SERVER CONSOLIDATION

HARDWARE-ASSIST

FULL VIRTUALIZATION

LIVE MIGRATION

DESKTOP VIRTUALIZATION

TOM KILLALEA, AMAZON.COM

When you dig into the details of supposedly “overnight” success stories, you frequently discover that they’ve actually been years in the making. Virtualization has been around for more than 30 years—since the days when some of you were feeding stacks of punch cards into very physical machines—yet in 2007 it “tipped.” VMware was the IPO sensation of the year; in November 2007 no fewer than four major operating system vendors (Microsoft, Oracle, Red Hat, and Sun) announced significant new virtualization capabilities; and among fashionable technologists it seems *virtual* has become the new *black*.

WHAT IS IT?

Virtualization is the provision of an abstraction between a user and a physical resource in a way that preserves for the user the illusion that he or she could actually be interacting directly with the physical resource. While you could imagine virtualizing any physical resource, the focus of this issue of *Queue* is the computing machine virtualization that is the current rage. The user gets a high-fidelity copy of what appears to be a complete computer system, while he or she is actually dealing with an abstraction layer known as the VMM (virtual machine monitor) that runs on the real machine and maps resources on behalf of the user.

meet the virt

PARAVIRTUALIZATION

*Virtualization technology isn't new,
but it has matured a lot over the past 30 years.*

It's interesting to go back to Gerald Popek and Robert Goldberg, writing in 1974, to see that from a model point of view very little has changed: "A virtual machine is taken to be an *efficient, isolated duplicate* of the real machine," they wrote, explaining the concept through the idea of the VMM: "As a piece of software a VMM has three essential characteristics. First, the VMM provides an environment for programs which is essentially identical with the original machine; second, programs run in this environment show at worst only minor decreases in speed; and last, the VMM is in complete control of system resources."¹

Figure 1 is an illustration from Popek and Goldberg's

1974 paper, compared side by side with a modern illustration from one of VMware's white papers.²

WHY IS IT INTERESTING?

In 1882 J.P. Morgan's house on New York's Madison Avenue became the first private residence with incandescent lighting.³ The lack of abstraction from the complexity and unpredictability of power generation/delivery, along with the lack of distance between him and the generation source (a steam engine, boiler, and pair of generators in his back garden), caused Morgan and his neighbors considerable inconvenience.

Abstractions are useful, particularly if they are simple

meet the virt

and efficient. The main benefit of any abstraction is the decoupling that it facilitates. With virtualization the user is able not to care about the hardware and how it actually behaves. As long as the performance characteristics are met, the user can also be freed from caring about who operates the hardware, where the hardware is located, and whose logo (if any) is on it. The ultimate extension of this is the utility computing model provided by virtualized compute services—for example, Amazon EC2 (Elastic Compute Cloud).

It's worth looking at the benefits of virtualization from two points of view: from the perspective of the user who is above the VMM and from the perspective of the infrastructure provider beneath it.

THE VIEW FROM ABOVE

To an application developer or service operator, virtualization can be very enabling. You can have access to (virtual) quantities of resources in combinations that aren't practical or cost effective in the physical world. You can achieve the illusion of having hardware components that are unattainable or fleets of systems that you can't afford. You can isolate applications into distinct environments to facilitate needs such as business continu-

ity, destructive testing, closer observation, or crash-only operation. You can spend less time upgrading operating systems and qualifying hardware. You can isolate an application that you don't trust in a jail, or indeed be jailed yourself if you're not to be trusted by others. Finally, it's important to bear in mind that various virtual machine environments with quite distinct requirements can run simultaneously on a single set of physical hardware resources.

In some organizations the combination of procurement policies and vendor lead time pressures application developers to specify production systems before their application is even designed. Guesswork ensues. Virtualization has the potential of reducing the cost of a bad guess, and beyond that it could facilitate self-scaling and right-sizing applications that make guesswork a thing of the past.

