ITPro Today™

COMPUTE ENGINES

Windows NT and VMS: The Rest of the Story

Mark Russinovich | Nov 30, 1998
Is NT really new technology?

When Microsoft released the first version of Windows NT in April 1993, the company's marketing and public relations campaign heavily emphasized the NT (i.e., New Technology) in the operating system's (OS's) name. Microsoft promoted NT as a cutting-edge OS that included all the features users expected in an OS for workstations and small to midsized servers. Although NT was a new OS in 1993, with a new API (i.e., Win32) and new user and systems-management tools, the roots of NT's core architecture and implementation extend back to the mid-1970s.

And now...the rest of the story: I'll take you on a short tour of NT's lineage, which leads back to Digital and its VMS OS. Most of NT's lead developers, including VMS's chief architect, came from Digital, and their background heavily influenced NT's development. After I talk about NT's roots, I'll discuss the

more-than-coincidental similarities between NT and VMS, and how Digital reacted to NT's release.

A Brief History of NT

NT's history is closely tied to that of David N. Cutler, NT's chief architect. After graduating from Michigan's Olivet College in 1965, Cutler worked for DuPont. Although computers weren't his first interest, he ran simulations on Digital machines as part of his job at DuPont. Before long, Cutler was knowledgeable about software and decided he wanted to develop OSs rather than application software. He joined Digital in 1971 and worked at Digital's famous "Mill" facility in Maynard, Massachusetts, developing OSs for the PDP-11 family. RSX-11M is the first OS in which Cutler incorporated major concepts and design principles that later surfaced in NT. RSX-11M is a PDP-11 OS Digital developed for industrial and manufacturing control.

In 1975, Digital realized that its competitors were developing 32-bit processors and that this technology would lure customers away from PDP's 16-bit architecture. Gordon Bell, a legendary figure in computer history and then vice president of engineering for Digital, drove the development of the 32-bit processor, which Digital eventually named VAX. By this time a star within Digital, Cutler was part of the initial VAX development team. Digital had charged Cutler, along with Dick Hustvedt and Peter Lipman, with designing VAX's OS, VMS. Digital's primary design goals for VAX hardware included backward compatibility with PDP-11 processors and enough flexibility that VAX could be the basis for low-end desktop workstations as well as enterprise-level servers. Digital also made VMS backward compatible with RSX-11M and designed VMS to run on different size machines. Of this development period, Digital states in its company history that it was "betting the business" on VAX and VMS. In an eerie echo of Digital's statement, Bill Gates recently claimed that Microsoft is "betting the business" on NT 5.0.

In 1977, Digital announced VAX-11/780 and VMS 1.0, making the first product shipments in 1978. As the project leader and one of VMS's main architects, Cutler continued work on successive releases of VMS, but he became restless at Digital. In 1981, Cutler threatened to leave Digital. To retain its star developer, Digital gave Cutler about 200 hardware and software engineers. Cutler moved his group to Seattle and started a development center. This elite group's goal was to design a new CPU architecture and OS that would lead Digital into the 1990s. Digital called the Cutler group's hardware project Prism, and its OS Mica.

In 1988, Digital executives cancelled Cutler's project and laid off many of its group members. Cutler decided to leave Digital, but before he could do so, Microsoft executives learned of the development and realized they had an ideal opportunity to hire Cutler. At the time Cutler left Digital, the release of VMS was version 5.0 (today's version is 7.1).

In August 1988, Bill Gates hired Cutler. One of Cutler's conditions for moving to Microsoft was that he could bring around 20 former Digital employees with him, including several Prism hardware engineers. Microsoft readily met this demandthe company knew hiring an OS architect of Cutler's stature was a coup, and few engineers had Cutler's track record. In addition, Gates felt that Microsoft's long-term future depended on the development of a new OS that would rival UNIX.

