BASICS OF z/VM Virtualization

BY BILL BITNER & BRIAN WADE, PH.D.

The z/VM product has its roots in an IBM product named VM/370, released in August 1972. VM/370 grew from earlier unsold versions of virtual machine technology dating back to 1967. This article explores some of the current product's constructs and concepts. Let's start with the basics.

VIRTUAL MACHINES

The IBM System z processors operate according to a data processing machine architecture defined in *z/Architecture Principles of Operation*. This book explains the machine's instruction set, its I/O processing rules, and how it runs programs. Engineers implement z/Architecture in IBM System z machines such as those in the IBM z9 family.

The role of many operating system kernels is to create well-defined program execution contexts in which applications can run concurrently without interfering with one another. For z/VM, these execution contexts are called virtual machines. Where an ordinary operating system might create execution contexts for application programs, z/VM creates execution contexts for System z operating systems. In other words, z/VM's task is to create virtual machines adhering to z/Architecture. z/VM can create nearly any virtual configuration that could legitimately exist in the real hardware. Further, z/VM lets many such virtual machines exist and operate simultaneously on one instance of real hardware; in fact, in the most interesting cases, z/VM overcommits the real hardware. A virtual machine also is often known as a VM user ID, a VM logon, a VM guest, or a virtual server.

Just as a real System z machine consists of real processors, real memory, and real I/O devices, a virtual machine consists of virtual processors, virtual memory, and virtual I/O devices. For example, z/VM might equip a virtual machine with two virtual processors, 1GB of virtual memory, a virtual Ethernet card, and a virtual 3270 terminal device. Because of the histories of z/VM and its guests, some of the virtual devices found in a typical virtual machine's configuration correspond to real devices IBM no longer manufactures. For example, each virtual machine will typically have a virtual card reader, >

virtual card punch, and virtual printer. Many guests still effectively use these antiquated virtual devices even today.

Various System z operating systems can run in virtual machines. One, Conversational Monitor System (CMS), is sold as part of z/VM. CMS is an interactive, single-user operating system meant to run in a virtual machine. Though for many years CMS and its applications were a mainstay of IBM's offerings in interactive general-purpose computing, CMS recently has found its niche in supporting applications meant to help manage the z/VM system. The z/VM TCP/IP stack, for example, runs in a CMS virtual machine. CMS isn't the only System z operating system that can run virtually. Other System z operating systems, such as z/OS, z/VSE, z/TPF, and Linux for System z, can all run in virtual machines on z/VM.

The heart of z/VM is a multi-programming, multi-processing operating system kernel known as the Control Program (CP). (Be careful when using the term "CP," for in some communities, "CP" means "central processor.") CP is the component of z/VM that creates and dispatches virtual machines on the real System z hardware.

CP supports hundreds of commands. Some control the configuration of the virtual machine. For example, the CP DEFINE command lets a customer create a virtual device and add it to a virtual machine's configuration. Though CP lets customers change virtual machine configurations nearly at will, customers usually define their virtual machines ahead of time in a user enrollment and definition file called the CP directory. The CP directory describes virtual machines and sets attributes for each one, such as number of virtual processors, amount of virtual memory, and complement of disk devices.

Because of the nesting capabilities of IBM's System z virtualization technology, the System z computer on which z/VM perceives it's running might itself be mythical. A commonly seen nesting situation is the one where z/VM runs in a Logical Partition (LPAR) of a System z computer. A System z hardware facility called the LPAR hypervisor creates and manages logical partitions in much the same way that z/VM creates and manages virtual machines. System z machines contain special hardware that lets this commonly found two-level nesting arrangement run with almost no performance penalty.

Virtualization of Processors

Because z/Architecture defines that a System z data processing system can house one to 64 processors, each virtual machine z/VM creates can have one to 64 virtual processors. Usually, these are all general-purpose instruction processors. However, z/VM does let the customer define some of a guest's virtual processors to be specialty processors such as System z Application Assist Processors (zAAPs), System z Integrated Information Processors (zIIPs), or System z Integrated Facility for Linux (IFL) processors. z/VM dispatches guests' virtual processors on the processors of the System z machine it's controlling.

z/VM provides control over processor resources by letting a system administrator assign a share value to each virtual machine. This share value typically sets the minimum amount of processor resource a virtual machine can expect CP to allocate to it. z/VM also lets the system administrator define a maximum share value to prevent a guest from excessively consuming processor resource. The z/VM system administrator or system operator can adjust share settings while virtual machines are running.

