

Intel's CPU Extensions Transform Virtualization

Virtualization has traditionally presented its share of design challenges in information-assurance-based systems. But now, Intel's VT-x and VT-d CPU extensions are changing the game and showing potential to become the de facto path to virtualization.



Author: Stuart Fisher, LynuxWorks, Inc.

Virtualization is a technology we have all recently come to understand on our desktop systems. Many of us today host multiple operating systems on the same computer using commercially available software. Virtualization is the method by which programs—including operating systems—can run in a software environment as though they were running on native hardware. This environment is called a Virtual Machine Monitor (VMM), also referred to as a hypervisor. Figure 1 shows a notional architecture of a VMM environment. The VMM layer is hosting a guest operating system.

In the military embedded space, applications have typically been hosted on physically separate machines to enforce safety or security requirements. With recent developments in processor architecture, it is now possible to migrate such applications onto a single machine and provide software separation using hypervisors and separation kernels. This is important to the defense industry as it facilitates legacy-code hosting on the same platform as newly developed architectures, reducing hardware footprint and achieving software certification for safety and security.

For commercial or military applications, hardware support provided by Intel® VT-x

and VT-d CPU extensions simplifies processor virtualization, enabling reductions in VMM software size and complexity. This results in VMMs that can support a wider range of legacy operating systems while maintaining high performance and efficiency.

In the following sections, we will examine some of the technical difficulties developers have discovered while bringing virtualization into information-assurance-based systems. We will also discuss how Intel's hardware extensions are helping to overcome such issues.

Intel virtualization extensions: VT-x and VT-d

The advances being made today by CPU makers and hypervisor developers are helping to define the way for future virtualization platforms. New CPU extensions are not only helping to meet the high-performance requirements of future systems, but they're also making it easier to implement and support legacy operating systems.

In years to come, implementations such as VT-x and VT-d will play an increasingly important role in virtualized systems as industry adopts these types of implementations as effective hardware assis-

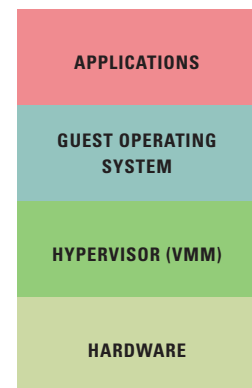


Figure 1

tance standards for future CPU architectures.

Meanwhile, hardware and software vendors are working together to make current hypervisor designs simpler and more efficient. Intel VT-x and VT-d technology leads the curve in this area for x86 architectures. Intel was the first hardware provider to offer virtualization assistance at the hardware level and is already in its second generation of this technology. Initial performance tests show that this technology is enabling fully virtualized operating systems to compare with para-virtualized versions, reducing the need for operating system modifica-

tion. Para-virtualized operating systems, by definition, require a level of porting from their native versions to support the hypervisor environment. LynxSecure's embedded hypervisor from LynxWorks™ takes full advantage of these hardware extensions and provides full virtualization for its future guest operating systems.

Emergence of virtualization techniques

Operating systems for Intel architectures are written to run directly on the native hardware, so they naturally assume they fully own the computer's resources: the x86 architecture offers levels of privilege, and operating-system code expects to run at the highest privilege level. This is fine when run as a native OS, but when virtualizing the x86 architecture, the guest OS runs at a lower-level privilege than the underlying VMM. This is necessary, as the VMM must be able to manage shared resources.

There are also differing instruction semantics when instructions are run at a different privilege level compared to that of the native implementation. The difficulty in trapping these types of instructions and privilege instruction requests at runtime was the challenge that originally made the x86 architecture so difficult to virtualize. Then in 1998, VMware developed the binary translation method.

Since the adoption of binary translation, competing hypervisor companies have differentiated their wares by employing a range of techniques to address the same problem, each trying to leverage their solution in the marketplace. The problem with that is because there are no industry standards for VMMs, we now have three different options to choose from for handling sensitive and privileged instructions:

- Binary translation
- OS-assisted (also referred to as para-virtualization)

- Hardware-assisted or full virtualization

Binary translation

The binary translation technique was, and still is, the de facto method by virtue of the number of VMware copies around the world. Its principle is to translate the nonvirtualizable privileged instructions with new sequences at runtime while user instructions execute directly on the native hardware. This combination provides full virtualization as the guest operating system is decoupled from the underlying hardware by the VMM. This method requires no hardware assist or operating system assist. The main advantage of this approach was that it allowed virtualization to become possible on x86 platforms, something thought impossible prior to this technique. The main disadvantage to this approach is that it requires OS modification at runtime, which reduces performance compared to hardware-assist techniques.

Para-virtualization

Para-virtualization is the technique whereby the VMM and guest operating systems communicate by use of hypercalls. In this situation, nonvirtualizable privileged instructions are removed and replaced with hypercalls. These hypercall interfaces also handle other critical kernel operations such as interrupt handling and memory management. Para-virtualization differs from binary translation and full virtualization in that it requires modification of the guest operating system. It should be noted that in most cases, para-virtualization offers the best performance of the three virtualization options.

Hardware-assisted virtualization

In contrast, hardware-assisted virtualization, such as Intel VT-x technology, has the advantage over traditional software

techniques because Intel controls the CPU. By introducing a new CPU execution mode feature that allows the hypervisor to run in a root mode below the normal privilege levels, the previously described issues relating to privileged instructions are overcome. Early releases of this technology, however, were slow, making para-virtualized techniques seem more beneficial. However, we are now seeing hardware-assisted performance quickly approach near-native levels.

Memory management and device I/O is a key area where hardware-assisted techniques are helping hypervisor developers. Native operating systems expect to see all of the system's memory. To run multiple guest operating systems on a single system, another level of memory virtualization is required.

