Document Type: Tutorial
NI Supported: Yes
Publish Date: Oct 30, 2009

Virtualization Technology Under the Hood

Overview

It is important for engineers evaluating virtualization technology to understand how virtualization software works. This white paper explains the various types of virtualization software that are available, and discusses the various hardware and software techniques that enable virtualization.

Table of Contents

Introduction

Virtualization is a technology that many engineers are evaluating with the goals of lowering cost, reducing footprint, and creating better integrated systems. In order to make informed decisions on the use of virtualization in measurement and automation applications, it is important to understand the basic virtualization architectures available and the technologies that make them work behind the scenes. This paper will present an overview of this information and help readers form a virtualization plan for their engineering systems.

It is assumed that the reader has a basic understanding of virtualization technology and its benefits. For a review of these topics, please visit the Virtualization Basics white paper.

Virtualization Architectures

Engineering applications can make use of two major virtualization architectures, hosted and bare-metal. Each architecture has different implications on I/O access, determinism, and ease of use that should be considered before building a virtualized system. This section will present an overview of each architecture, its mechanism for communicating with I/O devices (since this is a major factor in engineering applications), and other benefits and drawbacks.

Hosted

Overview

In this architecture, a base operating system (such as Windows) is first installed. A piece of software called a hypervisor or virtual machine monitor (VMM) is installed on top of the host OS, and allows users to run various guest operating systems within their own application windows. Common products that use this architecture today include VMWare Workstation and Parallels Desktop for Mac. A diagram of the hosted virtualization architecture is shown below.

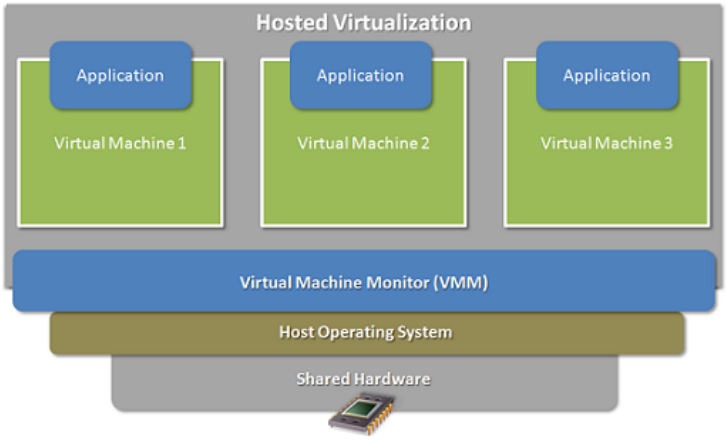

Figure 1. Hosted virtual machine monitors install on top of an underlying host operating system (OS).

I/O Access

In the hosted virtualization architecture, each virtual machine (guest operating system) commonly only has access to a limited subset of I/O devices. The host operating system retains ownership of the physical I/O connected to a given computer, and the virtual machine monitor (VMM) provides an emulated view of the actual hardware (when possible) to each virtual machine (VM). Because the VMM does not have knowledge of most non-generic I/O devices such as PCI data acquisition cards, it does not present these emulated devices to VMs. Only generic devices like network interface cards and CD-ROM drives are emulated.

In addition, many hosted virtualization solutions support passthrough functionality for the USB port. This feature allows users to access USB devices from individual VMs directly, providing for limited I/O capabilities beyond the emulated devices mentioned above. For example, it may be possible (using hosted virtualization software) to access NI USB data acquisition devices from a guest OS and acquire data.

In actuality, several software components work together to make I/O possible in a hosted virtualization architecture. For example, the VMWare Workstation product directs I/O requests from virtual machines through a low-level VMM component, then through a driver, and finally to a user-level application component called VMApp. In the end, the VMApp component passes I/O requests through the host operating system. The key point to remember is that I/O requests are ultimately passed through the host OS in a hosted virtualization architecture. A diagram of this process is shown below.

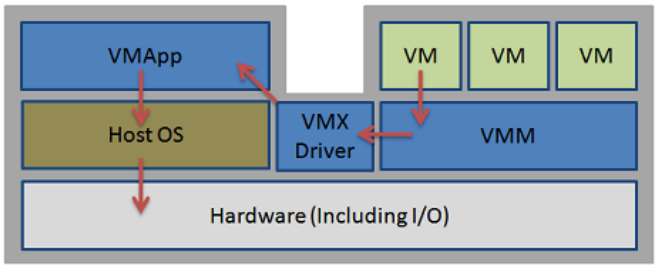

Figure 2. Hosted virtual machine monitor (VMM) software is typically made up of different components that all communicate to channel virtual machine (VM) I/O requests through the host operating system (OS).