The Heart of Virtualization

System z virtualization depends on a hardware instruction called Start Interpretive Execution (SIE, pronounced "sigh"). This instruction gives CP a means to tell System z hardware to run a virtual processor. CP and the System z hardware cooperate in their use of a memory-resident control structure—called the SIE State Descriptor or the SIE Block—to track the state of the virtual processor. The SIE block contains such things as the virtual processor's last register values, the pointers to the owning guest's DAT tables, and other state information that lets a virtual processor continue to run instructions. The address of the SIE block is an operand of the SIE instruction.

Owing to the completeness of the virtual processor description stored in the SIE block, the System z hardware can handle many of the virtual processor's needs without requiring the help of CP. However, some conditions, such as occurrence of a page fault or the guest's use of an I/O instruction, are too complex for the hardware to handle on its own. When the System z hardware encounters such a condition, it ends the virtual processor's time in SIE and gives control back to CP. This phenomenon is called a SIE exit or a SIE break. CP responds to the SIE break by handling the condition and eventually redispatching the virtual processor by issuing another SIE instruction.

A customer obtains best utility from a System z machine when the machine spends its time in SIE running guests, instead of outside of SIE running CP. z/VM performance engineers have means to keep track of CPU time spent inside and outside of SIE and to keep track of the reasons why virtual processors leave SIE. An important aspect of z/VM performance management is building guest operating systems and their applications to be aware they're running virtually and to accomplish their work using techniques that tend to promote remaining in SIE. IBM has spent many years tuning its long-lived System z operating systems, such as CMS, z/OS, and z/VSE, to use SIE-friendly techniques. Linux for System z, being comparatively young, is still evolving.

Virtualization of Memory

Memory can be defined in the CP directory or can be changed with the CP DEFINE command. The virtual machine memory is usually defined as a contiguous range. However, one can define gaps in the memory. This can be helpful in some situations.

Of course, all guest memory is virtual. To virtualize memory, z/VM harnesses the System z processor's Dynamic Address Translation (DAT) facility. With DAT, the System z processor translates each instruction address or operand address from virtual to real, using translation tables CP maintains. CP overcommits physical memory by keeping resident only those guest pages that appear to have been needed in the recent past, constantly adjusting the DAT tables to preserve the memory illusion for the guests. When physical memory is scarce, CP moves stagnant guest pages first to expanded storage (a high-speed page storage buffer) and eventually to disk. CP brings these pages back to memory if the guest ever needs them again.

CP manages the level of physical memory overcommitment by considering a guest's apparent memory need as part of deciding whether to run (dispatch) the guest. If CP determines that letting a virtual machine run could cause the system to enter a thrashing condition because of an insurmountable lack of physical memory, CP prohibits

CSI International provides mainframe enterprises with excellent products, outstanding customer support, and fair pricing.

COME GROW WITH US

We are investing in new products, services, and support offerings that provide real value. Choose CSI and join the growing number of customers that are saying NO to outlandish upgrade fees and visions that never reach fruition.

CSI International
sales@CSI-International.com
www.CSI-International.com
800.795.4914

"We deliver what the competition can only promise."

the virtual machine from being dispatched for a period of time. Instead of moving the virtual machine to the dispatch list to run, CP places the virtual machine on the eligible list, which is a list of virtual machines that are ready to run, but which CP is holding back because there appears to be too little physical resource available to run them.

