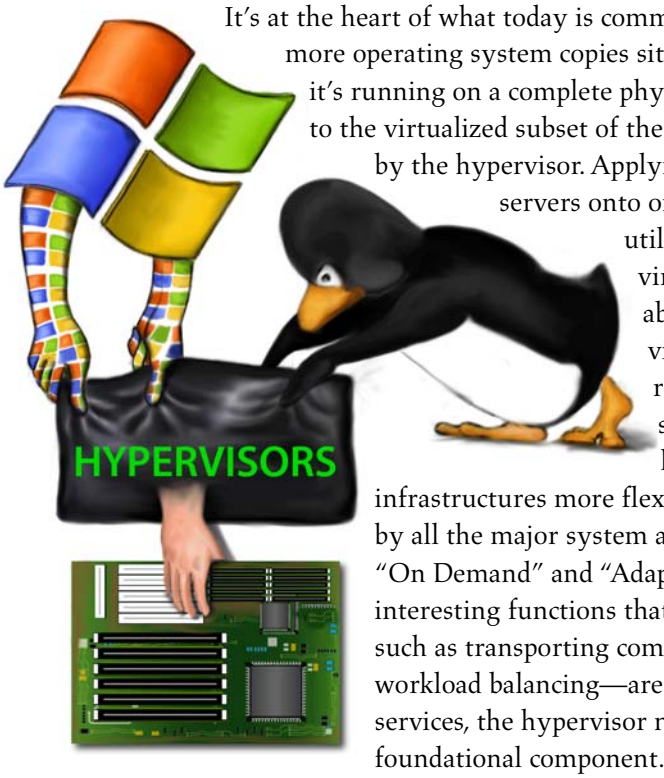
ghaff@
illuminata.com

Hypervisor Home

Quick Note

Gordon Haff
22 May 2006

IT wars oft rage over arcane technical distinctions in a manner reminiscent of Jonathan Swift's Lilliputians arguing incessantly over whether eggs should be broken on the large end or the small one. However, sometimes the arcana really matter—if not the details themselves, then their ultimate effects. The mounting war over hypervisors is one of the battles that has real-world consequences. It's often framed as a technical and architectural discussion. It is legitimately that, but only up to a point. It's at least as much a tussle among vendors trying to establish control and secure an advantaged market position.



A hypervisor is a software layer that abstracts and virtualizes a physical system. It's at the heart of what today is commonly called server virtualization.[1] One or more operating system copies sit on top of a hypervisor, each believing that it's running on a complete physical system—when in fact it only has access to the virtualized subset of the hardware, a "virtual machine" parceled out by the hypervisor. Applying this capability to consolidate multiple servers onto one—and thereby increase overall hardware utilization—got much of the early server virtualization press. More recently, the abstraction provided has itself started to be viewed as something important in its own right. By breaking the tight linkages between software and the hardware on which it runs, hypervisors help make datacenter infrastructures more flexible and dynamic—a direction being charted by all the major system and storage vendors as part of rubrics like "On Demand" and "Adaptive Enterprise." While many of the interesting functions that make virtualization interesting to users—such as transporting complete system state for disaster recovery or workload balancing—are realized though higher-level software and services, the hypervisor nonetheless remains an essential foundational component.

The tempest a brewin' is over where the hypervisor will live in the high volume part of the market, x86 servers.[2] Will it be just another feature of the Windows or Linux operating system? Or, instead, should it be independent of any particular operating system—and therefore more likely obtained from an ISV, or even

1 In fact, virtualization happens at many different locations within a server. However, the term "server virtualization" is often more narrowly construed as specifically referring to virtual machine technology, especially on x86 systems.

2 We speak here primarily of native hypervisors that interact directly with the hardware. In products such as VMware Server (the freely downloadable follow-on to GSX Server) and Microsoft Virtual Server, virtualization essentially piggybacks a host OS.

ultimately shipped by system vendors as part of the hardware? It's no small distinction for either the vendors involved, or for how users will obtain and deploy virtualized infrastructure.

Building into the OS

All the major x86 operating system vendors are building in virtualization layers. Statements by Microsoft exec Jim Allchin place Microsoft strongly in the "hypervisor as OS feature" camp—unsurprising, given Microsoft's historical propensity to tightly meld once-external functions into Windows. Not that the melding will happen soon; full server virtualization for Windows ("Veridian") looks like it will have to wait for an update to its next major server OS release, suggesting it won't be available until 2008 or thereabouts.[3]

As for Linux, its core virtualization component is Xen, the Open Source project originally developed at the University of Cambridge. Xen is slated to be included in Novell's SUSE Linux Enterprise Server 10 and Red Hat Enterprise Linux 5, as well as other commercial and community Linux distributions. Xen bundled into Linux will thus become available by late 2006 or early 2007. Sun is also working to incorporate Xen into Solaris as a complement to Solaris Containers, which take a different cut at subdividing system resources.[4]

The OS vendors want to own the virtualization layer because it increases the depth of their software stack—and, thereby, their scope of control. Especially with the advent of chip features such as Intel's VT and AMD's "Pacifica" that assist virtualization, hypervisors themselves will become less economically valuable. But "less" is very different from "not," especially over the transitional next few years. And over time, in the absence of any real standardization of hypervisor APIs and interfaces, control of the hypervisor also implies control of management layers and the other applications where there's real money to be made.

