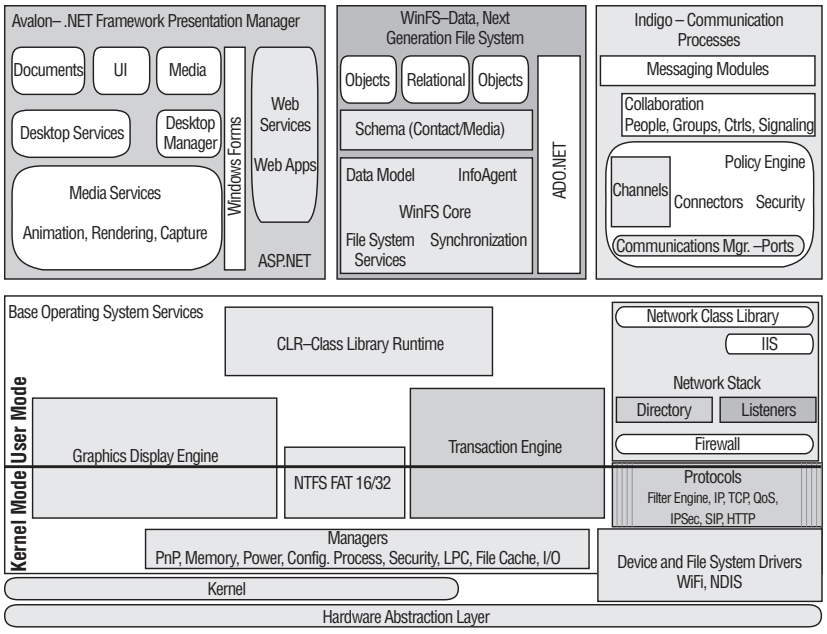


FIGURE 1.3 Microsoft's original plans for their Windows Server 2008 project included reworking all of the higher-level system functions.

For an image map with a description of each of these separate modules, go to

`http://www.ftponline.com/resources/longhorn/baseos.asp`

Nearly all of the user mode modules were to be replaced in the new operating system. Among the changes were:

Avalon Avalon was Microsoft's new presentation layer and next-generation user interface. Avalon became the Windows Presentation Foundation (WPF), and is the graphical front end to Microsoft .NET Framework 3.0 (once called WinFX).

Avalon introduces multiple new graphics standards. The API supported Extensible Application Markup Language (XAML) vector graphics (a form of XML), as well as extending 3D graphics with Direct3D. With the introduction of Direct3D version 10, Microsoft has a new video standard for rendering that introduces advanced shading and rendering technologies. Windows Server 2008 dropped support for the competing industry standard OpenGL, which now requires third-party support. Microsoft Silverlight is an XAML/JavaScript implementation of WPF that enables Flash-type vector animation based on .NET programming code.

Aero Aero was the user experience module that brought the 3D graphics envisioned in the Windows Palladium project to commercialization. Aero overlies the presentation module. While Aero is much more important in the Vista Client, you can enable Aero on Windows Server 2008 as well.

Indigo Indigo was the first network communication API. Indigo became the Windows Communication Foundation (WCF), and is a message service that allows programs to operate either locally or remotely. With WCF a program written in any language can be targeted to operate in a .NET virtual machine at runtime, and can run both locally and remotely as needed. WCF creates a service, a host environment to contain the service, a contract that defines the service methods, and the endpoints to which clients can communicate. It's rather similar in concept to how web services are supposed to operate.

WinFS Windows Future Storage (WinFS) is a new relational database storage subsystem containing the next version of the Windows file system. WinFS didn't make the first release of Windows Server 2008, although some of the technologies were to be integrated into the next generation of Microsoft SQL Server (Katmai) and ADO.NET Entity Framework. According to Steve Balmer, CEO at Microsoft, this project is still alive and will be released at some unspecified later date.

CLR Common Language Runtime (CLR) is the virtual machine module of Microsoft .NET. CLR is the Microsoft implementation of the Common Language Infrastructure (CLI) standard, which allows code written in high-level languages such as C# or Visual Basic .NET to be compiled using the Microsoft Intermediate Language (MSIL) as needed for runtime. MSIL is Microsoft's implementation of the Common Intermediate Language.