The tendency of a virtual machine to incur page faults is directly related to the amount of physical memory CP has set aside for the virtual machine's pages. Usually, CP makes this determination on its own. However, the CP SET RESERVE command lets the operator influence the determination, telling CP to reserve at least said minimum number of real frames to hold the guest's pages. When the operator uses SET RESERVE, he's offering a guest favored status with respect to memory consumption. Using SET RESERVE is a valuable performance tuning technique.

z/VM lets virtual machines share memory, which helps reduce memory requirements. z/VM has three kinds of shared memory. The first, a Discontiguous Saved Segment (DCSS), is a range of guest memory addresses for which all participating guests see the same physical memory pages. A z/VM systems programmer can place commonly used data or programs in a DCSS, letting many virtual machines share one physical copy.

With the second type of shared memory, a Named Saved System (NSS), participating guests share a physical copy of the data; in addition, the "data" is typically a bootable operating system. This lets many guests share, for example, a single copy of the Linux kernel or CMS nucleus. Booting from memory offers speed advantages as well as memory economy.

The third type of shared memory, a VM Data Space, is similar to a DCSS, but offers much more addressability. With VM Data Spaces, the shared data are in one or more distinct address spaces, each address space being entirely available for sharing. The participating guests access those address spaces using a System z operand addressing architecture called Access Register (AR) mode. With AR mode, a single System z instruction can refer to operands located in more than one address space.

Virtualization of I/O Devices

z/VM uses various methods to provide devices to virtual machines. First, CP can dedicate, or attach, a real device to a virtual machine. This gives the virtual machine exclusive use of the entire real device. Tape drives are typically attached to virtual machines. CP also can virtualize a device, which means it gives a guest a portion of a real device. This can be a portion in time, such as of a processor, or a portion of the device's storage capacity, such as of a disk drive. Simulation of devices is a third approach. Earlier we discussed devices such as a virtual card reader. This is an example of a device where real hardware isn't present, but CP simulates it using memory and disk. The last approach we'll mention, emulation, is when CP uses hardware of one type to create the illusion of a similar type. For example, CP uses modern SCSI disks to cause guests to believe that an older, no-longer-manufactured style of disk, called Fixed Block Architecture (FBA), is present.

z/VM provides disks to guests in various ways. While CP can dedicate entire disk volumes to virtual machines, more common is for CP to divide real disk volumes into disjoint, contiguous cylinder or block ranges called minidisks, thereby letting many guests each use some fraction of a real volume's storage capacity. Minidisks can be exclusive to virtual machines, or many virtual machines can use a single minidisk simultaneously, thereby sharing data.

One particularly interesting kind of disk z/VM can provide for a guest is the Virtual Disk in Storage or VDISK. A VDISK appears to the guest as an FBA disk drive with extremely fast performance. The VDISK performs well because z/VM backs the VDISK in paged memory instead of on real disk hardware. Because CP doesn't back a VDISK with permanent disk, the data is volatile. Even so, VDISKs are handy in several situations. In particular, VDISKs are an especially good choice for Linux swap space.

One last disk feature worth mentioning is Temporary Disk (TDISK). The system administrator can assign CP a pool of disk volumes it can use to instantiate minidisks users need for only a short time. The CP DEFINE command lets the z/VM user define such a minidisk; when a user does so, CP finds some free space in the pool and uses it to create the minidisk. When the user no longer needs the minidisk, he issues the CP DETACH command to dispose of it, and CP clears the space and returns it to the pool.

