

Fault Tolerance for Virtual Environments – Part 2

April 2008

In Part 1 of this series, we described how virtualization can significantly reduce the capital and operational costs of a data center. It does this by creating several *virtual machines* on a single physical server. Each virtual machine runs its own operating system, known as a *guest operating system*, such as Linux, Windows, or Unix. Each operating system thinks that it is running in its own virtual server.

The physical layer requests made by the guest operating systems are adjudicated by an intervening layer, the *hypervisor*. The hypervisor, in effect, multiplexes the requests from the operating system and allows only one request at a time to be passed to the physical server.

Consequently, the number of physical servers required by a data center can be reduced, in some cases dramatically. This reduction in physical hardware is accompanied by associated reduction in capital and operating costs. Fewer machines mean less space, less air conditioning, less power, less power backup, less maintenance, and less system administration.

In Part 2 of this series, we look at the various architectures that are being used to provide virtualization.

Virtualization Architectures

Virtualization requires a virtualization layer between the virtual machines and the physical machine. The virtualization layer is responsible for multiplexing the access requests of the VMs to the resources of the physical machine. There are two primary ways in which virtualization is implemented in currently available products:

- Operating System Virtualization, in which the virtualization layer sits on top of a *host operating system* that is installed on the physical server. The host operating system provides the interfaces between the virtual environments and the physical processor and its I/O devices.
- Bare-Metal Virtualization, in which the virtualization layer (the hypervisor) sits directly on top of the hardware with no intervening host operating system. The hypervisor in this case provides the common device drivers.

Operating System Virtualization

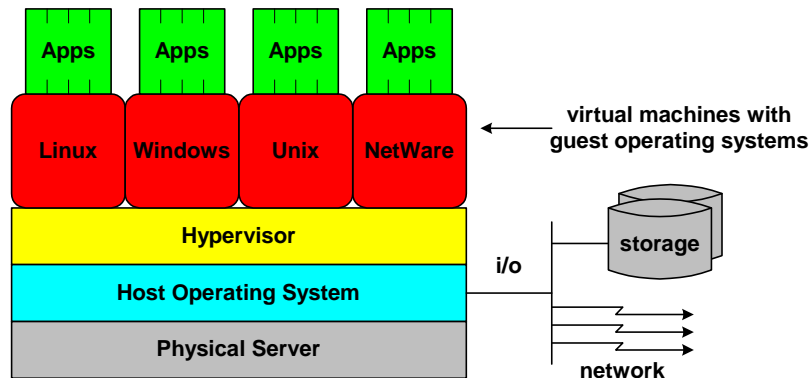
Operating system virtualization emulates the underlying host operating system for its guest operating systems. There are two ways that products create virtual environments on top of a host operating system:

- Virtualization of the Physical Server, in which the host operating system creates virtual machines that provide virtual views of the underlying physical server.

- Virtualization of the Host Operating System, in which the host operating system creates virtual views of itself (also called *containerization*).

Virtualization of the Physical Server

With this technique for virtualization, a host operating system is installed on the physical server. A virtualization layer, the hypervisor, is then installed on top of the operating system. On top of the hypervisor are created several virtual machines. Each VM can run a guest operating system. The guest operating systems may be a mix of any operating systems supported by the product, and each runs its own applications..



Virtualization with a Host OS

The hypervisor traps calls from the guest operating systems to the physical server and multiplexes access to the physical layer through the host operating system so that each virtual machine thinks that it has sole access to the physical server. The I/O devices that a virtual machine can access are those supported by the host operating system.

When a virtual machine is created, it appears as an empty physical server with a blank disk. The disk must first be formatted, and then a guest operating system must be installed. Thereafter, applications may be installed as if this were a stand-alone server running the guest operating system.

Therefore, once the virtualized system is configured and virtual machines are created, the administrative procedures are identical to those of separate physical servers. No special knowledge of virtualization is needed of the application administrators. This includes the installation of the operating system, the installation and management of the applications, and the management of data storage, whether used for files or for a relational database.

One concern with this method of virtualization is that if the host operating system should crash, all virtual machines running on the physical server are taken down. Since operating system crashes tend to be more frequent than physical server crashes, this creates an availability issue.