Network Class Libraries Network Class Libraries are a set of objects that provide simple access to network services. You can set object properties, events, and methods, and use these objects in code to access the features of network protocols. The work in new network services in Windows Server 2008 has resulted in an entirely new TCP/IP stack (among other changes) for both Vista and Windows Server 2008.

WinFS and the Road beyond Cairo

WinFS is the latest incarnation of the Object File System that Microsoft has been talking about for nearly eight years now. So its failure to be included in Windows Server 2008 is certainly a major disappointment to many of us who follow the Windows platform professionally. WinFS, as you can clearly see in Figure 1.3, was at the center of Microsoft's plans for their new operating system. You can take WinFS's deletion as a measure of how extraordinarily difficult it is to switch the hier-

archical file structure found in FAT and NTFS to an object-oriented relational database structure, where data is interwoven by tags and metadata.

Step back in time (if you would) to 1998 when the concept of an object-oriented file system was part of the Cairo project (circa 1991–1995). Cairo was the Twilight Zone of Windows Server, and the road to Cairo had many potholes. Cairo was abandoned, the object-oriented file system was put on hold, and a less ambitious project called Windows NT 5.0 came into being. According to legend, Windows NT 5.0 was the last major version of the Windows operating system that didn't bear a code name. NT 5.0 was branded as Windows 2000 Server.

The object-oriented file system was also a natural tool for the Component Object Model (COM) (begun in 1993), as COM supports object-oriented distributed IPC. A new file system called Storage+ was described (but never shipped) that supported COM. COM is the direct ancestor of a slew of technologies, including Object Linking and Embedding (OLE), OLE Automation, ActiveX, COM+, and Distributed COM. COM laid the groundwork for the current era's technologies: Microsoft .NET Framework and Web Services supported by the Windows Communication Foundation (WCF).

Given this long history of development, it's worth noting exactly why this technology has garnered so much interest at Microsoft. As it stands now, data stored by applications running on Windows can be "rich" in the sense that they contain complex data, but they aren't easily integrated with the data from other applications. This is a long-standing computer industry problem. A hospital might have a database that stores text fields, memo fields (large text fields), number fields, picture fields, even things commonly called BLOBs (Binary Large Objects) in a unified data store. That database might know that one field stores phone numbers, another field stores the X-ray images of its patients, and so on, but another application wouldn't be able to have knowledge of this metadata without a lot of additional programming.

Here's where the object-oriented file system comes into play. Suppose the hospital wanted to do a search for people of a certain age, who lived in a certain location, and whose X-ray indicated that they had a condition of concern in common so that they could contact those people with a letter. You can certainly do many, if not most of those things, in a database, but you couldn't do all of that from your operating system because the data is stored in proprietary or different formats, and stored without the metadata necessary to tie it all together. But if the file system could do this, you would achieve the following:

- Data independence of any application
- Integrated storage with a single data model
- Single instance storage where only one instance of the same object is stored
- Advanced search and data collection
- Expert systems and advanced data mining

So it's clear why Microsoft continues to develop technology in this area, and why what they learn becomes part of every major new data store such as Exchange, SQL Server, or Active Directory, that we see—just not, for the time being, the Windows Server file system.

The Child of .NET

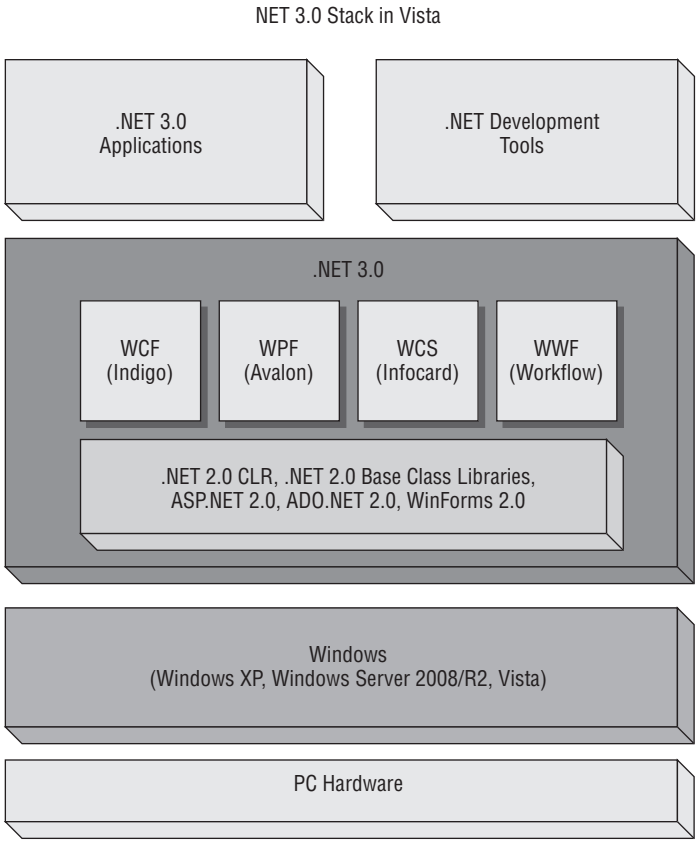
Every version of the Windows operating system has at least one major initiative that it becomes known for. With Windows 95 and Windows NT 3.5 and 4.0, the major advances were made in networking and internetworking. Windows 95 gave us Internet Explorer integrated into the operating