Microsoft's internal project name for the new OS was OS/2 NT, because Microsoft's intention was for the new OS to succeed OS/2 yet retain the OS/2 API as its primary interface. The success of Windows 3.0 in April 1990 altered Microsoft's thinking and its relationship with IBM. Six weeks after Microsoft released Windows 3.0, Microsoft renamed OS/2 NT as Windows NT, and designated the Win32 API (a 32-bit evolution of Windows 3.0's 16-bit API) NT's official API. Gates decided that compatibility with

the 16-bit Windows API and the ability to run Windows 3.x applications unmodified were NT's paramount goals, in addition to support for portions of the DOS, OS/2, and POSIX APIs. From 1990 to NT's public release in August 1993, Cutler's team was in a mad dash to complete NT, and the project grew to involve more than 200 engineers and testers. Figure 1 shows a timeline of the major events in the history of NT.

NT and VMS

Most of NT's core designers had worked on and with VMS at Digital; some had worked directly with Cutler. How could these developers prevent their VMS design decisions from affecting their design and implementation of NT? Many users believe that NT's developers carried concepts from VMS to NT, but most don't know just how similar NT and VMS are at the kernel level (despite the Usenet joke that if you increment each letter in VMS you end up with WNTWindows NT).

TABLE 1: VMS and NT Terminology Translations

VMS Term	NT Translation
Interrupt Priority Level (IPL)	Interrupt Request Level (IRQL)
Asynchronous System Trap (AST)	Asynchronous Procedure Call (APC)
Fork Procedure	Deferred Procedure Call (DPC)
I/O Request Packet (IRP)	I/O Request Packet (IRP)
Bug Check	Bug Check
System Service	System Service
sys.exe	ntoskrnl.exe
Paged Pool	Paged Pool
Nonpaged Pool	Nonpaged Pool
Look aside List	Look aside List
Section	Section

As in UNIX and most commercial OSs, NT has two modes of execution, as Figure 2 shows. In user mode, applications execute, and OS/2, DOS, and POSIX execute and export APIs for applications to use. These components are unprivileged because NT controls them and the hardware they run on. Without NT's permission, these components cannot directly access hardware. In addition, the components and hardware cannot access each other's memory space, nor can they access the memory associated with NT's kernel. The components in user mode must call on the kernel if they want to access

hardware or allocate physical or logical resources.

The kernel executes in a privileged mode: It can directly access memory and hardware. The kernel consists of several Executive subsystems, which are responsible for managing resources, including the Process Manager, the I/O Manager, the Virtual Memory Manager, the Security Reference Monitor, and a microkernel that handles scheduling and interrupts. The system dynamically loads device drivers, which are kernel components that interface NT to different peripheral devices. The hardware abstraction layer (HAL) hides the specific intricacies of an underlying CPU and motherboard from NT. NT's native API is the API that user-mode applications use to speak to the kernel. This native API is mostly undocumented, because applications are supposed to speak Win32, DOS, OS/2, POSIX, or Win16, and these respective OS environments interact with the kernel on the application's behalf.

VMS doesn't have different OS personalities, as NT does, but its kernel and Executive subsystems are clear predecessors to NT's. Digital developers wrote the VMS kernel almost entirely in VAX assembly language. To be portable across different CPU architectures, Microsoft developers wrote NT's kernel almost entirely in C. In developing NT, these designers rewrote VMS in C, cleaning up, tuning, tweaking, and adding some new functionality and capabilities as they went. This statement is in danger of trivializing their efforts; after all, the designers built a new API (i.e., Win32), a new file system (i.e., NTFS), and a new graphical interface subsystem and administrative environment while maintaining backward compatibility with DOS, OS/2, POSIX, and Win16. Nevertheless, the migration of VMS internals to NT was so thorough that within a few weeks of NT's release, Digital engineers noticed the striking similarities.

