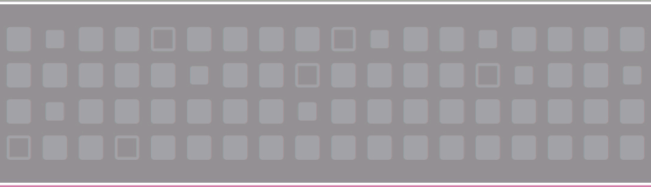
WHITE PAPER

Understanding Full Virtualization, Paravirtualization, and Hardware Assist



vmware®

Contents

Introduction

In 1998, VMware figured out how to virtualize the x86 platform, once thought to be impossible, and created the market for x86 virtualization. The solution was a combination of binary translation and direct execution on the processor that allowed multiple guest OSes to run in full isolation on the same computer with readily affordable virtualization overhead.

The savings that tens of thousands of companies have generated from the deployment of this technology is further driving the rapid adoption of virtualized computing from the desktop to the data center. As new vendors enter the space and attempt to differentiate their products, many are creating confusion with their marketing claims and terminology. For example, while hardware assist is a valuable technique that will mature and expand the envelope of workloads that can be virtualized, paravirtualization is not an entirely new technology that offers an "order of magnitude" greater performance.

While this is a complex and rapidly evolving space, the technologies employed can be readily explained to help companies understand their options and choose a path forward. This white paper attempts to clarify the various techniques used to virtualize x86 hardware, the strengths and weaknesses of each, and VMware's community approach to develop and employ the most effective of the emerging virtualization techniques. Figure 1 provides a summary timeline of x86 virtualization technologies from VMware's binary translation to the recent application of kernel paravirtualization and hardware-assisted virtualization.

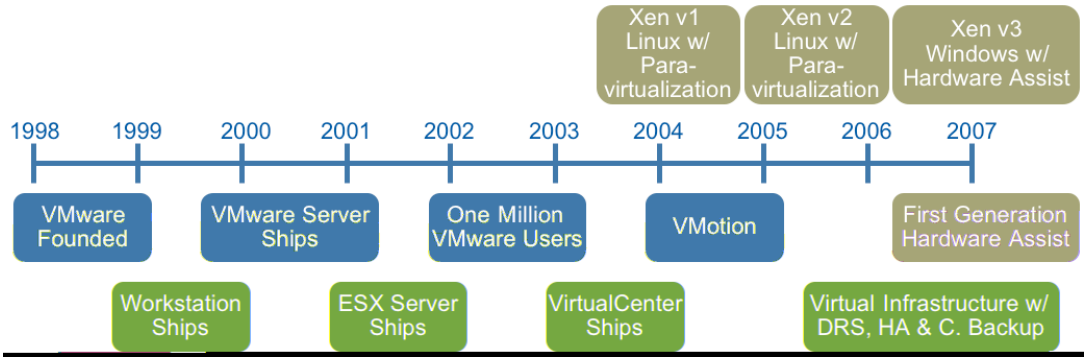


Figure 1 – Summary timeline of x86 virtualization technologies

Overview of x86 Virtualization

The term virtualization broadly describes the separation of a service request from the underlying physical delivery of that service. With x86 computer virtualization, a virtualization layer is added between the hardware and operating system as seen in Figure 2. This virtualization layer allows multiple operating system instances to run concurrently within virtual machines on a single computer, dynamically partitioning and sharing the available physical resources such as CPU, storage, memory and I/O devices.