system (for good and ill, mostly ill), while NT provided the necessary wide area network (WAN) protocols that made the Internet accessible. With Windows Server 2000, the big advance was the incorporation of Active Directory, domains, and security structures.

Windows Server 2008 will most likely become best known for the APIs that managed the code necessary to integrate the .NET Framework into the operating systems. .NET 3.0 isolates the user interface aspects of an application on the client, while allowing the services themselves to be abstracted. Services become truly distributed so that they can run seamlessly on a local server or remote server.

Windows .NET 3.0 communicates with Windows Server 2008 through the four server APIs shown in Figure 1.4.

FIGURE 1.4
The .NET 3.0 stack in Vista and now Windows Server 08



Windows Workflow Foundation (WWF) WWF is an API that allows an application to schedule and manage tasks, as well as sequences of tasks as integrated processes that are called workflows. WWF provides the system calls to apply business logic to objects, and the necessary mechanics to ensure that transaction integrity is maintained.

WWF manages objects in the .NET namespace, has a workflow engine, and supports program code created by Visual Studio 2005. The Microsoft website for WWF is at

`http://msdn2.microsoft.com/en-us/netframework/aa663328.aspx`

Windows CardSpace (WCS) Formerly InfoCard, WCS is a software framework that provides access to secure digital identity stores, and access to authentication mechanisms for verifying logins on the Internet. CardSpace is part of the Microsoft Identity Metasystem initiative, whose goal is to unify identities across vendors and applications.

CardSpace applications demand virtual identification cards from users, and then create a secure token that is passed to the authentication authority users to define virtual information cards that can be passed to applications as user queries. When the user successfully answers the questions, the service transmits a digitally signed XML token that provides access to the application. CardSpace resources can be found at

`http://msdn2.microsoft.com/en-us/netframework/aa663320.aspx`

Windows Communication Foundation (WCF) WCF is the new IPC communication stack and was described previously under the Indigo project. WCF is a service-oriented model that accesses the .NET namespace and that can span multiple network transport protocols, security systems, and messaging formats. Since WCF can span different services and protocols, it can also communicate across different topologies and architectures. Microsoft hopes to unify many of its older web services with WCF. For more information about WCF, refer to the MSDN site at

`http://msdn2.microsoft.com/en-us/netframework/aa663324.aspx`

Windows Presentation Foundation (WPF) WPF is the user interface portion of the service and was described in detail under the Avalon project. More technical information about WPF may be found on MSDN at

`http://msdn2.microsoft.com/en-us/netframework/aa663326.aspx`

While all of these subsystems are APIs, they will be expressed by applications as a set of common features and functions that is enabled by Windows Server 2008 in the same sense that Active Directory was expressed in domain and application logons and in the common management routines used for diverse hardware and software. The .NET footprint will give Windows Server 2008 and Vista a certain look and feel that is unique to this particular server and client.

Kernel Modifications

From a hardware perspective (Figure 1.5) the kernel serves the same function that a conductor in an orchestra does. From a software perspective, the kernel is the program that interprets program commands to determine what operations are possible and what operations are allowed. The kernel doesn't directly communicate with the microprocessor. Instead, the kernel passes its commands in through to an assembler, which translates the commands into assembly or machine language, which is the actual microcode that the microprocessor understands. Modern assemblers create object code from assembly instructions.

Hardware I/O is made a kernel function for good reason. The kernel runs in what is called *protected mode*, isolating software from hardware. Figure 1.5 shows the basic topology of the kernel. The kernel isolates hardware from bad programs and bad programmers, and isolates good programs and good programmers from bad hardware. When a device fails, the kernel provides a graceful recovery from catastrophic failure.