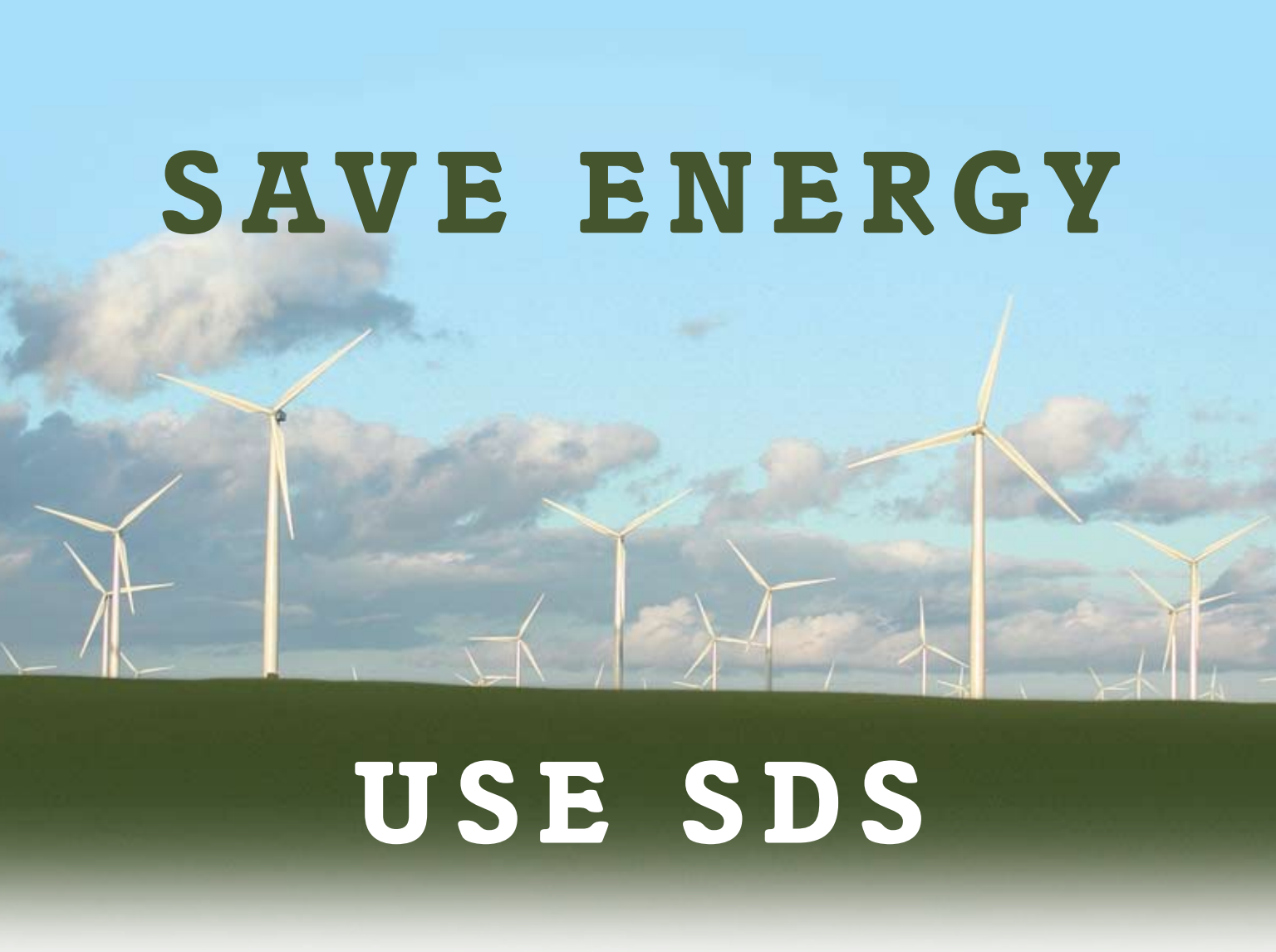
Because CP mediates access to minidisks, it can use memory to improve their performance. Central to z/VM's minidisk strategy is the CP Minidisk Cache (MDC). With MDC, CP uses real memory or expanded memory to cache recently read portions of minidisks. This greatly improves performance for minidisks that are frequently read, such as those containing object code libraries or frequently used binaries. The minidisk cache is a write-through cache, which means that if a guest writes to blocks that are cached, CP updates the cache and commits the change to the minidisk before informing the guest that the write is complete. A z/VM system administrator or operator can use the CP SET MDCACHE command to control or configure the minidisk cache.

Network connectivity is an important concern in many environments. z/VM meets customers' network needs by offering several networking options. CP can dedicate network devices to virtual machines. The dedicated device can be a channel-to-channel adapter, an IBM Open Systems Adapter (OSA) that provides Ethernet connectivity, or a HiperSockets device, a kind of network adapter that connects one LPAR to another. z/VM also has its own TCP/IP stack, which guests can use as if it were an IP router. A common network option used today is the virtual switch. Here, CP equips each virtual machine with a simulated IBM OSA and connects all those simulated OSAs to a simulated LAN segment called a guest LAN. Also connected to the guest LAN is a real OSA that CP manages. With this configuration established, CP can provide packet- or frame-switching functions for the guests, just as a real switch would in a real external network. In this way, the guest LAN becomes an extension of a real external LAN segment.