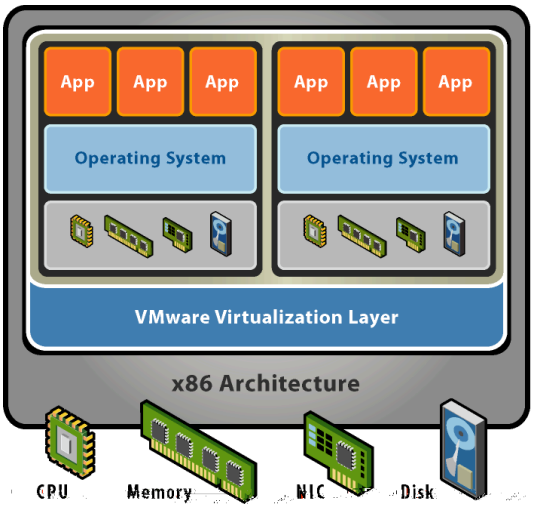


Figure 2 – x86 virtualization layer

As desktop and server processing capacity has consistently increased year after year, virtualization has proved to be a powerful technology to simplify software development and testing, to enable server consolidation, and to enhance data center agility and business continuity. As it turns out, fully abstracting the operating system and applications from the hardware and encapsulating them into portable virtual machines has enabled virtual infrastructure features simply not possible with hardware alone. For example, servers can now run in extremely fault tolerant configurations on virtual infrastructure 24x7x365 with no downtime needed for backups or hardware maintenance. VMware has customers with production servers that have been running without downtime for over three years.

For industry standard x86 systems, virtualization approaches use either a hosted or a hypervisor architecture. A hosted architecture installs and runs the virtualization layer as an application on top of an operating system and supports the broadest range of hardware configurations. In contrast, a hypervisor (bare-metal) architecture installs the virtualization layer directly on a clean x86-based system. Since it has direct access to the hardware resources rather than going through an operating system, a hypervisor is more efficient than a hosted architecture and delivers greater scalability, robustness and performance. VMware Player, ACE, Workstation and Server employ a hosted architecture for flexibility, while ESX Server employs a hypervisor architecture on certified hardware for data center class performance.

To better understand the techniques employed for x86 virtualization, a brief background on the component parts is useful. The virtualization layer is the software responsible for hosting and managing all virtual machines on virtual machine monitors (VMMs). As depicted in Figure 3Figure 3, the virtualization layer is a hypervisor running directly on

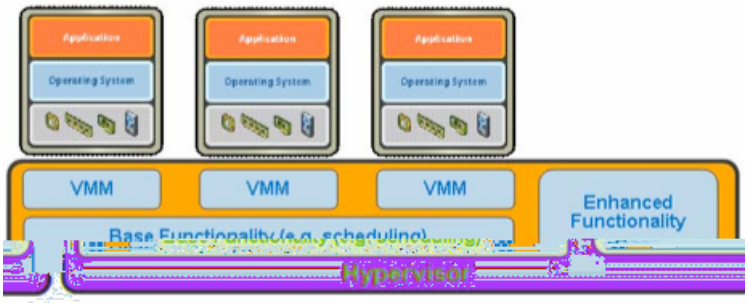


Figure 3 – The hypervisor manages virtual machine monitors that host virtual machines

the hardware. The functionality of the hypervisor varies greatly based on architecture and implementation. Each VMM running on the hypervisor implements the virtual machine hardware abstraction and is responsible for running a guest OS. Each VMM has to partition and share the CPU, memory and I/O devices to successfully virtualize the system.

CPU Virtualization

The Challenges of x86 Hardware Virtualization

X86 operating systems are designed to run directly on the bare-metal hardware, so they naturally assume they fully 'own' the computer hardware. As shown in Figure 4, the x86 architecture offers four levels of privilege known as Ring 0, 1, 2 and 3 to operating systems and applications to manage access to the computer hardware. While user level applications typically run in Ring 3, the operating system needs to have direct access to the memory and hardware and must execute its privileged instructions in Ring 0. Virtualizing the x86 architecture requires placing a virtualization layer under the operating system (which expects to be in the most privileged Ring 0) to create and manage the virtual machines that deliver shared resources. Further complicating the situation, some sensitive instructions can't effectively be virtualized as they have different semantics when they are not executed in Ring 0. The difficulty in trapping and translating these sensitive and privileged instruction requests at runtime was the challenge that originally made x86 architecture virtualization look impossible.

