VIRTUALIZATION DEFINED - EIGHT DIFFERENT WAYS

Introduction

Imagine a typical office...

Alice: Hey Bob, great news! I just heard our IT department is going to implement virtualization in the data center. It's about time we supported virtualization.

Bob: That is great news; virtualization is the wave of the future. What are we virtualizing? The servers? The web applications? The storage network?"

Alice: Hmmm…good question; they didn't specify. But how many different versions of virtualization could there be?

Alice has asked the million-dollar question: What does "going virtual" really mean in today's IT world? Virtualization as a concept is not new; computational environment virtualization has been around since the first mainframe systems. But recently, the term "virtualization" has become ubiquitous, representing ***any*** type of process obfuscation where a process is somehow removed from its physical operating environment. Because of this ambiguity, virtualization can almost be applied to any and all parts of an IT infrastructure. For example, mobile device emulators are a form of virtualization because the hardware platform normally required to run the mobile operating system has been emulated, removing the OS binding from the hardware it was written for. But this is just one example of one type of virtualization; there are many definitions of the term "virtualization" floating around in the current lexicon, and all (or at least most) of them are correct, which can be quite confusing.

This paper focuses on virtualization as it pertains to the data center; but before considering any type of data center virtualization, it's important to define what technology or category of service you're trying to virtualize. Generally speaking, virtualization falls into three categories: Operating System, Storage, and Applications. But these categories are very broad and don't adequately delineate the key aspects of data center virtualization. It's helpful to distill these broader categories into eight, specific categories to thoroughly understand the differences (and similarities) between the definitions of virtualization.

Operating System Virtualization

The most prevalent form of virtualization today, virtual operating systems (or virtual machines) are quickly becoming a core component of the IT infrastructure. Generally, this is the form of virtualization end-users are most familiar with. Virtual machines are typically full implementations of standard operating systems, such as Windows Vista or RedHat Enterprise Linux, running simultaneously on the same physical hardware. Virtual Machine Managers (VMMs) manage each virtual machine individually; each OS instance is unaware that 1) it's virtual and 2) that other virtual operating systems are (or may be) running at the same time. Companies like Microsoft, VMware, Intel, and AMD are leading the way in breaking the physical relationship between an operating system and its native hardware, extending this paradigm into the data center. As the primary driving force, data center consolidation is bringing the benefits of virtual machines to the mainstream market, allowing enterprises to reduce the number of physical machines in their data centers without reducing the number of underlying applications. This trend ultimately saves enterprises money on hardware, co-location fees, rack space, power, cable management, and more.

Application Server Virtualization

Application Server Virtualization has been around since the first load balancer, which explains why "application virtualization" is often used as a synonym for advanced load balancing. The core concept of application server virtualization is best seen with a reverse proxy load balancer: an appliance or service that provides access to many different application services transparently. In a typical deployment, a reverse proxy will host a virtual interface accessible to the end user on the "front end." On the "back end," the reverse proxy will load balance a number of different servers and applications such as a web server. The virtual interface—often referred to as a Virtual IP or VIP—is exposed to the outside world, represents itself as the actual web server, and manages the connections to and from the web server as needed. This enables the load balancer to manage multiple web servers or applications as a single instance, providing a more secure and robust topology than one allowing users direct access to individual web servers. This is a one:many (one-to-many) virtualization representation: one server is presented to the world, hiding the availability of multiple servers behind a reverse proxy appliance. Application Server Virtualization can be applied to any (and all) types of application deployments and architectures, from fronting application logic servers to distributing the load between multiple web server platforms, and even all the way back in the data center to the data and storage tiers with database virtualization.

Application Virtualization

While they may sound very similar, Application Server and Application Virtualization are two completely different concepts. What we now refer to as application virtualization we used to call "thin clients." The technology is exactly the same, only the name has changed to make it more IT-PC (politically correct, not personal computer). Softgrid by Microsoft is an excellent example of deploying application virtualization. Although you may be running Microsoft Word 2007 locally on your laptop, the binaries, personal information, and running state are all stored on, managed, and delivered by Softgrid. Your local laptop provides the CPU and RAM required to run the software, but nothing is installed locally on your own machine. Other types of Application Virtualization include Microsoft Terminal Services and browser-based applications. All of these implementations depend on the virtual application running locally and the management and application logic running remotely.

VIRTUALIZATION DEFINED - EIGHT DIFFERENT WAYS

Management Virtualization

Chances are you already implement administrative virtualization throughout your IT organization, but you probably don't refer to it by this phrase. If you implement separate passwords for your root/administrator accounts between your mail and web servers, and your mail administrators don't know the password to the web server and vise versa, then you've deployed management virtualization in its most basic form. The paradigm can be extended down to segmented administration roles on one platform or box, which is where segmented administration becomes "virtual." User and group policies in Microsoft Windows XP, 2003, and Vista are an excellent example of virtualized administration rights: Alice may be in the backup group for the 2003 Active Directory server, but not in the admin group. She has read access to all the files she needs to back up, but she doesn't have rights to install new files or software. Although she is logging into the same sever that the true administrator is logs into, her user experience differs from the administrator. Management virtualization is also a key concept in overall data center management. It's critical that the network administrators have full access to all the infrastructure gear, such as core routers and switches, but that they not have admin-level access to servers

Network Virtualization

Network virtualization may be the most ambiguous, specific definition of virtualization. For brevity, the scope of this discussion is relegated to what amounts to virtual IP management and segmentation. A simple example of IP virtualization is a VLAN: a single Ethernet port may support multiple virtual connections from multiple IP addresses and networks, but they are virtually segmented using VLAN tags. Each virtual IP connection over this single physical port is independent and unaware of others' existence, but the switch is aware of each unique connection and manages each one independently. Another example is virtual routing tables: typically, a routing table and an IP network port share a 1:1 relationship, even though that single port may host multiple virtual interfaces (such as VLANs or the "eth0:1" virtual network adapters supported by Linux). The single routing table will contain multiple routes for each virtual connection, but they are still managed in a single table. Virtual routing tables change that paradigm into a one:many relationship, where any single physical interface can maintain multiple routing tables, each with multiple entries. This provides the interface with the ability to bring up (and tear down) routing services on the fly for one network without interrupting other services and routing tables on that same interface.