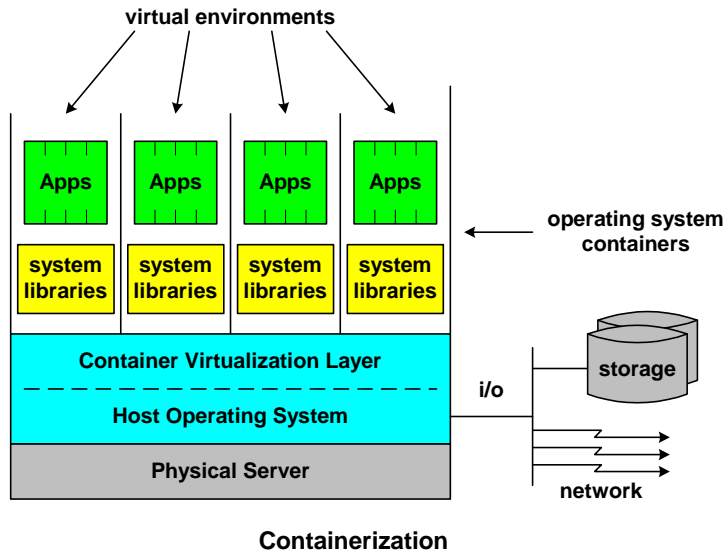
An example of a product that creates virtual machines on top of a host operating system is the GSX Server from VMware (www.vmware.com). The GSX server can run on any physical server that is x86-based or that runs the Pentium instruction set. The host operating system can be any one of a number of versions of Windows or Linux. Guest operating systems can be any mix of a wide variety of versions of Windows, Linux, Unix, and NetWare.

Virtualization of the Host Operating System

This technique is also known as *containerization*. As opposed to creating virtual views of the underlying physical server, containerization provides virtual views of the underlying host operating

system. Rather than creating virtual machines, containerization creates independent containers that each provide a *virtual environment* reflecting the underlying operating system.

The host operating system is typically some version of Windows or Linux. A virtualization layer within the operating system creates containers, each containing a set of system libraries called by the applications in each container. The system libraries replace the guest operating system normally installed in a virtual machine.



Application calls to the operating system are intercepted by the system libraries in the container occupied by the application. These libraries emulate the underlying operating system. So far as the applications are concerned, they are interfacing directly with the host operating system as if they had exclusive use of it.

The devices that are available to the applications are the devices supported by the host operating system. Therefore, most of the common devices are generally supported.

One limitation of containerization is that all applications see the same host operating system. Therefore, they must all be configured to run under that operating system. Their configurations must match the host operating system's version and even its patch level. Therefore, pure virtualization is not provided since there is not the freedom to run any version of a guest operating system and its applications on the physical server.

As with operating system virtualization that exposes the physical server, as described above, a crash of the operating system in this case will take down the entire virtualized environment.

An example of a containerization product is Virtuozzo from SWsoft (www.swsoft.com). It supports both Windows and Linux host operating systems. Though the Windows version is reflected in the virtual environment so that all applications must run under that version of Windows, the use of Linux is more flexible in that the applications are not so severely restrained and may run some different versions of Linux. Virtuozzo is also available as open source (OpenVZ).

In addition, Sun's Solaris operating system supports containerization (see www.sun.com/datacenter/consolidation/virtualization).

Bare-Metal Virtualization

Bare-metal virtualization emulates the underlying physical server for its guest operating systems. Bare-metal virtualization does not use an underlying host operating system as with operating system virtualization. Rather, the hypervisor sits directly on top of the physical server (thus the name *bare metal*). There are two forms of bare-metal virtualization in use:

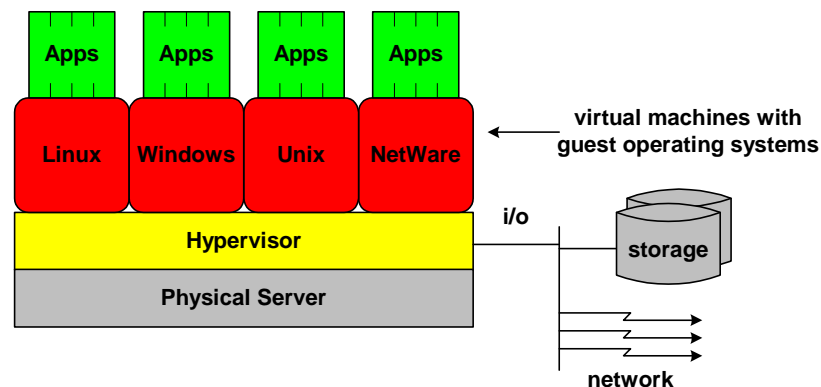
- Bare-Metal Hypervisor, in which a hypervisor sitting on top of the physical server provides multiplexing services to the physical server's resources for the virtual machines. I/O devices are managed by the hypervisor.
- Paravirtualization, in which a bare-metal hypervisor provides access to the physical server's processor and memory but in which access to I/O devices is via a privileged guest operating system running in its own virtual machine.