This can be thought of as virtualizing the Memory Management Unit (MMU) to support the guest OS. The guest OS continues to control memory mapping within its OS, but cannot see the full machine memory. It is the responsibility of the VMM to map guest physical memory to actual machine memory. When the guest OS changes its virtual memory mapping, the VMM updates the shadow pages to enable direct lookup. The disadvantage to MMU virtualization is that it creates some overhead for all virtualization techniques, which can cause a hit in performance. It is this area where Intel's VT-x technology is providing efficiency gains.

Intel VT-d steps in

Device and I/O virtualization is the final component required to allow full virtualization to take place using hardware assist alone. This involves managing and routing I/O requests between virtual devices and the shared physical hardware. In para-virtualized systems, this translation is performed in software, but with a significant overhead. The latest

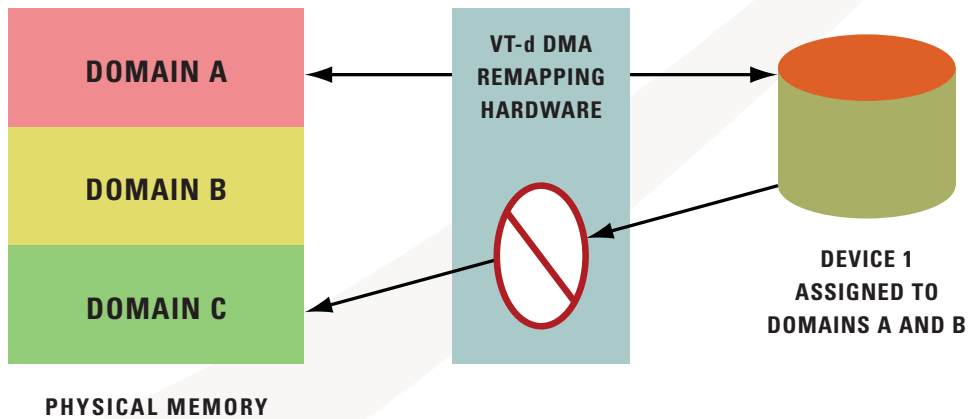


Figure 2

advances using Intel's VT-d technology for directed I/O solves these issues and has become a standard for x86 virtualization.

Intel VT-d enables system software to create multiple Direct Memory Access (DMA) protection domains. Each protection domain is an isolated environment containing a subset of the host physical memory. Depending on the software usage model, a DMA protection domain might represent memory allocated to a Virtual Machine (VM) or the DMA memory allocated by a guest-OS driver running in a VM or as part of the VMM itself. The VT-d architecture enables system software to assign one or more I/O devices to a protection domain. DMA isolation is achieved by restricting access to a protection domain's physical memory from I/O devices not assigned to it. This occurs by using address-translation tables, thereby providing the necessary isolation to assure separation between each virtual machine's computer resources.

When any given I/O device tries to gain access to a certain memory location, DMA remapping hardware looks up the address-translation tables for access permission of that device to that specific protection domain. If the device tries to access what is outside of the range it is permitted to access, the DMA remapping hardware blocks the access and reports a fault to the system software (Figure 2).

As alluded to previously, virtualization allows for the creation of multiple virtual machines on a single server. This consolidation maximizes hardware utilization, but applications require a significant amount of I/O performance. Software-based I/O virtualization methods use emulation of the I/O devices.

With this emulation layer the VMM provides a consistent view of a hardware device to the VMs, and the device can be shared among many VMs. However, it could also slow down the I/O performance of high I/O performance devices. In contrast, VT-d can address loss of

native performance or of native capability of a virtualized I/O device by directly assigning the device to a VM.

In this model, the VMM restricts itself to a controlling function for enabling direct assignment of devices to its partitions. Rather than invoking the VMM for all (or most) I/O requests from a partition, the VMM is invoked only when guest software accesses protected resources (such as I/O configuration accesses, interrupt management, and so on) that impact system functionality and isolation.

To support direct VM assignment of I/O devices, a VMM must enforce isolation of DMA requests. I/O devices can be assigned to domains, and the DMA remapping hardware can be used to restrict DMA from an I/O device to the physical memory presently owned by its domain.

Virtualization's "crystal ball"

Over time, as hardware virtualization CPU extensions such as Intel's VT-x and VT-d evolve, we

should expect to see the focus on software virtualization techniques diminish.

As hardware-assisted features mature and become less vendor-specific, traditional hypervisors will become more of a commodity utilizing a standard set of features while still competing on performance and functionality.

Features such as software separation, particularly in the military and aerospace market, will be important differentiators. Leading the field in these areas are companies such as LynuxWorks with its secure MILS separation kernel, LynxSecure. This technology affords software separation as well as traditional hypervisor functionality, providing a platform for emerging security standards and protection profiles.



Stuart Fisher is a product manager for LynxSecure at LynuxWorks, Inc. He has more than 15 years of experience in the embedded market, both in engineering roles and customer-interfacing positions. Stuart is based in Sutton Coldfield, England and is a graduate of the University of Coventry, where he earned a Bachelor of Engineering degree in Computing and Electronics.



1.800.255.5969



LynuxWorks, Inc.
855 Embedded Way
San José, CA 95138-1018
408.979.3900
408.979.3920 fax
www.lynuxworks.com

LynuxWorks Europe
50 Broadway
London SW1H 0RG
United Kingdom
+44 208 906 9506
+44 208 906 2338 fax

©2011 LynuxWorks, Inc. LynuxWorks and the LynuxWorks logo are trademarks, and LynxOS and BlueCat are registered trademarks of LynuxWorks, Inc. Linux is a registered trademark of Linus Torvalds. All other trademarks are the trademarks and registered trademarks of their respective owners.

All rights reserved. Printed in the USA.