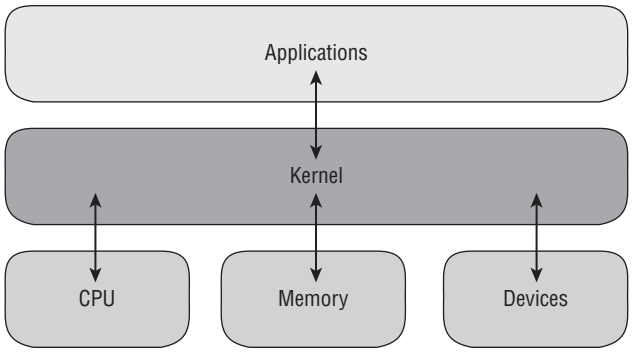


FIGURE 1.5
An operating system's kernel isolates hardware from software.

In Figure 1.6, the kernel stack is shown as a program stack, in contrast to the more hardware-centric illustration shown in Figure 1.5. The central features of the kernel are:

- Processor scheduling and threading
- Memory access and management
- Device I/O, often using device drivers
- Synchronization and communications
- Interprocess communication (IPC), or the timed management of events

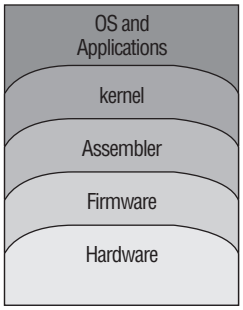


FIGURE 1.6
The kernel communicates with hardware through the assembler.

Changes to an operating system's kernel have a fundamental impact on all application and services. Although Microsoft doesn't advertise kernel changes to the general public, in part because they are more often used by developers rather than end users, Windows Server 2008 has several kernel enhancements that will significantly improve the system. Changes were made in Windows Server 2008 to the following system kernel functions:

- Memory and heap management. Memory is allocated from a large pool of available and unused memory that is called the *heap*. The heap is managed by indirect reference using a set of algorithms that organize the memory, allocate memory, and release the memory when it is no longer needed. Changes to the heap management routines can result in faster performance and better reliability. Every so often, Microsoft changes these routines, as is the case for the Windows Server 2008 and Vista core.
- Registry management; new features include performance optimizations, transactional support for Registry reads and writes, Registry filtering, and a Registry virtualization feature that can run legacy applications in isolated nonadministrator accounts.

- A new service model; a new type of service called Delayed Auto-Start decreases your server's apparent boot time by waiting until after your system boots to start these additional unattended started services. Other changes to services included a change in the Service Control Manager to isolate services by running them at the lowest level of privilege that they can allow, thus reducing their security vulnerabilities.
- Improvements to the Hardware Abstraction Layer (HAL) mechanism that support the new EFI and BCD boot environments (see the upcoming section "The New Boot Environment").
- The new Windows Hardware Error Architecture (WHEA). WHEA continues Microsoft's long, hard road down the path to meaningful system error diagnostics. It builds on the PCI Express Advanced Reporting to provide more detailed information about system errors and a common reporting structure.
- Kernel patch protection, code signing to ensure code integrity, and support for a more restricted version of protected processes—all of which are security features. These features are not new, but have been upgraded.
- A new dynamic link library (DLL) loader and improved thread pool.
- Hardware features such as dynamic partitioning, improvements to PCI/PCI Express interface, and the new boot environment.
- Power management and the ACPI 2.0 (Advanced Configuration and Power Interface) power-saving feature.
- Support for improved Plug and Play (PnP).
- Kernel-based digital signature security for drivers that can prevent unauthorized streamed content, for system patches, and for kernel-mode software installation.

Windows Server 2008 and Vista SP1 share the same kernel. That means that the first refresh of Vista to SP1 will bring many of these enhancements not only to the Windows Server platform but to the Vista client as well. It also means that Vista SP1 was in fact a major upgrade for Vista, as it is very rare for Microsoft to do a full kernel swap of a client as part of a point release. Vista SP1 is essentially a "new" operating system, and a full partner of Windows Server 2008.