Hardware Virtualization

Hardware virtualization is very similar in concept to OS/Platform virtualization, and to some degree is required for OS virtualization to occur. Hardware virtualization breaks up pieces and locations of physical hardware into independent segments and manages those segments as separate, individual components. Although they fall into different classifications, both symmetric and asymmetric multiprocessing are examples of hardware virtualization. In both instances, the process requesting CPU time isn't aware which processor it's going to run on; it just requests CPU time from the OS scheduler and the scheduler takes the responsibility of allocating processor time. As far as the process is concerned, it could be spread across any number of CPUs and any part of RAM, so long as it's able to run unaffected.

Another example of hardware virtualization is "slicing": carving out precise portions of the system to run in a "walled garden," such as allocating a fixed 25% of CPU resources to bulk encryption. If there are no processes that need to crunch numbers on the CPU for block encryption, then that 25% of the CPU will go unutilized. If too many processes need mathematical computations at once and require more than 25%, they will be queued and run as a FIFO buffer because the CPU isn't allowed to give out more than 25% of its resources to encryption. This type of hardware virtualization is sometimes referred to as pre-allocation.

Asymmetric multiprocessing is a form of pre-allocation virtualization where certain tasks are only run on certain CPUs. In contrast, symmetric multiprocessing is a form of dynamic allocation, where CPUs are interchangeable and used as needed by any part of the management system. Each classification of hardware virtualization is unique and has value, depending on the implementation. Pre-allocation virtualization is perfect for very specific hardware tasks, such as offloading functions to a highly optimized, single-purpose chip. However, pre-allocation of commodity hardware can cause artificial resource shortages if the allocated chunk is underutilized. Dynamic allocation virtualization is a more standard approach and typically offers greater benefit when compared to pre-allocation. For true virtual service provisioning, dynamic resource allocation is important because it allows complete hardware management and control for resources as needed; virtual resources can be allocated as long as hardware resources are still available. The downside to dynamic allocation implementations is that they typically do not provide full control over the dynamicity, leading to processes which can consume all available resources.

Storage Virtualization

As another example of a tried-and-true technology that's been dubbed "virtualization," storage virtualization can be broken up into two general classes: block virtualization and file virtualization. Block virtualization is best summed up by Storage Area Network (SAN) and Network Attached Storage (NAS) technologies: distributed storage networks that appear to be single physical devices. Under the hood, SAN devices themselves typically implement another form of Storage Virtualization: RAID. iSCSI is another very common and specific virtual implementation of block virtualization, allowing an operating system or application to map a virtual block device, such as a mounted drive, to a local network adapter (software or hardware) instead of a physical drive controller. The iSCSI network adapter translates block calls from the application to network packets the SAN understands and then back again, essentially providing a virtual hard drive.

VIRTUALIZATION DEFINED - EIGHT DIFFERENT WAYS

Storage Virtualization - Continued

File virtualization moves the virtual layer up into the more human-consumable file and directory structure level. Most file virtualization technologies sit in front of storage networks and keep track of which files and directories reside on which storage devices, maintaining global mappings of file locations. When a request is made to read a file, the user may think this file is statically located on their personal remote drive, P:\My Files\budget.xls; however, the file virtualization appliance knows that the file is actually located on an SMB server in a data center across the globe at //10.0.16.125/finance/alice/budget-document/budget.xls. File-level virtualization obfuscates the static virtual location pointer of a file (in this case, on Alice's P:\ drive) from the physical location, allowing the back-end network to remain dynamic. If the IP address for the SMB server has to change, or the connection needs to be re-routed to another data center entirely, only the virtual appliance's location map needs to be updated, not every user that needs to access their P:\ drive.

Service Virtualization

And finally, we reach the macro definition of virtualization: service virtualization. Service virtualization is consolidation of all of the above definitions into one catch-all catchphrase. Service virtualization connects all of the components utilized in delivering an application over the network, and includes the process of making all pieces of an application work together regardless of where those pieces physically reside. This is why service virtualization is typically used as an enabler for application availability. For example, a web application typically has many parts: the user-facing HTML; the application server that processes user input; the SOA gears that coordinate service and data availability between each component; the database back-end for user, application, and SOA data; the network that delivers the application components; and the storage network that stores the application code and data. Service virtualization allows each one of the pieces to function independently and be "called up" as needed for the entire application to function properly. When we look deeper into these individual application components, we may see that the web server is load-balanced between 15 virtual machine operating systems, the SOA requests are pushed through any number of XML gateways on the wire, the database servers may be located in one of five global data centers, and so on. Service virtualization combines these independent pieces and presents them together to the user as a single, complete application.

While Service virtualization may encompass all the current definitions of virtualization, it's by no means where IT will stop defining the term. With the pervasive and varied use of the word (as well as the technologies it refers to), there may never be a "final" definition for virtualization; it will continue to evolve and expand as more and more technologies become less and less dependent on rigid operating environments.