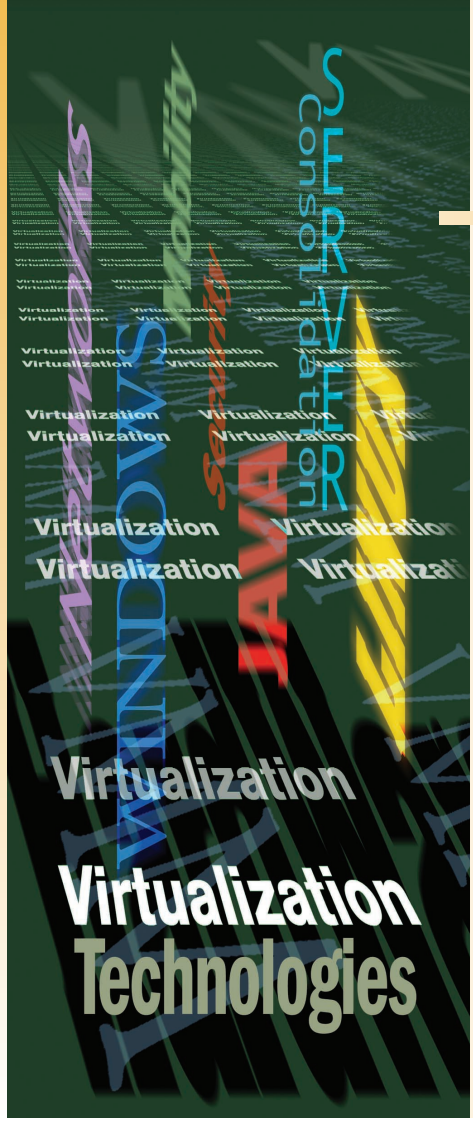

Resource Virtualization Renaissance

Virtualization technologies encompass a variety of mechanisms and techniques used to address computer system problems such as security, performance, and reliability by decoupling the architecture and user-perceived behavior of hardware and software resources from their physical implementation.

Renato Figueiredo
University of Florida

Peter A. Dinda
Northwestern University

José Fortes
University of Florida

In his 1992 Turing Award lecture, Butler Lampson quoted David Wheeler's comment that "any problem in computer science can be solved with another layer of indirection." Indeed, the indirection layer that resource virtualization introduces is increasingly being used to address many computer systems problems in security, performance optimization, system administration, and reliability, among others.

The articles in this special issue of *Computer* bring the reader up to date on the state of the art in resource virtualization technologies and show how virtualization has been, is, and can be further used to solve important computer systems problems.

While its roots trace back decades, resource virtualization has drawn renewed attention in recent years. Successful commercial products from VMware and Microsoft have extended the reach of virtual machines to commodity platforms. Virtualization needs have prompted extensions to the IA-32 instruction set (Intel's VT/Vanderpool, AMD's Pacifica) and to IBM's POWER architecture. Stable open source virtual servers (VServer, User-mode Linux) and paravirtualized VMs (Xen) have been developed. Language-level VMs (Sun Java, Microsoft CLR) run in a variety of platforms, ranging from embedded to server systems. Enterprises commonly employ virtual LANs and private networks (VLANs, VPNs). System area networks use storage virtualization to simplify the management of complex, heterogeneous storage configurations.

Virtualization technologies encompass a variety of mechanisms and techniques used to decouple the architecture and user-perceived behavior of hardware and software resources from their physical implementation. Central to resource virtualization is the introduction of an indirection layer below the execution environment seen by applications and operating systems. A key component of virtual machines, the virtual machine monitor (VMM), provides a layer between software environments and physical hardware that is programmable, transparent to the software above it, and makes efficient use of the hardware below it.

Through indirection, VMMs and similar networking and storage resource virtualization techniques provide the capability of multiplexing, on a single physical resource, many virtual systems that are isolated from each other. The insertion of a lower-level abstraction below existing abstractions is in marked contrast to the common approach of constructing a higher-level abstraction on top of the

Published by the IEEE Computer Society

existing highest-level abstraction in the software stack.

HISTORICAL PERSPECTIVE

VMMs, also known as hypervisors, first reached prominence in the early 1970s and achieved commercial success with the IBM 370 mainframe series. Virtualization allowed mainframes to run multiple operating systems simultaneously, thus making it possible to time-share expensive hardware without requiring software modifications to legacy systems, including single-user operating systems. In his survey published three decades ago in *Computer*,[1] Robert P. Goldberg identified the advantages of virtual machines beyond such a key driver—server consolidation—including mixing batch processing, supporting interactive applications and development tasks, and improving data security and reliability.