Those similarities could fill a book. In fact, you can read sections of *VAX/VMS Internals and Data*

Structures (Digital Press) as an accurate description of NT internals simply by translating VMS terms to NT terms. Table 1 lists a few VMS terms and their NT translations. Although I won't go into detail, I will discuss some of the major similarities and differences between Windows NT 3.1 and VMS 5.0, the last version of VMS Dave Cutler and his team might have influenced. This discussion assumes you have some familiarity with OS concepts (for background information about NT's architecture, see "Windows NT Architecture, Part 1" March 1998 and "Windows NT Architecture, Part 2" April 1998).

NT's processes are virtually the same as VMS's processes (Table 2, page 118, shows a comparison of VMS and NT processes). In NT, as in VMS, the process scheduler implements 32 priority levels. The process with the highest priority is always running, and processes with equal priority are scheduled in a round-robin pattern. The system considers the 16 high-priority levels realtime or fixed priorities, because the process scheduler doesn't manipulate priority in processes the system assigns to that range. The 16 low-priority levels (except 0, which the system reserves for the idle thread that executes when nothing else can) are dynamic because the scheduler, often with the input of device drivers, bumps priorities up in reaction to various conditions, such as when the process receives input from a device. This bumping

TABLE 2: Significant VMS and NT Similarities

VMS	NT
Process scheduler implements 32 priority levels split into halves	Process scheduler implements 32 priority levels split into halves
Process scheduler never lowers a process' priority below the priority level the application programmed	Process scheduler never lowers a process' priority below the priority level the application programmed
Uses boosting to handle CPU hogging	Uses boosting to handle CPU hogging
Supports SMP	Supports SMP
Digital introduces kernel threads in VMS 7.0	NT 3.1 uses kernel threads
Relies heavily on memory-mapped files	Relies heavily on memory-mapped files
Uses demand-paged virtual memory for physical memory management	Uses demand-paged virtual memory for physical memory management
Uses working sets with a clock-based replacement algorithm	Uses working sets with a clock-based replacement algorithm
Balance Set Manager uses	Balance Set Manager doesn't

procedure is called boosting. A defining aspect of the NT and VMS schedulers is that they never lower a process' priority below the priority level the application programmed. To handle CPU hogging, in which a process burns CPU cycles without regard to other processes in the system, the scheduler boosts the priority of starved processes that haven't executed for a defined period. Both VMS 5.0 and NT 3.1 schedulers support symmetric multiprocessing (SMP), which let them execute processes simultaneously on different CPUs in order to increase applications' performance.

swapping to handle the system's memory demands	use swapping
Supports a layered-driver model throughout the device driver stacks	Supports a layered-driver model throughout the device driver stacks
Implements asynchronous packet-based I/O commands	Implements asynchronous packet-based I/O commands
Represents resources as objects managed by an Object Manager	Represents resources as objects managed by an Object Manager
Security subsystem based on objects with access control lists (ACLs)	Security subsystem based on objects with ACLs
MONITOR	Performance Monitor
BACKUP	NT Backup

A major difference between NT process management and VMS process management is that NT processes contain one or more threads of execution, and NT's scheduler gives CPU time to threads, not processes. Digital didn't introduce kernel threads into VMS until version 7.0 in 1995. This addition is one of several enhancements Digital has made to VMS since NT's release that appear to be in response to NT capabilities. In turn, Microsoft added lightweight user-mode threads support to NT 4.0 in 1996, which it copied from the VMS implementation of threads.

The memory managers in NT and VMS are also similar. Both OSs implement virtual memory address maps that the system splits between the currently executing application and the kernel. Both NT and VMS rely heavily on memory-mapped files, especially for mapping the code for executing applications and implementing copy-on-write functionality (because of VAX hardware limitations, VMS provides

less efficient copy on demand funtionality). Physical memory management in NT and VMS relies on demand-paged virtual memory. VMS's memory manager assigns each process upper and lower limits (called working sets) for the amount of physical memory the system can assign them. This feature compartmentalizes applications so that an application with heavy memory demands minimally affects other processes. NT's memory manager incorporates working sets, along with many subtleties of the VMS working-set tuning algorithms.