Memory Support

The Windows operating system has a sophisticated memory manager, but in order to establish new capabilities in memory hardware and software access memory addressing is often changed. Microsoft made several changes to memory to support enterprise-class server systems, dynamic hardware partitioning, the new video architecture, and Terminal Services. They also modified their caching algorithms. One particular area that was completely reworked was the heap manager. The heap manager is a reference structure that points to the free store of unallocated memory. Since memory is used and released, this indirect system optimizes how memory is accessed and by whom. Memory access affects performance as well as the basic security of a system.

Non-Uniform Memory Architecture (NUMA) systems are large, multiprocessor, high-performance server systems that provide shared memory access. NUMA systems are particularly valuable to large enterprise operations because a NUMA system virtualizes hardware. In a multiprocessor system, you can have some processors running Unix and others running Windows, with each instance of an operating system running independently of the others. As the organization requires more computational power for one operating system, additional processors can be taken from one OS and given to the other. For NUMA to run effectively, it must be able to manage both CPUs and

memory as a dynamic pool. The Memory Manager in Windows Server 2008 was rewritten to allow better handling of the allocation of virtual addresses, kernel page tables, and for the support for very large Registries, all features that make NUMA more robust.

If you've been running Vista, you know that the video subsystem was substantially upgraded from previous Windows versions. Part of the new graphics subsystem in Windows Server 2008 and Vista offloads more graphics routines from the CPU onto the video card's graphics processing unit (GPU), which calls for considerably more video memory. The Aero Glass interface requires a minimum of 256MB of video RAM. Simply maintaining the Windows+Alt CoolSwitch, Flip, or Task Switcher application requires something on the order of 56MB of video RAM.

Terminal Server received a big enhancement in Windows Server 2008, and that is the subject of Chapter 12. The main operational difference from a user prospective is the Terminal Services RemoteApp feature. A RemoteApp is a program that you can run in a Terminal Server session that behaves as if it was running locally on the user's computer. RemoteApp removes the problems users had with managing Terminal Server sessions and moving between the local desktop and the remote desktop. When two or more RemoteApps run, they can do so within the same Terminal Server session. To support RemoteApps the Memory Manager introduces a new type of memory construct called Terminal Services session objects. RemoteApps makes the experience of using the virtual desktop of Terminal Server equivalent (also known as transparent) in terms of user experience to local applications, something that will make it much easier for users to learn and use.

The New Boot Environment

Microsoft has created a completely new boot environment for Windows Vista and Windows Server 2008 that greatly improves the installation of the operating system, handling of backups and restores from images, and direct system startup into a mini-OS that is capable of network installations, system diagnostics, and repairs. The most important part of this new feature is the Boot Configuration Data (BCD) store. BCD is Windows Server 2008 itself, albeit a stripped-down version of Windows.

The BCD can boot from the PC BIOS or from the newer Extensible Firmware Interface (EFI). EFI is the renamed version of the Intel Boot Initiative, which is now supported by an industry group called the Unified Extensible Firmware Interface (UEFI) Forum (`http://www.uefi.org`). EFI is a software interface between the operating system and your computer's firmware that maintains a current table of platform information; boot services such as console, bus, and file system services; and runtime services such as Date/Time and the information contained in nonvolatile RAM. One of the nicest features of EFI's boot manager is the ability to abstract device drivers, a bootable shell (EFI Shell), and an extensible architecture.

NOTE Sixty-four-bit Windows Server 2008 will not boot from a system with PC BIOS, or by using an abstraction layer. EFI is a requirement for any Intel Itanium system loading 64-bit Windows Server; Vista makes no such requirement on 64-bit chips such as AMD Opteron and others.

BCD is available during and after system boot, and while it was meant to support Vista and Windows Server 2008 it is a great general-purpose boot environment that can support numerous third-party tools. Indeed, many utilities will appear based on BCD in the future. There is some backward compatibility to Windows 2003 and XP; BCD will correctly launch `NTLDR.EXE`, which then draws its configurational data from the `BOOT.INI` file. Since BCD is so central to Microsoft's automated Windows deployment system, you'll find a more complete discussion of the BCD and deployment in the next chapter.

Windows Server 2008 SP1 and the 64-Bit Architecture

Microsoft has said on several occasions that Windows Server 2008 is the last version of the Windows operating system that would be designed in a 32-bit version. Therefore, if you aren't supporting an older application or some specialized hardware, installing the 64-bit version of Windows Server 2008 is a more forward-looking decision. Rumor is that Microsoft will release some service packs for its 32-bit server, but how long that version will continue to be supported is somewhat vague.

