Much has happened in the area of virtualization since the topic was first discussed in this column, so this is a good time to revisit (or re-introduce) virtualization and all its flavors to help you determine which type of virtualization could benefit your organization the most, and how to make it the most effective.

In each column, The WebSphere® Contrarian answers questions, provides guidance, and otherwise discusses fundamental topics related to the use of WebSphere products, often dispensing field-proven advice that contradicts prevailing wisdom.

What is virtualization?

It's been nearly three years since I last devoted a column to the topic of virtualization. Given the increased interest and adoption of this technology, I wanted to revisit this topic to, hopefully, provide some new perspectives, while at the same time review some aspects of this topic that haven't changed.

In its most common use today, the term virtualization refers to "an abstraction or a masking of underlying physical resources (such as a server) from operating system images or instances running on the physical resource" which is directly from that earlier article. Further, while recent use of this term typically refers to "server virtualization employing hypervisors," this isn't a new concept. Virtualization actually dates to the 1960s, when it was first employed on IBM® System 360-67 (I'm not young, but no, I didn't work on those machines!).

To provide a bit more background, a hypervisor can be classified into two types:

- **Type 1**, also known as "native" or "bare metal," where the hypervisor is the operating system or it's integral to the operating system. Examples of type 1 hypervisors would be VMware ESX and IBM PowerVM™ to name but two.
- **Type 2** refers to "hosted" or "software applications," where the hypervisor is an application running on the operating system. Some examples include VMware Server, VMware Workstation, and Microsoft® Virtual Server.
Before going on, I should point out that in this context a more precise categorization of the above is server virtualization. Any discussion of virtualization that covered only server virtualization would be incomplete, because you can also virtualize applications, subdividing the application servers and the resource allocation for an application (or portfolio of applications) using application virtualization. An example of an application virtualization technology is IBM WebSphere® Virtual Enterprise.

**Is virtualization for you, and if so what type?**

Chances are it's not of matter of if your enterprise is employing virtualization, but how your enterprise is employing virtualization, as well as how effective virtualization has been.

There are likely many measurements that can be applied to determine virtualization effectiveness, but because server virtualization is often viewed as a mechanism to improve server utilization, utilization would be one measure.

Another related measure is return on investment (ROI), which should increase as utilization increases. A couple of reports (see Resources) over the past few years show mixed results for server virtualization with only single digit increases in server utilization and mixed results in improving ROI. Several factors contribute to these mixed results; in some cases it's a simple lack of an effective process for tracking ROI, but another set of issues also likely contributes. This is because the partitioning of a physical server into multiple operating system images provides platform flexibility, enabling easy operating system re-provisioning or co-location as needs dictate -- and while server virtualization is excellent for server containment that provides consolidated, dedicated, and isolated environments, it also has its drawbacks: server virtualization can be difficult to manage across an enterprise or within a data center because, typically, server virtualization is focused on individual servers.

Moreover, the real gain in increasing utilization comes from managing or distributing the workload intelligently across many servers. It is this latter area that application virtualization can provide real benefit, because application virtualization is "application aware" in terms of application priority, application resource use, and even request and user priority -- all areas that server virtualization lacks. This awareness enables application virtualization (that is, WebSphere Virtual Enterprise) to:

- Direct application-specific requests to the server that can process them the fastest.
- Redirect application requests to another server when a given server reaches the limit of all available resources.
- Ensure that load on one application does not affect the response time of the others, in the case of co-located applications.
- Automatically detect that an error is being returned from an overloaded application and route future application requests to another server capable of running the application.
- Dynamically optimize the resource allocation (CPU and memory) for an application portfolio.

Application virtualization delivers complementary and value added benefits to infrastructures using server virtualization. Both approaches can increase utilization of servers, which in turn leads to server consolidation (and decreased costs). Application virtualization does so by intelligently
managing and optimizing application workload, driving the work to the right place, and controlling the application mixture. Hardware virtualization provides its benefits by dividing a server into multiple pieces.