With the advent of low-cost minicomputers and personal computers, the need for virtualization declined. Although VMMs (VM/386) and even hardware support (V8086 mode) for PCs existed during the 1980s and early 1990s, it was not until the advent of Virtual PC and VMware's workstation and server products in the late 1990s that practical, general-purpose VMMs became available for commodity hardware that could run any operating system.

While modern servers are much less expensive and more powerful than the machines of decades past, their total cost of ownership includes maintenance, support, and administration, as well as the costs associated with security breaches and system failures. Server consolidation thus remains a key driver for virtualization, with researchers and industry particularly focusing on improving security and availability.

Today, commercial VMMs by VMware and Microsoft are widely used, with millions of copies having been sold. In addition to their use in server environments, these virtual machine technologies also are used in desktop environments to run multiple operating systems, typically Windows and Linux.

IN THIS ISSUE

The virtualization layer is marked by its simplicity: VMMs often have orders-of-magnitude fewer lines of code than modern operating systems. However, while VMMs had proven to be successful for processors that met the requirements for virtualizability laid out by Gerald J. Popek and Robert P. Goldberg,[2] many of the leading modern microprocessor architectures, including Intel's IA-32, did not meet such requirements. The resurgence of VMMs has relied on ingenious software-based solutions and has prompted architectural changes to meet virtualization requirements.

In "The Architecture of Virtual Machines," James E. Smith and Ravi Nair describe the common architecture traits of the wide range of approaches to virtualization of computer systems and provide a summarizing taxonomy.

Server consolidation remains a key driver for virtualization, with researchers and industry focusing on improving security and availability.

Modern VMMs: Techniques and their challenges

In "Virtual Machine Monitors: Current Technology and Future Trends" by Mendel Rosenblum and Tal Garfinkel, we learn how VMware overcomes these limitations to provide the virtual machine abstraction on commodity IA-32 hardware. The authors also provide an overview of the wide range of uses of VMMs. In "Intel Virtualization Technology," Fernando C.M. Martins and coauthors present, for the first time, the VT (codenamed "Vanderpool") extensions to the IA-32 and IA-64 architectures. These extensions, which will ship soon, provide hardware assistance to make writing VMMs for commodity hardware much easier and faster.

Some believe that a traditional VMM is too heavyweight for some uses, such as running multiple Web servers on the same hardware. Paravirtualization is a lighter-weight approach in which the VMM's interface provides a system call interface designed to support the straightforward porting of operating system kernels. Paravirtualization offers scalability and performance advantages, but requires porting. In "Rethinking the Design of Virtual Machine Monitors," Andrew Whitaker and coauthors make the argument for paravirtualization and present the Denali VMM.

Virtualization: A broader perspective

The VMMs described so far function essentially as operating systems whose applications run on other operating systems. Developers have also shown considerable interest in virtualization at other levels of the software stack to, for example, extend existing operating systems to present the abstraction of multiple servers. Examples include VServer and Ensim's products. Web hosting and other domains already commonly use virtual server technology.

Language runtimes that interpret and translate binaries compiled for abstract architectures enable

Resource virtualization provides a vehicle to use in practice many of the results from distributed systems research.

portability by providing programs with an identical execution environment regardless of the resource. As with OS-level VMMs, language-level VMs have a long history, including UNCOL from the 1960s and the UCSD P-System from the 1970s and 1980s. Today, Sun's Java and Microsoft's CLR VMMs dominate the market for language-level virtualization technologies.

An architectural-level VMM can use co-designed hardware and software to provide different instruction-set architectures from a single microprocessor core. The Transmeta Crusoe processor is an architectural-level VMM that can execute IA-32 code with very low power requirements. Other approaches to virtualization include emulators and binary translators.

Virtualization techniques also can aid development and testing. Peter Magnusson describes one such application in "The Virtual Test Lab" in the Embedded Computing column in this issue.

Network virtualization makes it possible to completely decouple the distributed computing environment an application sees from its underlying physical resources. In "Virtual Distributed Environments in a Shared Infrastructure," Paul Ruth and coauthors describe Violin, a virtual network that presents as its abstraction the commonplace layer 2 and 3 abstractions of Ethernet and the Internet Protocol. An entire virtual internet can be created with Violin to interconnect VMs.