You can run 32-bit programs on 64-bit Windows Server 2008 in emulation. An emulator called WoW64 (Windows-on-Windows 64-bit) that runs in user mode interfaces between the 32-bit version of `NTDLL.DLL` and the kernel, intercepting kernel calls and redirecting them. WoW64 provides core emulation for `NTOSKRNL.EXE`. The `WOW64.WIN.DLL` performs memory management for 32-bit Windows programs, "thunking" data. Thunking data means that there is either a delayed computation or a mapping of machine data from one system form to another for compatibility. In this case, it is the latter. WoW64 maps 32-bit calls to 64-bit addresses, analyzes the returned data, and maps the data back to the 32-bit application running on Windows Server 2008. Support for 32-bit programs on Vista or Windows Server 2008 is good but not perfect, and with all emulations performance suffers. Native 64-bit programs almost always run faster because the code is optimized for higher throughput.

Virtualization and the Hypervisor

Most studies of network computing have consistently shown that the majority of servers operate with very low CPU utilization rates. If nearly 80 percent to 90 percent of your processing power is idle, you have not gotten your money's worth and there's an opportunity to realize considerable value. That's why virtualization is a hot topic; virtualization saves you money across the board with machines, plant costs, software, and personnel costs.

One of Microsoft's major project goals for Windows Server 2008 was to deliver a virtual machine technology that was bulletproof, highly compatible with the array of Windows software, and robust. The theme of virtualization is a common one, and is applied to the concept of dynamic allocation of resources. Virtualization abstracts physical hardware through a programmed reference system that dynamically points to and manages available assets. Physical memory is always virtualized with swap files on disk, and storage can be virtualized through RAID and across systems with a volume-based namespace.

Microsoft's goals were to virtualize CPU, memory, storage, and even network interfaces, and thus to be able to hot-swap virtual machines across hardware even as they are running. (The ability to hot-swap an entire operating system instance is something that is called Live Migration, but that is one feature that didn't make the first release of Microsoft's Windows Server 2008 virtualization package and will ship in a later version.) Imagine being able to run several operating systems or instances of an operating system, all isolated in their own virtual machine. You simply assign processors from a pool to an application instance upon demand. Virtualization is very powerful.

Windows Server virtualization is the result of a project that was code-named Viridian, and is an ongoing project at Microsoft. The group formed shortly after Microsoft acquired Connectrix and their Virtual PC program. Windows Server virtualization achieved some of these goals, and surely there is more to come. Microsoft System Center Virtual Machine Manager 2007, part of the Microsoft System Center family of system management products, permits you to define policies about resources and performance and lets you control all these features from a single console.

Virtualization is achieved using a program called a hypervisor. A hypervisor acts as an I/O redirector. Depending on the type of hypervisor you have, it is possible to run virtual machines that are compatible with any operating system that is compatible with the platform. For the x86 architecture,

that would mean any or all of the following: different versions of Windows or DOS, Linux, NetWare, Solaris for x86, and maybe one day Macintosh. The Macintosh's Boot Camp utility allows you to run Windows on a Macintosh in a virtual machine on x86 today; an even better solution exists with Parallels (`http://www.parallels.com/`). You can find a rather extensive table comparing different virtual machine technologies at

`http://en.wikipedia.org/wiki/Comparison_of_virtual_machines`

Be forewarned that this is an area of great activity, so the list is volatile and subject to change.

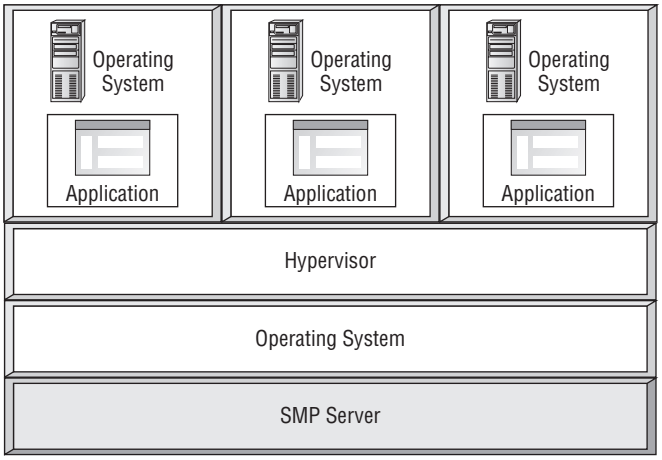
A hypervisor controls an entire computer system. Hypervisors have been around for quite some time in various forms, and they are one of the favorite methods hackers use to take full control over systems. Indeed, Microsoft's hypervisor technology (which is now called Microsoft Hyper-V and is available as an option for Windows Server 2008) will probably make it impossible for competitors to gain this type of access in the future. A hypervisor can run two or more operating systems on:

- The same processor
- Different cores of a multicore processor
- Different cores on different single or multicore processors

There are actually two types of hypervisors, and each has some limits on its capabilities:

Type 1 Hypervisor This is a lightweight operating system that runs in a layer that loads over hardware and below the operating system that the hypervisor "hosts." The hypervisor boots the system and receives privileged access to Ring 0 and other low-level services. The first Type 1 hypervisor was probably IBM's CP/CMS system; IBM still uses the technology with their z/VM OS product. Other Type 1 hypervisors are VMware's ESX Server and Sun Microsystems' Logical Domains hypervisor, which was introduced in 2005. Figure 1.7 shows a Type 1 hypervisor.

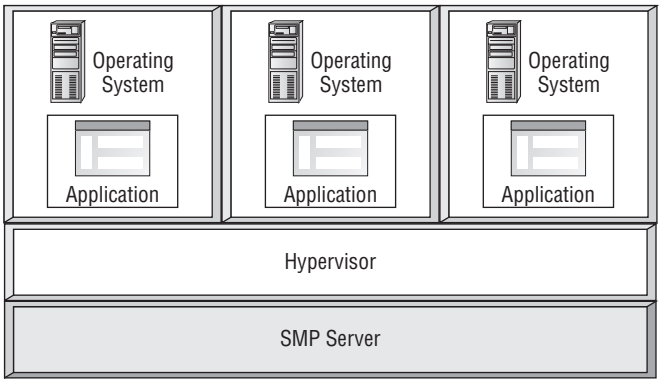
FIGURE 1.7
Type 1 Hypervisor



Type 2 Hypervisor Windows Server 2008 virtualization programs use a different type of hypervisor, referred to as a Type 2 hypervisor or Type 2 virtual machine, as shown in Figure 1.8. This type of program runs after the operating system has been loaded, and the guest operating system runs "on top of" or using the services of the hypervisor. The two most popular virtualization products in the world are probably VMware (`http://www.vmware.com`), which is now owned by EMC, and Virtual PC, which was acquired by Microsoft from Connectix. VMware is

the market leader. Both VMware and Virtual PC are Type 2 virtual machines. Figure 1.8 shows a Type 2 hypervisor.

FIGURE 1.8
Type 2 operating system



Virtual machines are often used to test applications and system configurations, to set up systems for performance testing (such as web clients used for loading), and to sandbox applications that might be unstable (such as beta software) or unsafe (testing viruses and malware). Sandboxing isolates applications from other programs on your system, and is used both for security and compatibility reasons.

Hyper-V adds virtualization capabilities directly into Windows Server 2008 (Figure 1.9), and creates a thin-layer Type 1 hypervisor that will co-exist with their type 2 Virtual Server product. You can run both or either of these two products individually, or migrate from the Virtual PC to the more powerful hypervisor product. The advantages of such a system is that it will allow finer control over hardware than a Type 2 system, easier dynamic allocations, and subsocket partitioning.

When Windows Server 2008 ships, Windows Server virtualization will allow systems to run up to 16 cores or logical processors at a time. The original plan was to allow for 64 processor instances. Other features that were scaled back to allow Windows Server 2008 to ship at the beginning of 2008 were some of the hot-add capabilities for both storage and memory. Hot swap, hot add, and Live Migration, as well as 64-processor limits, are features that competing products already have.

This version finally adds support for 64-bit programs running in 64-bit guest sessions, something that the earlier version couldn't do. Eventually Microsoft virtualization will permit users to migrate live systems entirely from one virtual system to another, and support the hot swap and hot additions of processors, memory, networking interfaces, and storage—but these features did not ship with the first release of Windows Server 2008 and were delayed until later. A Type 1 hypervisor should enable features like the hot edition of a network card or processor without reboot. The current plans are to have Windows running up to eight processors, and from 4GB to 32GB per each virtual machine.