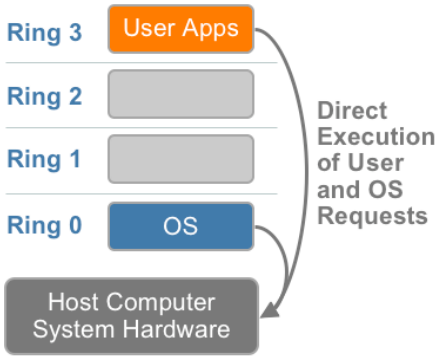


Figure 4 – x86 privilege level architecture without virtualization

VMware resolved the challenge in 1998, developing binary translation techniques that allow the VMM to run in Ring 0 for isolation and performance, while moving the operating system to a user level ring with greater privilege than applications in Ring 3 but less privilege than the virtual machine monitor in Ring 0. While VMware's full virtualization approach using binary translation is the de facto standard today based on VMware's 20,000 customer installed base and large partner ecosystem, the industry as a whole has not yet agreed on open standards to define and manage virtualization. Each company developing virtualization solutions is free to interpret the technical challenges and develop solutions with varying strengths and weaknesses.

As clarified below, three alternative techniques now exist for handling sensitive and privileged instructions to virtualize the CPU on the x86 architecture:

- Full virtualization using binary translation
- OS assisted virtualization or paravirtualization
- Hardware assisted virtualization (first generation)

Technique 1 – Full Virtualization using Binary Translation

VMware can virtualize any x86 operating system using a combination of binary translation and direct execution techniques. This approach, depicted in Figure 5, translates kernel code to replace nonvirtualizable instructions with new sequences of instructions that have the intended effect on the virtual hardware. Meanwhile, user level code is directly executed on the processor for high performance virtualization. Each virtual machine monitor provides each Virtual Machine with all the services of the physical system, including a virtual BIOS, virtual devices and virtualized memory management.

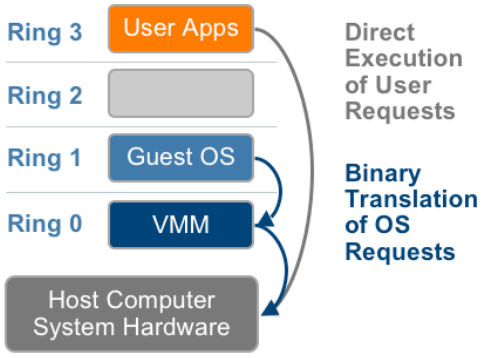


Figure 5 – The binary translation approach to x86 virtualization

This combination of binary translation and direct execution provides Full Virtualization as the guest OS is fully abstracted (completely decoupled) from the underlying hardware by the virtualization layer. The guest OS is not aware it is being virtualized and requires no modification. Full virtualization is the only option that requires no hardware assist or operating system assist to virtualize sensitive and privileged instructions. The hypervisor translates all operating system instructions on the fly and caches the results for future use, while user level instructions run unmodified at native speed.

Full virtualization offers the best isolation and security for virtual machines, and simplifies migration and portability as the same guest OS instance can run virtualized or on native hardware. VMware's virtualization products and Microsoft Virtual Server are examples of full virtualization.

Technique 2 – OS Assisted Virtualization or Paravirtualization

"Para-" is an English affix of Greek origin that means "beside," "with," or "alongside." Given the meaning "alongside virtualization," paravirtualization refers to communication between the guest OS and the hypervisor to improve performance and efficiency. Paravirtualization, as shown in Figure 6, involves modifying the OS kernel to replace non-virtualizable instructions with hypercalls that communicate directly with the virtualization layer hypervisor. The hypervisor also provides hypercall interfaces for other critical kernel operations such as memory management, interrupt handling and time keeping.