Benefits and Drawbacks

One benefit of using a hosted virtualization architecture is ease of installation and configuration. For example, the VMWare Workstation software can be set up in minutes by running a basic installer in Windows. Once installed, an engineer can create several virtual machines that run different operating systems – all on the same physical computer. In addition, VMMs that use hosted virtualization commonly run on a wide variety of PCs. Since a host operating system provides drivers for communicating with low-level hardware, VMM software can be installed on most computers without customization.

As mentioned above, hosted virtualization architectures are not capable of emulating or providing passthrough to many PCI I/O devices. In addition, since I/O requests from virtual machines must be directed through a host OS, performance can be degraded. Another drawback to hosted virtualization is the lack of support for real-time operating systems. Because the underlying host OS dictates scheduling amongst its applications and the VMM, it is usually not possible to run a real-time OS inside of a VM deterministically when using hosted virtualization.

Use Cases

In light of the benefits that hosted virtualization provides, it is commonly used for testing beta software (to eliminate the need for a dedicated testing machine), or to run legacy applications. Hosted virtualization also provides quick support for running different operating systems on one PC, which can be useful for engineers who need frequent access to various applications written for these different operating systems.

Bare-Metal

Overview

The second common architecture for virtualization is bare-metal. In this architecture, a VMM (also called hypervisor) is installed that communicates directly with system hardware rather than relying on a host operating system. See the illustration below for a graphical view of this architecture.

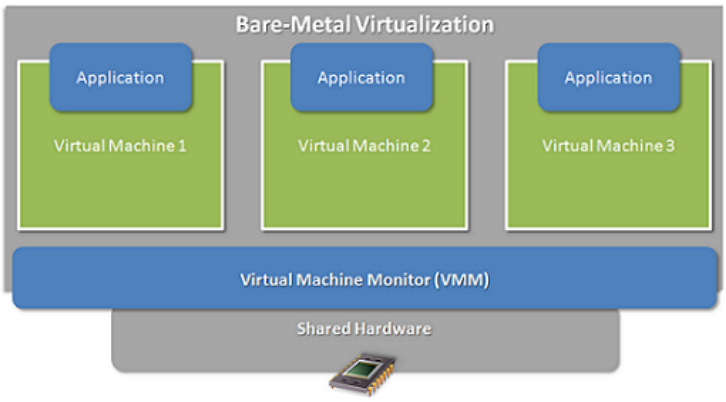

Figure 3. Bare-metal virtual machine monitor software is installed directly on system hardware.

I/O Access

Bare-metal virtualization solutions provide a number of options for I/O access from virtual machines. The reader should note that because bare-metal virtualization does not rely on a host operating system, a hypervisor using this architecture can communicate with I/O devices directly. For I/O devices to be shared between virtual machines (e.g. ethernet, hard drives, etc.), the hypervisor software must contain a low-level driver to communicate with the device. It must also be able to emulate each shared device for guest virtual machines.

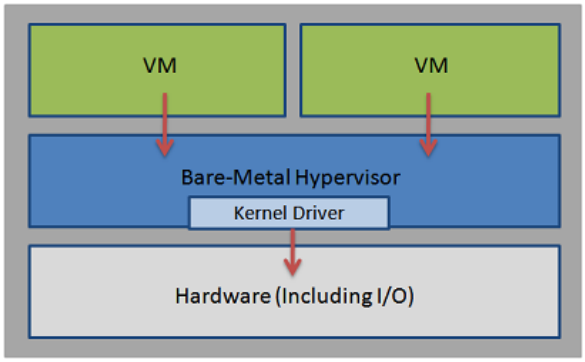

Figure 4. In the bare-metal virtualization architecture, system I/O devices that are shared between virtual machines must be accessed by a kernel driver that is part of the hypervisor.

Another way that bare-metal hypervisors can approach I/O access is to assign individual devices to specific virtual machines. This is called partitioning, and can greatly improve I/O performance and support for engineering applications. Since partitioned I/O devices (such as PCI data acquisition boards) can be accessed directly from VMs using their native drivers, less intervention by the VMM is needed. A diagram showing direct (partitioned) I/O access in a bare-metal virtualization architecture is shown below.

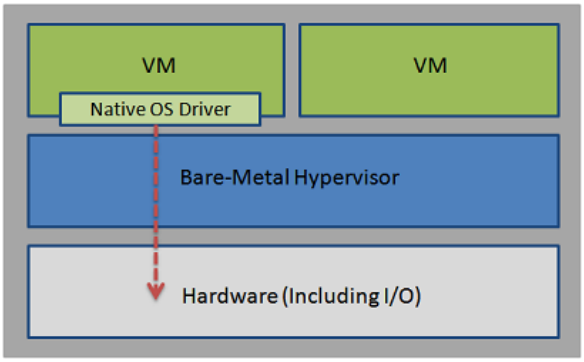

Figure 5. Bare-Metal virtualization software can enable I/O devices to be partitioned amongst individual virtual machines for direct access.