**Optimizing virtualization delivery**

Related to both server virtualization and application virtualization is a virtualization appliance, which according to Wikipedia is a "minimalist virtual machine image with a pre-installed and pre-configured application (or applications) and operating system environment."

An example technology of a virtualization appliance is the IBM WebSphere Cloudburst™ Appliance, which contains virtual machines images that are targeted at a specific virtualization container (such as VMware, PowerVM, z/VM®) and contains a pre-wired, pre-configured, production ready software stack packaged inside virtual machine images (for example, IBM WebSphere Application Server, IBM WebSphere Portal, IBM DB2®). Delivery of an image in this manner minimizes the cost and complexity of the installation, configuration, and maintenance of running complex stacks of software.

By managing not only the virtual machine image catalog, but also the placement of these images onto hypervisors, as well as the application infrastructure middleware deployment inside the images, the total virtualization environment provisioning and deployment is simplified and standardized across the organization.

**Further optimizations**

Since I detest the term "best practices" I'll close with some final thoughts on how best to optimize a virtualized environment. As I noted at the beginning of this article, there have been many advancements in virtualization technology in the past few years, but a number of fundamentals remain unchanged since my original article. Among these fundamentals is my advice to:

- **Never overcommit memory** -- at least not with response time sensitive Java™ applications! As explained in my earlier article, the garbage collection mechanism in Java can have a profound performance impact when memory is overcommitted. In environments that are either infrequently used or are hosting applications for which response time is not critical (for example, application development environments), some minimal amount of memory overcommit might be possible without significant impact. That said, I've worked with a number of clients over the past three years who either weren't aware of the pitfalls associated with Java application workload memory overcommit, or somehow thought that the issues related to this topic did not apply to them -- they did. While there has been some research into better integration (awareness) between hypervisors and JVMs to optimize memory allocation and swapping, commercial availability of any solution remains quite a way off.

- **Use care when partitioning CPU cores.** This also still applies, but unlike memory overcommit, advances in CPU speeds and increased core counts in multi-core CPUs have made CPU core partitioning more practical. That said, it's important to recognize that CPU core partitioning doesn't create more resources, it simply enables you to divide and allocate the CPU core capacity across multiple images and the application workloads running on
those images. At the end of the day, there still needs to be adequate underlying physical CPU capacity to meet response time and throughput requirements when partitioning CPU cores. Otherwise, poor performance will result.

- **Monitor using both operating system and hypervisor tools.** If you monitor performance only using native OS tools, such as `top` or `vmstat`, you're getting only partial and possibly misleading information. OS tools only show resource use based on what is allocated to the virtual machine, which is likely very different from the resource pools that are available on the hypervisor. If you want an accurate picture of performance on the physical server, you need to measure using the hypervisor tools, and then use that information to adjust and optimize resource allocations to each virtual machine.

**Conclusion**

Hopefully you've found this re-examination of virtualization useful and will be able to employ some of the technologies described here in an optimal fashion.

Further, to show you that I'm not totally against suggesting best practices I provide one in closing: don't run with scissors. That was something my mother always told me, and we all know that our mothers are never wrong!
Resources

- The WebSphere Contrarian: Effectively leveraging virtualization with WebSphere Application Server
- WebSphere Virtual Enterprise product information
- WebSphere CloudBurst Appliance product information
- Virtualization Management Index
- Using VMware ESX Server with IBM WebSphere Application Server
- IBM developerWorks WebSphere
About the author

Tom Alcott

Tom Alcott is Senior Technical Staff Member (STSM) for IBM in the United States. He has been a member of the Worldwide WebSphere Technical Sales Support team since its inception in 1998. In this role, he spends most of his time trying to stay one page ahead of customers in the manual. Before he started working with WebSphere, he was a systems engineer for IBM's Transarc Lab supporting TXSeries. His background includes over 20 years of application design and development on both mainframe-based and distributed systems. He has written and presented extensively on a number of WebSphere run time issues.

© Copyright IBM Corporation 2011, 2015
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)