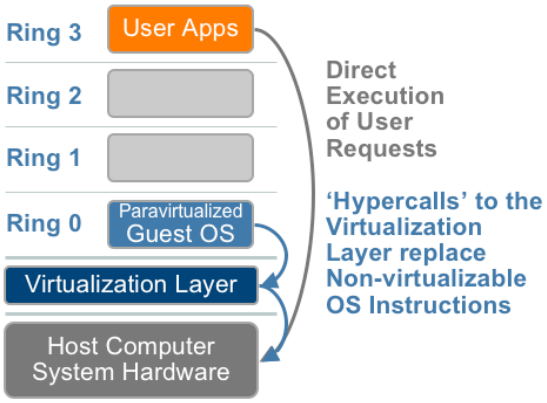


Figure 6 – The Paravirtualization approach to x86 Virtualization

Paravirtualization is different from full virtualization, where the unmodified OS does not know it is virtualized and sensitive OS calls are trapped using binary translation. The value proposition of paravirtualization is in lower virtualization overhead, but the performance advantage of paravirtualization over full virtualization can vary greatly depending on the workload. As paravirtualization cannot support unmodified operating systems (e.g. Windows 2000/XP), its compatibility and portability is poor. Paravirtualization can also introduce significant support and maintainability issues in production environments as it requires deep OS kernel modifications. The open source Xen project is an example of paravirtualization that virtualizes the processor and memory using a modified Linux kernel and virtualizes the I/O using custom guest OS device drivers.

While it is very difficult to build the more sophisticated binary translation support necessary for full virtualization, modifying the guest OS to enable paravirtualization is relatively easy. VMware has used certain aspects of paravirtualization techniques across the VMware product line for years in the form of VMware tools and optimized virtual device drivers. The VMware tools service provides a backdoor to the VMM Hypervisor used for services such as time synchronization, logging and guest shutdown. Vmxnet is a paravirtualized I/O device driver that shares data structures with the hypervisor. It can take advantage of host device capabilities to offer improved throughput and reduced CPU utilization. It is important to note for clarity that the VMware tools service and the vmxnet device driver are not CPU paravirtualization solutions. They are minimal, non-intrusive changes installed into the guest OS that do not require OS kernel modification. Looking forward, VMware is helping develop paravirtualized versions of Linux to support proofs of concept and product development. Further information is provided later in this paper on page 11.

Technique 3 – Hardware Assisted Virtualization

Hardware vendors are rapidly embracing virtualization and developing new features to simplify virtualization techniques. First generation enhancements include Intel Virtualization Technology (VT-x) and AMD's AMD-V which both target privileged instructions with a new CPU execution mode feature that allows the VMM to run in a new root mode below ring 0. As depicted in Figure 7, privileged and sensitive calls are set to automatically trap to the hypervisor, removing the need for either binary translation or paravirtualization. The guest state is stored in Virtual Machine Control Structures (VT-x) or Virtual Machine Control Blocks (AMD-V). Processors with Intel VT and AMD-V became available in 2006, so only newer systems contain these hardware assist features.

Figure 7 – The hardware assist approach to x86 virtualization

Due to high hypervisor to guest transition overhead and a rigid programming model, VMware's binary translation approach currently outperforms first generation hardware assist implementations in most circumstances. The rigid programming model in the first generation implementation leaves little room for software flexibility in managing either the frequency or the cost of hypervisor to guest transitions[1]. Because of this, VMware only takes advantage of these first generation hardware features in limited cases such as for 64-bit guest support on Intel processors.

Memory Virtualization

Beyond CPU virtualization, the next critical component is memory virtualization. This involves sharing the physical system memory and dynamically allocating it to virtual machines. Virtual machine memory virtualization is very similar to the virtual memory support provided by modern operating systems. Applications see a contiguous address space that is not necessarily tied to the underlying physical memory in the system. The operating system keeps mappings of v t l

a n u

To run multiple virtual machines on a single system, another level of memory virtualization is required. In other words, one has to virtualize the MMU to support the guest OS. The guest OS continues to control the mapping of virtual addresses to the guest memory physical addresses, but the guest OS cannot have direct access to the actual machine memory. The VMM is responsible for mapping guest physical memory to the actual machine memory, and it uses shadow page tables to accelerate the mappings. As depicted by the red line in Figure 8, the VMM uses TLB hardware to map the virtual memory directly to the machine memory to avoid the two levels of translation on every access. When the guest OS changes the virtual memory to physical memory mapping, the VMM updates the shadow page tables to enable a direct lookup. MMU virtualization creates some overhead for all virtualization approaches, but this is the area where second generation hardware assisted virtualization will offer efficiency gains.