Benefits and Drawbacks

In addition to the improved I/O performance possible when partitioning devices such as PCI data acquisition boards between individual VMs, bare-metal virtualization architectures have the benefit of supporting real-time operating systems. Since they do not rely on an underlying host operating system, bare-metal hypervisors can implement features to bound interrupt latency and enable deterministic performance. This means that engineers using bare-metal virtualization can run real-time and general purpose operating systems in parallel on the same processing hardware.

Bare-metal virtualization does, however, have some drawbacks that should be considered. Any drivers needed to support various hardware platforms must be included in the hypervisor, in addition to drivers for devices that will be shared amongst virtual machines. Furthermore, since bare-metal hypervisors are not installed on top of a host OS, they are typically more difficult to install and configure than a hosted solution.

Use Cases

The low-level nature of bare-metal hypervisors and the I/O access they provide makes them useful for deployed applications that use multiple operating systems. Specifically, applications that must provide real-time data processing and provide access to general purpose OS services (such as a graphical user interface) can benefit from bare-metal virtualization.

Summary

When implementing a system that uses virtualization, a first step should be to determine whether a hosted or bare-metal architecture makes more sense (or a combination of both). Overall, hosted virtualization can provide many benefits during the development process including reducing the cost of beta testing software, running legacy applications, and supporting applications written for different operating systems. The limited I/O capabilities provided in most hosted virtualization solutions should be considered carefully when designing deployed applications.

On the other hand, bare-metal virtualization provides several benefits such as real-time OS support and increased I/O access with partitioning. Integrating a bare-metal hypervisor into an application, however, may require additional installation work and careful consideration of underlying hardware support.

Virtualization Techniques

Thus far in this paper, VMM software has been treated as a black box that supports running multiple operating systems in parallel. This section will introduce several of the underlying techniques that may be used by a VMM to isolate individual virtual machines from computer hardware. Note that software solutions using either of the virtualization architectures mentioned above may use any one or more of these techniques.

It should be noted that all of the techniques mentioned have the same end goal: to intercept any virtual machine instruction that could affect system state (shared resources) in any way. Each technique is simply a different way of achieving this goal.

Binary Translation

VMMs that use binary translation dynamically alter code that is executing to avoid affecting system state. Any time a virtual machine's compiled code contains a privileged instruction (e.g. accessing an I/O device), the underlying VMM can use binary translation to appropriately redirect the I/O request and prevent conflicts between individual VMs. Hosted virtualization software typically makes use of binary translation; one example is the VMWare Workstation product previously mentioned.

Because performance is degraded whenever code must be translated (any switch from a virtual machine to the VMM), virtualization software that uses binary translation typically inspects and translates groups of instructions at a time. By minimizing the amount of times that the VMM must interfere with virtual machine execution, software using binary translation can minimize the performance impact a user experiences.

Hardware Assist

Rather than modifying VM code as it is running, the hardware assist approach uses special processor technology to avoid changing system state upon privileged instructions. Both Intel and AMD have started including new virtualization features in their processors (called Intel-VT and AMD-V) to automatically call a VMM when necessary. Many bare-metal hypervisors make use of this technology.

As is the case with binary translation, performance can be degraded each time that a virtual machine's execution must be interrupted by the hypervisor. To minimize this impact, processors that incorporate hardware features for virtualization can be configured to only interrupt VM execution when absolutely necessary. For example, if a specific I/O devices is partitioned then it may be accessed by its assigned VM without hypervisor intervention.

Paravirtualization

One final technique for accomplishing virtualization is called paravirtualization. This method involves explicitly modifying an operating system so that it is "aware" of being virtualized and automatically calls the underlying VMM when necessary. These calls are commonly referred to as hypercalls. While paravirtualization can improve performance greatly by minimizing the number of times that a VMM must be called, it requires gaining access to OS source code so that modifications can be made.

Conclusion

When incorporating virtualization into a measurement or automation application, either hosted or bare-metal VMMs can be used. Each architecture has different implications on I/O, ease of installation, determinism, and other factors that must be considered. At a low level, these architectures depend on techniques such as binary translation, hardware assist, and paravirtualization to accomplish virtualization.

As virtualization becomes more common in the engineering world, it is increasingly important for engineers to understand virtualization technology, the basic types of virtualization, and the underlying techniques that make it possible. By making informed decisions, designers can optimize the performance of virtualized systems while balancing I/O and other needs.

Additional Resources

For more information on both virtualization and multicore technologies, visit the following resources.

Webcast: Introduction to Virtualization

White Paper: Virtualization Basics

Multicore Programming Resources