Bare-metal virtualization basically emulates the underlying microprocessor chip for its virtual machines. It is generally more efficient than operating system virtualization because there is no intervening operating system.

Bare-Metal Hypervisor

A bare-metal hypervisor runs directly on the physical server. It establishes virtual machines by providing an emulated hardware interface to each virtual machine, each of which can be running a different guest operating system. It traps calls made by the guest operating systems and multiplexes them to the physical processor. It handles all device interrupts and device management.

The bare-metal hypervisor can support dissimilar guest operating systems running in its virtual machines. Typically, different versions of Windows, Linux, and Unix, among others, can be running simultaneously in the different virtual machines so long as they are designed to run in an x86 architecture. However, the guest operating systems can only use those device drivers supported by the hypervisor.



Bare-Metal Virtualization

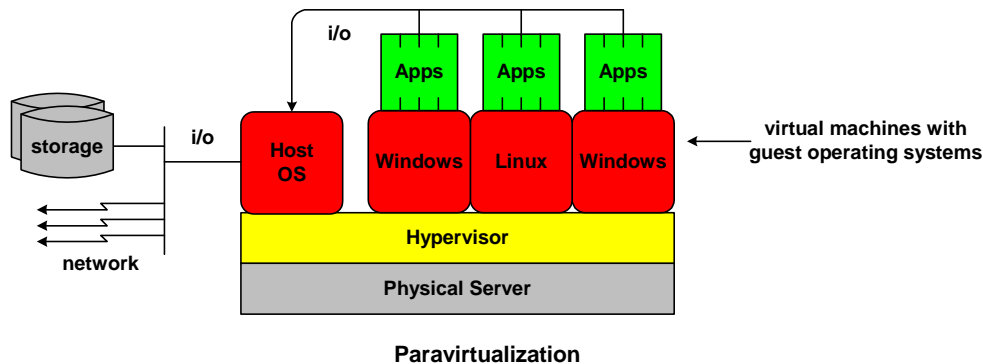
A technical challenge with bare-metal hypervisors is that the calls to server hardware made by guest operating systems must be modified to calls to the hypervisor instead. Rather than requiring special virtualized versions of the operating systems, the hypervisor at runtime modifies the calls in the standard guest operating system to call the hypervisor instead. This process is called *runtime binary translation*. A limitation of this approach is that only those guest operating systems that are supported by the hypervisor's runtime binary translation can be used in the virtual machines.

The leading example of a bare-metal hypervisor is ESX from VMware. ESX supports Windows, Linux, Unix, and Novell NetWare guest operating systems. It is available as an embedded component built into the server hardware. (See http://www.vmware.com/files/pdf/esx_datasheet.pdf).

Paravirtualization

Paravirtualization uses a hypervisor similar to the bare-metal hypervisor described above except that the paravirtualization hypervisor only provides emulation of the underlying physical server's processor and memory. I/O devices are supported via a privileged guest operating system that is running in its own virtual machine.

The device drivers in other guest operating systems are replaced with stubs that communicate via shared memory with stubs in the privileged guest operating system for device access. In effect, the privileged guest operating system acts as a gateway to its devices for other guest operating systems. Any device supported by the privileged guest operating system can be used by guest operating systems running in the other virtual machines.



Paravirtualization brings with it two advantages over the bare-metal hypervisor previously described:

- A wider range of I/O devices can be used since special device drivers do not have to be developed for the hypervisor. Any device supported by the privileged guest operating system can be used.
- Much lower overhead is required to support the virtual machines since the hypervisor does not have to support I/O.

Perhaps the leading example today of paravirtualization is Xen (originally Xen Soft) from Citrix (www.citrixserver.com). Xen runs on a wide variety of hardware platforms and imposes a very small footprint. The entire Xen code base is less than 50,000 lines of code. Xen supports Linux and Windows guest operating systems.

Xen is also available as open source (www.xen.org).

Summary

There are several ways in which virtualization can be achieved. Virtualization hypervisors can be dichotomized as operating system hypervisors or as bare-metal hypervisors. Each has its own advantages, and each is provided by various products in the marketplace.

However, virtualization brings with it a serious problem. It is the “all the eggs in one basket” problem. If a physical server fails, rather than bringing down one application, it can bring down

many applications. Furthermore, should the physical server or a host operating system need to be upgraded, all of these applications may have to suffer a planned outage.

In our next part of this series, we look at the problem of physical server failure and the mechanisms that are provided by current products to recover from such failures.