THE VIEW FROM BELOW

Running multiple virtual machines, perhaps with different operating systems, on a single physical machine is the most widely discussed advantage from the point of view of the hardware infrastructure provider. Such server consolidation facilitates more efficient use of physical infrastructure assets.

There are other interesting benefits from the infrastructure provider's point of view. You might make changes to the physical infrastructure to scale capacity, to repair equipment, or to modify the configuration of

Virtualization—Then and Now

Illustration courtesy of VMware

FIG 1

devices and subsystems without involving your users, as long as the fidelity of their abstraction is preserved and their performance expectations are met. You can enable the user to keep running an old application even if it expects an operating system that doesn't run on modern hardware. You can have homogeneity in your physical infrastructure and host operating system and still provide your users with choices and potentially with heterogeneity at the guest operating system level if required. Of course, you can also use virtualization to protect the hardware from your users and to do inspection and policy enforcement.

WHY NOW?

Why does a technology such as virtualization "tip" and quite suddenly gain so much momentum? One turning point was the 1997 Disco work by Mendel Rosenblum and colleagues at Stanford University on using virtual machines to run multiple commodity operating systems on a single scalable multiprocessor.⁴ Their later breakthrough in 1998, after founding VMware, allowed users to run multiple instances of x86-compatible operating systems on a single commodity PC.

As the adoption of open source operating systems on commodity hardware gained momentum, in 2003 researchers at Cambridge released Xen, an open source VMM that allows multiple commodity operating systems to share conventional hardware very efficiently.⁵ The availability of both commercial and open source VMMs for commodity hardware and the excessive (for many applications) capacity of modern PCs and servers have combined to lead to the rapid adoption of virtualization in recent years.

HOW IS IT DONE?

The VMM provides the hardware abstraction that encapsulates and isolates a given virtual machine from other virtual machines on the physical machine. The guest virtual machine may have privileged and nonprivileged instructions that need to be run. For efficiency the nonprivileged ones can be executed directly without the involvement of the VMM. The privileged instructions are caught by the VMM where they can be simulated or mapped to a physical resource.

In the Xen implementation the VMM can operate at a higher privilege level than the supervisor code of the guest operating systems in the virtual machine, and as a result it is described as a *hypervisor*. On x86 architectures the supervisor code from the operating system in the guest VM is typically run in the otherwise unused ring-

level one, and is thus isolated from applications in the guest virtual machine that will run in ring three.

AT WHAT LEVEL VIRTUALIZATION?

The special focus of this issue of *Queue* is on machine or platform virtualization, but there are other levels at which virtualization can be provided to facilitate useful abstractions.

Application virtual machines provide an abstraction from other applications within the same operating system, rather than below the operating system. Aside from abstraction and isolation, their purpose is to allow applications to be portable, without modification, to different computer architectures or operating systems. Examples include the Java Virtual Machine and the Microsoft .Net CLR (Common Language Runtime).

A different level of abstraction is evident in the various approaches to providing managed desktops and application mobility to personal computer or terminal users. Among these are the research on the Collective architecture done at Stanford⁶ and the Sun Ray virtual thin client that Sun Microsystems uses for desktop virtualization.

In addition, a variety of efforts are under way to provide a higher level of abstraction by creating a desktop environment within the confines of a browser. One example is WebShaka's YouOS.

PERFORMANCE IMPACT

As noted earlier, performance is one of Popek and Goldberg's essential characteristics of a VMM.⁷ While presenting the implications of virtualization for software developers, Uli Drepper covers the performance topic extensively elsewhere in this issue ("The Cost of Virtualization"), while Scott Rixner discusses performance impacts specifically in the area of network I/O ("Network Virtualization: Breaking the Performance Barrier").