Conversely, let in a "foreign" hypervisor and you create an entry point for equally foreign management applications[5]—which will tend to siphon away user dollars that would otherwise have gone to your own products.

Of course, the vendors don't put it quite this way. Rather, they present virtualization as a natural addition to operating system function. As Microsoft's Allchin puts it: Windows already "virtualizes the CPU to give processing." In this sense, VMs just take that virtualization to the next level. And, in fact, there's a long history of operating systems subsuming functions and capabilities that were once commonly purchased as separate products. Think file systems, networking stacks, and thread libraries.

Built-in-ness is clearly the big argument in favor of marrying server virtualization to the operating system. You're buying the operating system anyway, so there's no need to buy a separate product from a third-party.[6] Furthermore, because virtualization is part-and-parcel of the OS, any ISV application certified for the OS would presumably be certified to run in a virtualized instance of that OS as well. As ISVs have become more comfortable with virtualization, this is less an issue than in the past. Nonetheless, anything that eliminates opportunities for finger-pointing has to be seen as a plus. In addition, using a standard OS instance to control the hypervisor means that most of the operating system's standard drivers will work—and therefore support a wide range of hardware out-of-the-box.[7] A similar story applies to management tools, skills, and procedures.

However, just because the operating system companies present the OS as the only right and proper hypervisor home, that doesn't make it so. The innateness (at least, once available and

3 http://www.virtualization.info/2005/04/microsoft-will-not-embed.html

4 See our *New Containments for New Times*.

5 Such as VMware's Virtual Center.

6 It's possible that Microsoft may offer server virtualization as an add-on, but the basic argument still applies.

7 However, Virtual Iron, VMware, and XenSource can all leverage Linux drivers—often unchanged—so this difference in approaches isn't as great as it could be.

production-ready) of OS-embedded hypervisors may well make them the default choice for many. But they have their downsides as well.

Maintaining Independence

OS-embedded hypervisors are inherently allied to a specific operating system, and even to specific versions of that operating system. In part, this is a philosophical failing; it's just architecturally neater to have the virtualization layer be independent of any of the guest OSs running on top. Indeed, if it's to be wrapped together with anything, the server itself would be the better choice. After all, servers and storage already incorporate a variety of features (such as BIOS and Logical Block Addressing) to mask complexities and messiness of the underlying hardware. Therefore, it's not a big stretch at all to imagine that some day Intel motherboards and HP ProLiant servers will ship with a hypervisor as a standard feature.

Of course, IT shops aren't big on philosophy as a reason to do things, but there are practical downsides to having the hypervisor be part of the operating system as well.

It may not be that common today, at least outside of development/test environments, to run both Windows and Linux instances on the same physical server. However, having both environments *somewhere* in the shop is commonplace with IT organizations of all sizes and stripes. In this scenario, standardizing on one OS-based hypervisor would mean a Windows domain on servers with Linux guests, or the other way around. This would likely have license cost implications[8] as well as skill and training ones. Nor is keeping Windows and Linux servers apart from one other all that great an approach. Not only does it cut against the flexibility of a virtualized infrastructure, but it likely demands largely disjoint management applications and practices.

Furthermore, OS vendors are hardly the most disinterested parties to be working with and supporting competitive products. Yes, Microsoft did indeed state recently that it would support Linux guests. But between "don't run" and "enthusiastically embrace" lies a vast gulf. It remains to be seen whether Microsoft's Linux support will be sufficiently wholehearted to truly be a viable commercial solution.

The one truly relevant example from the non-x86 world, IBM's POWER Hypervisor, also suggests the benefits of a separate hypervisor. OS-independence became particularly important to IBM as its i5 (*née* iSeries and AS/400) and p5 (*née* pSeries) servers increasingly morphed into a single set of hardware; it wouldn't have done to require an i5 admin to learn Unix or Linux in order to perform virtualization tasks. And *vice versa*. Keeping the management operating system-independent by using the Hardware Management Console (HMC) or, more recently, the Integrated Virtualization Manager (IVM) avoids the complexities of managing through a full-blown, possibly-unfamiliar operating system.