Diagnostic and Programming Services

z/VM offers a variety of debug facilities, making it invaluable for developing operating systems for System z. Commands exist to display, search, or modify virtual machine memory. In displaying memory, CP can display the memory in hexadecimal, ASCII, EBCDIC, or even as disassembled assembler instructions.

The tracing facility, whose documentation exceeds 40 pages, is extensive. CP TRACE can trap references to memory, changes to registers, use of specific instructions, or arrival of interrupts, to



SAVE ENERGY

USE SDS

VIP

Best-of-Breed z/OS IP Management

- Locate performance-sapping glitches.
- Avoid resource-hog monitoring techniques.
- Maintain OSA and Enterprise Extender at peak performance.
- Perform live traces with ease.

SDS FTP Manager Transforming z/OS FTP

- Automate batch transfers to gain optimum throughput.
- Eliminate FTP-related productivity shortfalls.
- Thwart security violations.
- Expedite transfer tracking.

Putting your time and MIPS to best use.

26 YEARS 1982 - 2008 *A quarter-century of first-class software*

Free test drives available.
www.sdsusa.com • 800-443-6183

SOFTWARE DIVERSIFIED SERVICES sds

name a few. CP TRACE also includes the ability to issue commands when certain trace points are hit. The software developer can conduct tracing interactively, or let the trace run, collecting the trace records for later analysis.

z/VM provides several programming interfaces to let guests interact with CP. The System z architecture provides a special assembler instruction, Diagnose, which on real hardware performs diagnostic functions. Recognizing the utility of such a trappable instruction, CP implements an entire Application Programming Interface (API) built on Diagnose. To use the API, a guest builds a parameter list in memory, puts the address of the parameter list into a register, and then issues the Diagnose instruction. CP performs the requested operation and returns control to the guest.

CP provides more than 50 different functions through Diagnose. These functions include interrogating real device characteristics, performing I/O, or managing memory segments. CP also provides various communication APIs that connect virtual machines to one another. One of these methods, the Inter-User Communication Vehicle (IUCV), also lets a virtual machine communicate with CP. Over an IUCV connection to CP, a trusted virtual machine can help CP accomplish certain important system management functions such as accounting, performance monitoring, or security.

The Future

IBM's VM product family has thrived for four decades because IBM has constantly improved VM to match the market's virtualization needs. In the early '70s, VM hosted a small number of ordinary guest operating systems. In the late '70s and early '80s, the number grew modestly. In the mid-80s, CMS became popular as a general-purpose interactive computing platform, owing largely to a CMS-based email and calendar package known as Professional Office System (PROFS). To handle the growth, IBM sharpened VM's ability to run many lightweight, single-user interactive virtual machines; some customers ran 20,000 office users concurrently on a single hardware footprint. The late '90s saw CMS wither as a general-purpose interactive environment, while Linux ascended. At the turn of the century, Linux on the mainframe gained a foothold, again bringing VM's virtualization capabilities to the forefront. In response to the Linux boom, IBM again changed z/VM, improving its virtualization capabilities so that CP could begin to handle a large number of Linux guests as easily as it once handled many CMS guests. z/VM's ability to adapt to the needs of the systems running in its virtual machines is a strength that should carry it forward in the next three decades.

About the Authors

BILL BITNER is a senior software engineer in the IBM Systems and Technology Group in Endicott, NY. He joined IBM in 1985, and has worked on performance in various areas of VM. He currently leads the VM Performance team.
Email: bitnerb@us.ibm.com

BRIAN WADE is a senior software engineer in the IBM Systems and Technology Group in Endicott, NY. He joined IBM in 1986 after earning his Ph.D. in Electrical Engineering from the University of Notre Dame. He currently works in z/VM Performance.
Email: bkw@us.ibm.com