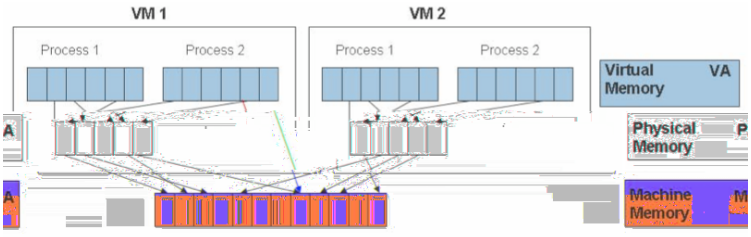


Figure 8 – Memory Virtualization

Device and I/O Virtualization

The final component required beyond CPU and memory virtualization is device and I/O virtualization. This involves managing the routing of I/O requests between virtual devices and the shared physical hardware.

Software based I/O virtualization and management, in contrast to a direct pass-through to the hardware, enables a rich set of features and simplified management. With networking for example, virtual NICs and switches create virtual networks between virtual machines without the network traffic consuming bandwidth on the physical network, NIC teaming allows multiple physical NICS to appear as one and failover transparently for virtual machines, and virtual machines can be seamlessly relocated to different systems using VMotion while keeping their existing MAC addresses. The key to effective I/O virtualization is to preserve these virtualization benefits while keeping the added CPU utilization to a minimum.

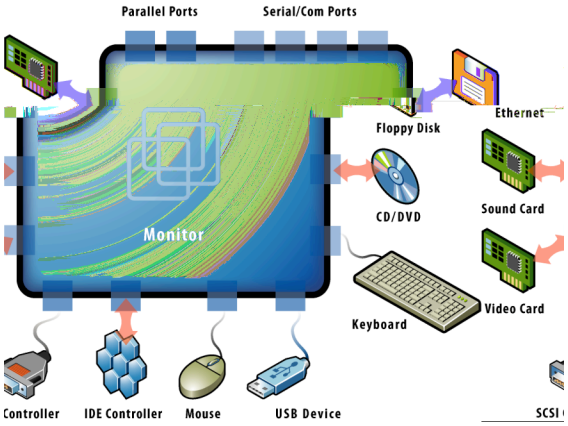


Figure 9 – Device and I/O virtualization

The hypervisor virtualizes the physical hardware and presents each virtual machine with a standardized set of virtual devices as seen in Figure 9. These virtual devices effectively emulate well-known hardware and translate the virtual machine requests to the system hardware. This standardization on consistent device drivers also helps with virtual machine standardization and portability across platforms as all virtual machines are configured to run on the same virtual hardware regardless of the actual physical hardware in the system.

Summarizing the Current State of x86 Virtualization Techniques

VMware is currently utilizing all of these x86 virtualization techniques either in production or in the development lab to deliver the best balance between performance and functionality. With the overview of virtualization techniques covered, the summary comparison in Figure 10 is useful for understanding the high-level strengths and weaknesses of each technique.