EMERGING OPPORTUNITIES

The key motivations behind virtualization summarized by Goldberg in 1974 remain important for today's computer systems. In the span of three decades, computers have become not only increasingly faster and cheaper, but also increasingly connected to other computers both within local area and across wide area networks. Exciting new opportunities for applying resource virtualization have arisen in this context, where large numbers of high-performance machines are connected to the Internet.

Resource virtualization provides a vehicle to use in practice many of the results from distributed systems research conducted in the 1980s and 1990s. For example, migration, checkpointing, load balancing, and redundancy through replication and group communication can be applied to VMs much more easily than to processes because of the encapsulated nature of the VM's state and the VM's very simple interface to the outside world. Further, virtual networks can maintain connectivity for the VMs without any intervention. The VMs can in turn run existing, unmodified applications and operating systems. Virtualization has already spurred new research in core distributed systems areas.

The encapsulated nature of VMs and virtual networks, combined with their isolation properties, also make utility and Grid computing much more straightforward.[3] Resource providers can offer the maximum possible flexibility—because users can install an entire operating system—and can do so with minimal administrative cost because the VM can be treated as a black box that only needs to have its resource share satisfied. We believe that virtualization will spur further research on economic models for distributed utility computing.

Since the 1990s, techniques and middleware for building applications that can adapt to varying resource availability to provide a stable user experience have been and continue to be developed. A key problem with this work is that it has either required application modifications or limited the application developer to using a particular middleware framework. Resource reservation systems have posed similar constraints.

The invisible and simple abstraction layer that resource virtualization provides makes it possible to run applications adaptively—or exploit reservable resources—without user or developer participation. A key research question is whether enough information, adaptation mechanisms, and control are available at this level to come close to per-application adaptation. Early results, such as those presented in the Violin article in this issue, suggest that this may be the case for many application domains. We believe that automatic adaptation and reservation at the virtualization layer offers an exciting area of research and have committed to it ourselves.

"Classic" OS-level VMMs are key to supporting legacy systems in virtualized environments. Nonetheless, there has been a substantial investment in recent years in the development of software using languages for which language-level VMMs are available. With language-level VMMs like Microsoft's CLR becoming language agnostic, and the performance of programs running on them using just-in-time compilation or translation creeping ever closer to the performance of native code, there are opportunities to explore possible synergies between OS-level and language-level VMMs.

Very different communities back these systems, and interesting issues may emerge as they learn from each other. The recently formed ACM/Usenix Conference on Virtual Execution Environments

(VEE) is a forum that will help bring together researchers from both communities.

Some operating system functionality requires hardware assistance. Although Popek and Goldberg defined the basic requirements for OS-level virtualization of a computer system, today we encounter collections of VMs interconnected with some form of overlay network. Because VMMs and overlay networks can benefit from hardware-level assistance, there are opportunities to investigate what hardware features computer systems and routers should support to facilitate the use of virtual networks of virtual machines. ■

References

1. R.P. Goldberg, "Survey of Virtual Machine Research," *Computer*, June 1974, pp. 34-45.
2. G.J. Popek and R.P. Goldberg, "Formal Requirements for Virtualizable Third-Generation Architectures," *Comm. ACM*, July 1974, pp. 412-421.
3. R.J. Figueiredo, P.A. Dinda, and J.A.B. Fortes, "A Case for Grid Computing on Virtual Machines," *Proc. Int'l Conf. Distributed Computing Systems* (ICDCS 03), IEEE CS Press, 2003, pp. 550-559.

Renato Figueiredo *is an assistant professor in the Department of Electrical and Computer Engineering at the University of Florida. His research interests are in distributed computing, operating systems, and computer architecture. Figueiredo received a PhD in computer engineering from Purdue University. He is a member of the IEEE and the ACM. Contact him at renato@acis.ufl.edu.*

Peter A. Dinda *is an assistant professor at Northwestern University, where he holds the Lisa Wissner-Slivka and Benjamin Slivka Junior Chair in Computer Science. His research interests are in parallel and distributed systems, performance analysis, and adaptive systems. He received a PhD in computer science from Carnegie Mellon University. He is a member of the IEEE and the ACM. Contact him at pdinda@cs.northwestern.edu.*

José Fortes *is a professor and a BellSouth Eminent Scholar at the University of Florida, where he is also the director of the Advanced Computing and Information Systems Laboratory. His research interests are in the areas of distributed computing and information processing, computer architecture, and nanocomputing. Fortes received a PhD in electrical engineering from the University of Southern California. He is a Fellow of the IEEE and a member of the ACM. Contact him at fortes@ufl.edu.*