The Independents

VMware most vocally cries that hypervisors should maintain independence from, and avoid affinity with, any specific operating system. It, too, has its own interests at heart. In part it's about selling its ESX Server hypervisor—whose prices will inevitably drop over time, but which still has real value today given its maturity, functional depth, and broad market penetration. At least as important, VMware wants to up-sell to management applications and other command and control software using ESX Server as the virtualization foundation layer.

[8] For example, Microsoft currently allows the user to run several Windows guests on its Virtual Server product at no additional charge (under some circumstances). If it carries similar practices forward to a future embedded hypervisor, that implies that it will tend to be more cost-effective to use a Windows hypervisor to host Windows guests.

x86 Hypervisor Characterizations and Components

	OS-embedded	*Independent*
Proponents	Microsoft, Novell, Red Hat	VMware, XenSource, Virtual Iron
Hypervisor Control	Standard operating system instance	Dedicated Domain0 controller or equivalent.
Key Hypervisor Technologies	Microsoft Windows, Xen	VMware ESX Server, Xen
Packaging	Component in Linux distribution. Unknown if Microsoft will bundle or offer as add-on.	Procured as separate product. Could potentially be embedded with hardware/system in future.
Drivers	Can use most standard OS drivers.	Requires drivers unique to hypervisor/controller, but can typically leverage Linux drivers.
Multi-OS Guest Support	Yes—but support and certifications may be limited or licensing may make unattractive.	Yes
Security	Same as standard OS instance.	Potentially simplified by fewer services and lower complexity in console OS.
Require Hardware Assist	Yes	No for VMware. Yes for XenSource and Virtual Iron.
First Available	Novell/Red Hat: ~2H06 Microsoft: ~2008	VMware ESX: 2002 Virtual Iron/XenSource: ~2H06

Xen is also being leveraged by Virtual Iron and XenSource as an autonomous hypervisor layer without ties to Linux or Solaris. Both companies are pairing the Xen hypervisor with their own "Domain0 controllers"—stripped-down operating systems that manage the guest VMs and handle the primary driver interfaces to the underlying hardware.[9] In contrast, the upcoming Linux and Solaris implementations use a standard instance of their respective operating systems for Dom0.

VMware ESX server also uses an independent hypervisor that it marries to a console OS, which provides the system management functions and initiates the execution of the virtualization layer and resource manager. Although VMware's approach has some architectural similarities to the Virtual Iron and XenSource designs, VMware's hypervisor directly handles a variety of functions—such as resource management, virtual networking, and device driver I/O—that in the case of Xen are delegated to Dom0. In this respect, VMware more resembles IBM's POWER Hypervisor in that its console OS only has to handle some fairly basic bootstrapping and management functions rather than being effectively part-and-parcel with the hypervisor.

Although it's hard to make sweeping generalizations about the "best" weight for a console OS—much depends on robustness and security as well as size—smaller and simpler tends to be better than full-blown Linux and Windows instances. The larger the operating system, the more complex and more difficult to harden and secure. If not inherently superior, a minimalist console at least has far fewer potential failure modes and attack points.

Conclusion

Although many of the virtualization products now being avidly promoted won't be truly production-ready for many months or even several years, it's not too early to start thinking seriously about ultimate objectives and goals. After all, it was often

[9] Virtual Iron's Dom0 derives from their previous proprietary hypervisor; XenSource hasn't disclosed whether its sourcing Dom0 from outside or writing it in-house. See our *Hypervisors in Boston* for more discussion of these companies' products and strategies.

tactical purchases and implementations that created the server sprawl that is among the IT challenges that virtualized infrastructures are being tasked to rationalize. Let's not go there again!

Among those important decisions is the manner in which server virtualization will be implemented. For just a few individual servers here and there, the choices may not matter much or, in any case, can reasonably be made on the basis of current feature, function, and price rather than a longer view. But as a few virtualized servers become a pervasively virtualized infrastructure, the planning horizon should likewise broaden.

As virtualization gets built into Linux and Windows, and as it matures into a production-class offering, many IT shops will be sorely tempted to just use whatever comes bundled rather than to search out and source third-party products that are more OS-agnostic, functionally rich, or mature. That may not even be a bad decision for environments that are largely tied to one particular operating system. But most shops are heterogeneous. Furthermore, developing application deployment styles—such as virtual appliances—map best to infrastructures that permit a relatively arbitrary mix of operating systems.[10] At the very least, virtualization choices should be *conscious* decisions and considered strategically and architecturally rather than by default.

[10] Virtual appliances wrap an operating system together with the other components on which it depends into a VM image that can be loaded directly onto a virtualized infrastructure—rather than installing it on an OS as has been the historical norm.