	Full Virtualization with Binary Translation	Hardware Assisted Virtualization	OS Assisted Virtualization / Paravirtualization
Technique	Binary Translation and Direct Execution	Exit to Root Mode on Privileged Instructions	Hypercalls
Guest Modification / Compatibility	Unmodified Guest OS Excellent compatibility	Unmodified Guest OS Excellent compatibility	Guest OS codified to issue Hypercalls so it can't run on Native Hardware or other Hypervisors Poor compatibility; Not available on Windows OSes
Performance	Good	Fair Current performance lags Binary Translation virtualization on various workloads but will improve over time	Better in certain cases
Used By	VMware, Microsoft, Parallels	VMware, Microsoft, Parallels, Xen	VMware, Xen
Guest OS Hypervisor Independent?	Yes	Yes	XenLinux runs only on Xen Hypervisor VMI-Linux is Hypervisor agnostic

Figure 10 – Summary comparison of x86 processor virtualization techniques

Full Virtualization with Binary Translation is the Most Established Technology Today

Full Virtualization with Binary Translation is currently the most established and reliable virtualization technology available. VMware's implementation also delivers the highest virtualization performance across commonly deployed Windows and Linux operating systems with the most robust feature set and greatest ease of management. With the exception of 64-bit guests running on Intel CPUs with binary translation, VMware can support both full virtualization and hardware assisted virtualization on production servers with the choice depending on the relative performance.

Full virtualization with binary translation will continue to be a useful technique for years to come as newer and faster hardware continues to advance binary translation performance for unmodified guest OSes, however, hardware assisted virtualization is where virtualization is going with processor paravirtualization being a performance enhancing stopgap along the way.

Hardware Assist is the Future of Virtualization, but the Real Gains Have Yet to Arrive

Intel and AMD's first generation hardware assist features released in 2006 are the first step in removing the need for hypervisors to employ binary translation and OS-assisted processor paravirtualization. As Xen has demonstrated, these early features have made it much easier to build a simple hypervisor without having to address binary translation or paravirtualize the operating system. Xen is using these hardware assist features to virtualize Windows, but with significantly reduced efficiency versus either VMware's binary translation or XenLinux's paravirtualization. The challenge is that these first generation implementations employ a rigid programming model with high hypervisor to guest transition overhead, resulting in lower virtualization performance than VMware's binary translation approach. VMware is working with partners Intel and AMD to enhance future hardware designs to better take advantage of software's inherent flexibility to address virtualization challenges.

Second generation hardware assist technologies are in development that will have a greater impact on virtualization performance while reducing memory overhead. Both AMD and Intel have announced future development roadmaps, including hardware support for memory virtualization (AMD Nested Page Tables [NPT] and Intel Extended Page Tables [EPT]) as well as hardware support for device and I/O virtualization (Intel VT-d, AMD IOMMU).

Compute-intensive workloads already run well with binary translation of privileged instructions and direct execution of non-privileged instructions, but NPT/EPT will provide noticeable performance improvements for memory-remapping intensive workloads by removing the need for shadow page tables that consume system memory. Increased performance and reduced overhead expected in future CPUs will provide motivation to use hardware assist features much more broadly, but don't expect revolutionary improvements. As processors get significantly faster each year, each year's processor performance increases will likely have a greater impact on virtualization capacity and performance than future hardware assist optimizations.

Over time, expect to see hardware assisted virtualization offset paravirtualization's near-term performance benefits related to processor and memory virtualization. As hardware assist is further developed for the CPU, memory and device I/O, the performance benefits of paravirtualization over hardware assisted direct execution will grow smaller. As hardware assist features develop and mature, hypervisors will commoditize as they increasingly leverage a common set of hardware assist features, but they will continue to compete on performance, manageability and features.

Xen's CPU Paravirtualization Delivers Performance Benefits with Maintenance Costs

Xen likes to position paravirtualization as the second generation of virtualization, labeling VMware's full virtualization technology as the first generation. The reality is that paravirtualization is an old and useful virtualization technique that does deliver performance benefits for some workloads but typically with added maintenance costs. No one is disputing the value of installing

virtualization management applications and device drivers into guest operating systems to increase performance, but data centers must weigh the advantage of reduced virtualization overhead versus the maintenance and support costs of running a modified guest OS kernel to enable processor paravirtualization. The performance advantage depends on the nature of the workload. Most user space workloads gain very little, and near native performance is not achieved for all workloads.

