Editor: **Diomidis Spinellis**
Athens University of Economics and Business, dds@aueb.gr

Virtualize Me

Diomidis Spinellis

THE VIRTUAL MACHINE (VM) is the most dazzling comeback in information technology. IBM implemented a VM platform architecture in the late 1960s in its CP/CMS operating system. The company's goal was to provide the time-sharing capabilities that its batch-oriented System/360 lacked. Thus a simple control program (CP) created a VM environment where multiple instances of the single-user CMS operating system could run in parallel. Thirty years later, virtualization was rediscovered when companies like VMware found ways to virtualize the less accommodating Intel x86 processor architecture. The popularity of Intel's platform and the huge amount of software running on it made virtualization an attractive proposition, spawning within a decade tens of proprietary and open source virtualization platforms.

Virtualization has progressed a lot from its primitive beginnings. Today, virtualization lets us run most modern operating systems in a VM that can be hosted on facilities ranging from our laptop to a datacenter in the cloud. Once an operating system runs as a guest in a VM host (also known as hypervisor), it becomes easy to control via high-level operations. You can save its image into a file, move it from one host to another, launch multiple clones, suspend it until it's needed, share it with others, rent it as a service, or ship it to a customer. Techniques such as paravirtualization, in which the guest offloads some heavy lifting to the hypervisor, allow the guest systems to run efficiently and share hardware with a minimal waste of resources. Virtualization means never having to beg for a server.

Virtualization has progressed a lot from its primitive beginnings.

As developers, we can benefit from virtualization technology in several ways: better developer environments, more efficient testing, easier distribution, and improved infrastructure management.

Development

Increasingly deep and wide application stacks have made provisioning a development environment a tricky task. Modern applications often depend on many layers of large infrastructure chunks that might include the application development framework, third-party libraries, a relational database, a message broker, scripting languages, the revision control system, debuggers and tracing tools, and logging facilities. Each component can introduce other dependencies and may require careful installation and configuration, while successive versions of the application might depend on diverse specific versions of infrastructure.

With VM technology, it's possible to create a virtualized development environment that contains all the tools, applications, and libraries that a programmer requires. A standardized development environment packaged and distributed as a VM can help bring developers up to steam in minutes rather than days. All that programmers have to do is download the development environment matching the product version they're working on and fire it up on their desktop workstation or even their laptop. Tools like SUSE Studio (http://susestudio.com) and the more specialized Vagrant (http://vagrantup.com) simplify the specification and creation of such a development environment.

Apart from convenience to developers and operations staff, a virtualized development environment offers many other advantages:

- A standardized development VM ensures that developers work on the same setup as the one used for the production system. This parity between development and production environments eliminates many errors that in the past would surface only when code moved into production.
- By creating a VM image shared by all developers, the (sometimes substantial) cost of setting up the production system's components is amortized rather than replicated among many developers.
- With a virtualized development environment, developers can work on a different operating system (say, Mac OS X) from that used for production.
- The VM environment allows the execution of sophisticated platform-specific tools, such as dtrace or Visual Studio, which might not otherwise be available on the developer's machine.
- Kernel and embedded-system development are much more comfortable on a VM than on the actual hardware, because the VM frees you from the need to physically access the hardware, and allows speedy recovery from crashes.

Virtualization means never having to beg for a server.

System designers and administrators also benefit from the availability of many prepackaged third-party VM images. These allow painless experimentation with a complex setup: just download the VM image, fire it up, and you're ready to go. Popular virtualization platforms come with tens to hundreds of VM images of preinstalled operating systems, development environments, or end-user applications. By using these, you can defer the investment for properly installing and configuring a system in your organization to the time when you've actually settled on the one you'll use.

Testing

Virtualization is also a boon when testing a system. Ensuring a pristine testing environment is tricky, especially after you've run many tests and haphazardly modified your setup to replicate a specific error. With a VM image of the testing environment, things become easy. You start each test or testing session on a fresh machine instance booted from the testing VM image. When you finish testing, you simply discard the instance, never to see it again. Handily, when you nail down a difficult-to-replicate bug, you can save the VM image where you performed the feat to pass it on to a developer for further examination. This technique and the parity among development, testing, and production systems eradicates many of the "works on my machine" reasons for not fixing bugs.

In addition, the ability of testers to launch on their computer VMs corresponding to diverse platforms makes it easier to test your product's portability and compatibility. In the past, such testing would require maintaining a testing laboratory with physical instances of each machine.

Deployment

VM images also simplify the deployment of your application. You can package all your system's components and setup into a so-called VM appliance, which your users can download and run at the touch of a button. Many cloud infrastructure providers support the hosting of such images, further simplifying the distribution. This practice eliminates problems that stem from the interaction of your application with the customer's setup and minimizes the support required to tailor your system to a customer's environment.

However, deploying applications as virtual appliances is not without problems. Once you go that route, you become responsible for keeping up to date all your system's infrastructure included in the VM: the operating system, third-party libraries, and other applications. If that requires deploying a completely new VM, you need to plan for how to transfer your customer's content and customizations from the old VM image to the new one. This might not be too difficult for stateless systems such as firewalls, spam filters, and caching proxies, especially if you provide a managed solution for storing their settings on your own infrastructure. On the other hand, patching together applications that run on separate VMs is more complex than if they were running on the same instance. On balance, deploying applications as virtual appliances is an attractive proposition for delivering stand-alone demos—but it's not a panacea for production setups.

Operations

Virtualization tremendously benefits the staff handling your operations, but this doesn't mean that you as a developer can't pick up some winnings in this area as well. Simplified and more effective infrastructure management means less work to support such features within the system you're implementing. Virtualization can easily provide many features that older systems once provided through heaps of native code. Application provisioning for higher-throughput could just mean firing up more VM images of your system behind a load balancer. System maintenance under high availability requirements can often be performed by running the new VM side-by-side with the old one on the same hardware and adjusting the load balancer settings. Disaster recovery is also simpler when you can regularly backup a VM's image to a remote site. Finally, the utility of virtualized instances is compounded by the fact that they can be run in the cloud through services such as the Amazon Elastic Compute Cloud and the Rackspace Cloud Servers. This means that you can set up your organization's infrastructure without tying up capital and resources on procuring and maintaining physical assets. But this is a topic for another column.

DIOMIDIS SPINELLIS is a professor in the Department of Management Science and Technology at the Athens University of Economics and Business and the author of the books *Code Reading* and *Code Quality: The Open Source Perspective* (Addison-Wesley, 2003, 2006). Contact him at dds@aueb.gr.

Post your comments online
by visiting the column's blog:

www.spinellis.gr/tools

IEEE HOT CHIPS 2012

Hot Chips 24: A Symposium on High Performance Chips

27–29 August 2012
Cupertino, CA

HOT CHIPS is one of the semiconductor industry's leading conferences on high-performance microprocessors and related integrated circuits. This year's emphasis is on real products and realizable technology related to microprocessors and integrated circuits. Chip designers, computer architects, system engineers, press and analysts, plus attendees from national laboratories and academia will have the opportunity to see presentations on a variety of "hot" topics, including embedded and reconfigurable processors, quantum computing, nano structures, wireless chips, network/security processors, and advanced packaging technology.

Register today!

http://www.hotchips.org/

IEEE

IEEE computer society