Virtualization is an important feature, and is now part of competitive server platforms such as SUSE and Red Hat Linux, both of which have incorporated VMware into their systems. These systems use the Xen open source virtual machine, which is currently at version 3.0. This version will run Windows as a guest, and XenSource is expected to release a compatibility layer for the Windows Server 2008 hypervisor. Systems that run Xen guests can use this compatibility layer to move their sessions to Windows Server 2008. Xen is a Type 1 hypervisor; it runs in Ring 0 while its guest OSs run in Ring 1.

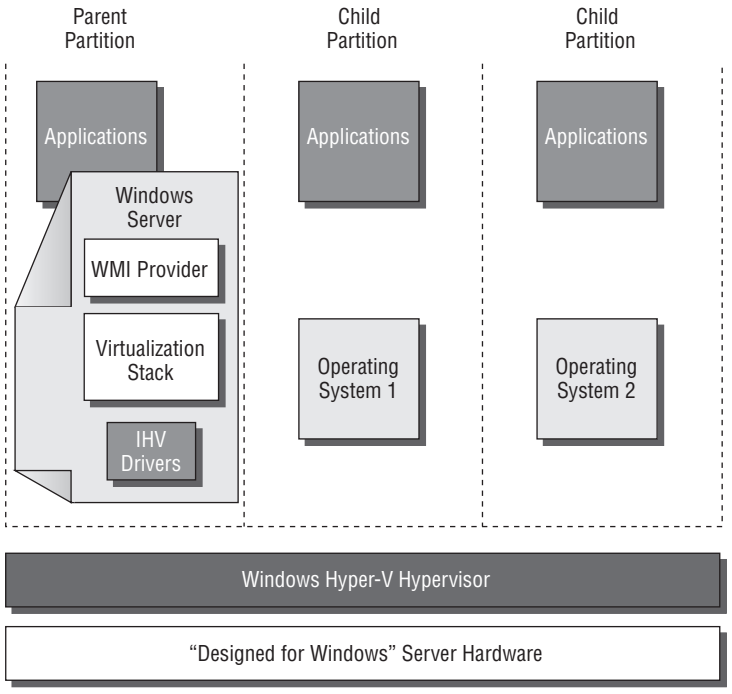


FIGURE 1.9
Windows Server 2008's virtualization architecture

Hardware vendors have also incorporated virtualization instructions directly into their processors over the last couple of years. Intel calls their technology for the x86 architecture Intel VT (for Virtualization Technology); while their virtualization technology for the 64-bit Itanium series is called VT-i. To be enabled, Intel's VT requires that both the chip set and the software support these features. AMD's hardware extensions are called AMD Virtualization, or AMD-V. All K8 processors with F stepping support AMD-V. Stepping is similar to the version number in software, in that it increases as the process gets older and more optimized; hopefully, making the processor faster and more stable. The Standard Performance Evaluation Corporation (SPEC) has a working group developing methods to test virtualization technologies using their developed benchmarks.

As you consider virtualization as an option, consider also that virtualization will probably change Microsoft's licensing model. Currently, servers are sold with a base license for which you purchase additional packages of licenses on a per-processor basis. With virtualization, Microsoft can change their model to a per-instance license. The pricing scheme at launch includes eight versions of Windows Server 2008, with Windows Server 2008 Standard, Enterprise, and Datacenter editions offered both with and without the Hyper-V virtualization technology. Virtualization will appear on both 32- and 64-bit versions of Server, with the exception of the Itanium version of Server, which comes only in the 64-bit version. On average, Hyper-V adds around $30 to the cost of a server license, per processor, for all of these versions.

Summary

In this chapter, you saw how Windows Server 2008 has advanced architecturally. Higher user-level services, presentation, and graphics got a major boost in this version. Some base modules in the kernel module were also reworked and improved. Virtualization is a work in progress, but one that will eventually be quite powerful. There were also advances in systems such as Terminal Server that will be discussed later in the book.

One of the changes in Windows Server technology is the creation of a new boot environment called Windows Preinstallation Environment (PE). Windows PE is a light version of Windows that will let you boot to a working instance of Windows Server 2008 for both installation and troubleshooting. With Windows PE, you can initiate network installations; create, modify, and use system images; and automate your deployment of multiple systems with fine control. The next chapter considers what Windows PE can do for you, and how all of this new technology will impact your installation and deployment of Windows Server 2008 and Vista.