Xen's first major challenge is that processor paravirtualization does not apply to unmodified guest OSes, so it is not an option for guests that can't be modified (e.g. Windows) and guests where modifications are undesired (e.g. when supported versions of Linux are required). Most data centers are not willing to run production business applications on a third party paravirtualized Linux kernel running on an open source hypervisor. Additionally, the invasive kernel modifications tightly couple the guest OS to the hypervisor with data structure dependencies, preventing the modified guest OS from running on other hypervisors or native hardware.

Even though major Linux distributions are starting to bundle paravirtualization into the OS kernel, deploying servers using these paravirtualized operating system can increase maintenance costs and reduce portability. As many companies have found XenLinux's paravirtualization approach unsuitable for enterprise use, many new virtualization vendors with open source Xen-derived offerings are abandoning Linux paravirtualization entirely. Virtual Iron, for example, has declared paravirtualization a "dead-end approach[2]" and is focusing entirely on full virtualization of unmodified guest operating systems using hardware assist.

This leads to Xen's second competitive challenge. Xen 3.x only recently introduced support for hardware assisted full virtualization of unmodified guest OSes (e.g. Windows). As VMware's binary translation is far more sophisticated and higher performing than Xen's employment of first generation hardware assist, Xen vendors can't compete with VMware's overall performance, reliability, and ease of management. Those that haven't given up on processor paravirtualization entirely often try to confuse the issue by implying that their Linux paravirtualization performance benefits carry over to unmodified guest OSes that use no processor paravirtualization and require hardware assist for virtualization.

To be clear, VMware does find processor paravirtualization to increase performance significantly on some workloads today, but the longer term performance delta when second generation hardware assist features are available is unclear. The performance difference may be reduced, eliminated, or expanded as enhancements to the paravirtualization interface may create new opportunities. It's an open question.

As VMware sees it, the major problem with processor paravirtualization is the need for guest OS modification that makes it dependent on a specific hypervisor to run. The Xen interface, for example, implements deep paravirtualization with strong hypervisor dependency. The OS kernel is closely tied to structures in the hypervisor implementation. This creates an incompatibility as the XenLinux kernel can't run on native hardware or other hypervisors, doubling the number of kernel distributions that have to be maintained. Additionally, it's limited to newer, open source operating systems as the intrusive changes to the guest OS kernel require OS vendor support. Finally, the strong hypervisor dependency impedes the independent evolution of the kernel.

[2] http://www.virtualiron.com/fusetalk/blog/blogpost.cfm?catid=3&threadid=10

VMware's Transparent Paravirtualization Balances Performance Benefits with Maintenance Costs

While Xen has implemented paravirtualization that is deeply rooted and intrusive to the Linux kernel, VMware has been working on a standard interface to gain the performance advantages of OS assisted paravirtualization while mitigating the maintenance costs. In 2005, VMware proposed a transparent paravirtualization interface, the Virtual Machine Interface (VMI), as a standardized communication mechanism between the guest operating system and the hypervisor.

Figure 11 – Transparent paravirtualization with the Virtual Machine Interface (VMI)

As depicted in Figure 11, the Virtual Machine Interface is a layer between the hypervisor and the paravirtualized guest OS. Transparent paravirtualization is delivered as the same guest OS kernel can run either natively on the hardware or virtualized on any compatible hypervisor. It works by design as all VMI calls have two implementations, inline native instructions for bare hardware and indirect calls to a layer between the guest OS and hypervisor in a virtual machine. Both implementations deliver high performance. Maintainability and extensibility is achieved as VMI versioning can allow independent development of the hypervisor and guest OS. The VMI layer also supports hypervisor diversity as VMI-Linux can run using an appropriate indirection layer on the Xen hypervisor. Portability is subjective, but a succession of Linux versions have been ported to the VMI interface.