As with the process manager, notable differences exist between NT's and VMS's memory manager. VMS's Balance Set Manager moves entire processes' memory footprints out of memory to paging files and back to memory in response to the overall memory demands of the system. Microsoft did not carry this mechanism, known as swapping, into NT's Balance Set Manager, although some of NT's Balance Set Manager's secondary responsibilities are the same as the secondary responsibilities of VMS's Balance Set Manager.

NT's I/O Manager is closely based on VMS's I/O Manager. Both OS's I/O Manager support a layered-driver model throughout the device driver stacks for different device types and implements asynchronous packet-based I/O commands, and its device drivers dynamically load and unload. Stackable and loadable drivers make NT and VMS very extensible. Either OS can divide functionality among several device drivers, with each driver implementing a different abstraction level. For example, the system can insert a fault-tolerant disk driver between a file system driver and a disk driver. This configuration lets the fault-tolerant disk driver receive a request the system sends to one logical drive (e.g., the C drive), then send the request to multiple physical drives to implement mirroring or striping. Asynchronous I/O enables applications and the kernel subsystems to initiate device requests and work while the requests are in progress, rather than wait idly for the requests to complete. NT's device driver

architecture and interrupt-request priority scheme are based on VMS. Descriptions of these aspects of the I/O Manager are applicable to both OSs with little variation.

As you can see by comparing Figure 2 and Figure 3, page 117, the Executive subsystems exhibit the most significant resemblance between VMS and NT. But many minor similarities exist in which it is clear that Microsoft derived NT's capabilities from VMS. For example, both NT and VMS represent resources as objects that the system manages through an Object Manager, which implements uniform reference counting and accounting. The Object Manager regulates resource allocation and calls the Executive subsystem functions that request notification of certain object operations. VMS object management is not formalized, like it is in NT, and the VMS Object Manager is just a loose connection of functions. Microsoft extended NT's Object Manager so that it provides a uniform naming model for all kernel resources.

NT's security subsystem is based on objects with discretionary access control lists. DACLs determine which users can perform various operations on those objects. Digital added a DACL enhancement to VMS's security model in version 4.0 in 1984. Therefore, VMS's security implementation is the predecessor to NT's. Microsoft even included systems tools similar to VMS's in NT, including the Performance Monitor, which is based on MONITOR, the extensible VMS performance monitor. VMS included a utility called BACKUP long before Microsoft developed NT's backup utility.

"Why the Fastest Chip Didn't Win" (*Business Week,* April 28, 1997) states that when Digital engineers noticed the similarities between VMS and NT, they brought their observations to senior management. Rather than suing, Digital cut a deal with Microsoft. In the summer of 1995, Digital announced Affinity for OpenVMS, a program that required Microsoft to help train Digital NT technicians, help promote NT

and Open-VMS as two pieces of a three-tiered client/server networking solution, and promise to maintain NT support for the Alpha processor. Microsoft also paid Digital between 65 million and 100 million dollars.

The Evolution of NT and VMS

Although Microsoft presents NT as a homegrown OS, NT is actually much older than its official 1993 birthdate. NT contains architectural and design influences from another company's flagship OS. Interestingly, throughout the 1990s, Digital introduced many NT features to VMS, and Microsoft has added VMS developments to NT. For example, VMS featured native clustering support in 1984, and 64-bit memory and system APIs in 1996. Microsoft did not introduce clustering support to NT until late last yearand only on a limited scaleand several years might pass before Microsoft releases 64-bit NT. Reciprocally, Microsoft released NT's first version with support for kernel-mode threads, system-wide event logging, and a configuration database called the Registry. VMS introduced kernal-mode threads in VMS 7.0 in 1995, and VMS 7.2 will include NT-style event logging and a Registry.

The saga goes on. Now that Compaq has acquired Digital, will VMS continue to evolve, or will NT seal the fate of its predecessor? One thing is certain: NT will continue to grow, leaving its origins further and further behind.