AVAILABILITY IMPACT

Virtualization on its own has little direct impact on availability. In some cases there can be a positive effect from the user's abstraction from the physical hardware. In addition, the spreading of multiple instances of an application across many small virtual machines can force the application owner to consider how to scale out rather than up. Mendel Rosenblum et al. have researched the possibility of encapsulation of the complete state of a running virtual machine, including its operating system, applications, data, and processes.⁸

This has led to what's now widely called *live migration*, the ability to suspend a running virtual machine

meet the virts

and subsequently to resume it on a different physical system. This is done not only for availability reasons such as working around planned downtime, but also in some cases so that a system can be resumed on a more appropriately sized virtual machine. Some applications will not deal gracefully with the suspension of time, however. As an approach to fault tolerance, the attempt to keep an instance alive through live migration contrasts markedly with the model advanced by the ROC (Recovery-oriented Computing) joint project between Stanford University and UC Berkeley.⁹ The ROC group doesn't focus on MTTF (mean time to failure). Such a focus might assume that failures are generally problems that can be known ahead of time and should be avoided (or if possible routed around or migrated around). Instead they take a crash-only approach that focuses more on MTTR (mean time to repair). The ROC model appears to be gaining traction among major Internet services.

BUSINESS IMPACT

One benefit of virtualization is the breaking of coupling between applications, operating systems, and the physical hardware. Some interesting possibilities arise as the relationship between users and the hardware that they consume is abstracted in a virtualized environment, as long as the fidelity characteristic is maintained. Not least of these possibilities is vastly greater resource utilization, especially in utility computing environments as discussed elsewhere in this issue by Werner Vogels ("Beyond Server Consolidation").

Yet another business consideration is the impact that virtualization might have on software licensing. Although a clear pattern has yet to emerge, one likely model is treating a virtual processor as equivalent to a real licensed processor or core. Another possibility is metered licensing

that is based on how long the virtual machine runs.

SUMMARY

Virtualization looks set to become more widely adopted. The capabilities of the technology have evolved considerably in recent years, and it now provides a useful abstraction that is of benefit both to the virtual machine user and to the physical infrastructure provider. Q

REFERENCES

1. Popek, G. J., Goldberg, R. P. 1974. Formal requirements for virtualizable third-generation architectures. *Communications of the ACM* 17(7): 412-421.
2. VMware. Understanding full virtualization, paravirtualization, and hardware assist; http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf.
3. Jonnes, J. 2003. Chapter 1 in *Empires of Light*. New York: Random House.
4. Bugnion, E., Devine, S., Govil, K., Rosenblum, M. 1997. Disco: Running commodity operating systems on scalable multiprocessors. *ACM Transactions on Computer Systems* 15(4): 412-447.
5. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A. 2003. Xen and the Art of Virtualization. *Proceedings of the ACM Symposium on Operating System Principles* (October).
6. Chandra, R., Zeldovich, N., Sapuntzakis, C., Lam, M. S. 2005. The Collective: A cache-based system management architecture. *Proceedings of the Symposium on Networked Systems Design and Implementation*.
7. See reference 1.
8. Sapuntzakis, C. P., Chandra, R., Pfaff, B., Chow, J., Lam, M. S., Rosenblum, M. 2002. Optimizing the migration of virtual computers. *Proceedings of the Symposium on Operating Systems Design and Implementation*: 377-390.
9. Patterson, D. A., Brown, A., Broadwell, P., Candea, G., Chen, M., Cutler, J., Enriquez, P., Fox, A., Kiciman, E., Merzbacher, M., Oppenheimer, D., Sastry, N., Tetzlaff, W., Traupman, J., Treuhaft, N. Recovery-oriented Computing (ROC): Motivation, definition, techniques, and case studies; http://roc.cs.berkeley.edu/papers/ROC_TR02-1175.pdf.

LOVE IT, HATE IT? LET US KNOW

feedback@acmqueue.com or www.acmqueue.com/forums

TOM KILLALEA has worked at Amazon.com since 1998 and is the vice president of technology with responsibility for infrastructure and distributed systems engineering.

© 2008 ACM 1542-7730/08/0100 \$5.00