VMware is continuing its collaboration with the Linux community to develop a paravirtualization interface that supports multiple hypervisors. In 2006, VMware released the VMI specification as an open specification, and the VMI proposal at the Ottawa Linux Symposium led to the development of the paravirt-ops interface in the Linux community. The paravirt-ops interface, which was developed by a joint team from IBM, VMware, Red Hat, and XenSource, incorporates many of the concepts of VMI including the support of transparent paravirtualization. Using this interface, a paravirtualized Linux operating system will be able to run on any hypervisor that supports it. VMware is actively working on the paravirt-ops guest OS interface which can use the VMI interface to a hypervisor, and paravirt-ops became a part of the official Linux kernel

demonstrate improved performance on CPU intensive workloads. Subsequent implementations on a bare-metal hypervisor architecture, such as the VMware ESX Server, will demonstrate improved CPU and I/O performance due to paravirtualization. VMware is planning to add support for paravirtualized operating systems as they become adopted in commercial operating system distributions across its virtual infrastructure platform products.

VMware is Fostering an Open Standards Approach to Virtualization

For the past several years, VMware has been collaborating with a group of leading technology vendors to define open virtualization standards. As an initial step, VMware has contributed its existing frameworks and APIs to facilitate the development of these standards in an industry neutral manner. VMware is proposing these open interfaces and formats because the most successful interfaces and formats in the technology business have been based on de facto customer deployed standards. VMware's technology has been deployed widely for over seven years and incorporates a significant amount of real-world experience.

In every industry, open interfaces and formats have proven to be a critical enabler to accelerating ubiquitous adoption, and virtualization is no different. As great as the momentum is today for virtualization, it is still in its early stages of adoption. VMware has taken this step to spur the growth of virtualization, accelerate solution delivery to customers and achieve widespread virtualization adoption.

VMware is also taking this step because partners and customers have asked for it. Open interfaces and formats benefit customers by providing access to a broader range of virtualization solutions that are compatible across an increased number of products. Open interfaces and formats benefit the industry by facilitating greater collaboration and innovation across an ecosystem of virtualization vendors to expand the market opportunities for all.

VMware has made available the following open interfaces and formats:

- **Virtual Machine Interface** — APIs between hypervisors and guest operating systems.
- **Management Interface** — Framework that governs the standardized operation and management of stand-alone virtual machine environments as well as highly dynamic, data center scale deployment of virtualized systems.
- **Virtual Machine Disk Format** — Virtual machine disk formats that enable virtual machine provisioning, migration and maintenance across platforms.

VMware intends this to be an open, vendor neutral effort. Any vendor that shares in the common goal of open virtualization standards can participate.

VMware Leverages a Multi-Mode VMM Architecture for Performance and Flexibility

Most startup virtualization vendors only have the resources to build a product that leverages a single strategy for virtualization. They naturally have a motivation to focus on the strengths of their virtualization approach and minimize their approach’s weaknesses and feature gaps. This tends to generate market confusion as companies make one-sided claims that distort reality.

With each product release, VMware employs the combination of these and future virtualization technologies that strikes the most effective balance between performance, stability, functionality and ease of management. VMware also proactively works with partners to develop an interoperable ecosystem for enterprise computing virtualization.

VMware offers a flexible "multi-mode" VMM architecture depicted in Figure 12 that enables a separate VMM to host each virtual machine. VMware allows you to select the mode that achieves the best workload-specific performance based on the CPU support available. The same VMM architecture is used for ESX Server, Player, Server, Workstation and ACE.

While today's workloads can employ a 32-bit BT VMM or a 64-bit VMM with BT or VT-x, tomorrow's workloads will be hosted on VMMs that support 32 and 64-bit versions of AMD-V + NPT and VT-x + EPT.

VMware provides a flexible architecture to support emerging virtualization technologies. Multi-mode VMM utilizes binary translation, hardware assist and paravirtualization to select the best operating mode for each workload and processor combination. Hardware assist will continue to mature and broaden the workloads that can be readily virtualized.

Figure 12 – Multi-mode VMM architecture

Conclusion

There is a range of strategies that VMware is pursuing to improve virtualization performance over time. Binary translation, hardware assist and OS assist (parq 0 0.000059 612 9

Revision: 20070911 Item: WP-028-PRD-01-01