

Virtualization Using AMD Servers and libVirt

Virtualization as a tool for development and deployment has come of age. Hardware advancements from manufacturers like AMD, and software tools such as libVirt, make creating and managing virtual machines a breeze. Let's look at how AMD, Red Hat, and others are helping push the virtualization envelope.

by Steve Schafer

Virtualization is a useful tool for development and enterprise solutions alike. For developers, the opportunity to use virtualized environments for various development and testing routines removes the necessity of having dedicated, physical hardware available for the same various environments. Instead of having dedicated Red Hat Enterprise, BSD, and Windows computers for testing, one physical machine can be used to run the environments virtually.

In the enterprise, the virtual machine is equally valuable—enabling in-use servers to be partitioned into virtual environments as the need for additional servers is realized, somewhat mitigating the need to purchase additional hardware to serve the purpose a VM can easily fulfill. Additionally, virtualization can allow older, and even obsolete, operating systems and applications to run on current hardware without porting or migration concerns. Lastly, virtual machines are extremely portable—moving or copying a VM between physical machines is trivial since the hardware is virtualized to the VM.

Until recently, virtual machines weren't as viable as tools and solutions. Although the tools existed for virtualizing hardware, the hardware itself wasn't powerful enough to provide enough resources for more than a few VMs. Also, the tools for creating VMs with adequate isolation and features weren't the easiest to understand and use. Thanks to companies like AMD and Red Hat, this is all about to change.

Virtualization in a Nutshell

Virtualization is the act of segmenting a computer's resources—CPU, I/O, memory, and other subsystems—into discrete pieces that look like a dedicated machine to applications. The segmentation is generally handled by a virtual machine manager (VMM) that isolates the VM from resources it shouldn't know about (like the rest of the memory not allocated to the VM) and acts as a translator to ensure that the VM's requests for resources are properly translated to requests the physical machine can handle. The more capable VMMs can detect and correct errors, as well as take predetermined actions when necessary.

For example, on a machine with 2GB of memory, you could set up a virtual machine that has access to only 512MB of memory. The VMM controls the access to the full 2GB, making the VM think it only has 512MB to work with and translating the requests for reads/writes to the proper segment of the full, physical 2GB.

Several virtual machines can be running on a single physical machine, each under the supervision of the VMM, and each blissfully ignorant of the other VMs running. A developer or system admin can run a variety of operating systems and applications on the same machine, even at the same time.

However, as the resource demands for operating systems and individual applications skyrocket, it becomes harder to segment a physical machine into large enough resource chunks that might be necessary to keep each VM well-fed and happy. Add in the overhead required by the virtual machine monitor, and the resource drain almost becomes too much for VMs to be useful.

AMD Servers—Ample Horsepower and VM Tools

In recent years the pure computing power available in a single, physical machine has outgrown the needs of most operating systems and applications. Two or more extremely fast processors, coupled with 10-20GB of fast RAM, provide plenty of power for a host operating system and most applications that run on top of the OS. In fact, with that amount of resources, several operating systems, running several apps apiece, can easily exist on the same physical platforms within VMs.

The underlying architecture of the latest machines makes sharing resources much easier than previous designs. The HyperTransport bus, for example, provides mega-bandwidth suitable for sharing across several VMs.

AMD has also addressed some of the VMM overhead issues with their new AMD-V™ architecture. The AMD-V architecture provides a set of instructions and architectural constructs that act as a super-privileged mode for the VMM to utilize for monitoring and control of the VMs running on the machine.

Moving a majority of the VMM monitoring and control overhead to hardware has the net effect of increasing the bandwidth available to the individual VMs.

Note: Additional information on AMD's AMD-V technology (previously dubbed AMD-SVM) can be found in [this article on DevX](#).

Red Hat Virtualization Tools

It's no surprise that the software-side of virtualization is realizing vast improvements as well. One of the leaders on the Linux side of the fence is libVirt, an open source project stewarded and driven by Red Hat, with contributions from Red Hat, IBM, Novell, Bull, VMware, and others.

LibVirt—Stability and Standards Through Simplicity

The libVirt project is a community-sponsored project that aims to bring more simplicity and standards to the Linux VM world. At its core, libVirt is a C toolkit that provides interaction with virtualization capabilities of the Linux operating system (and those related to Linux).

Note: The libVirt project can be found on the Web at www.libvirt.org.

The goal of libVirt is to provide the lowest possible generic and stable layer to manage VMs running on a machine. To accomplish this goal, libVirt will not try to be all things related to virtualization—instead libVirt will provide consistent and stable APIs to enable other tools to be built and used on top of the libVirt layer.

Although the premise for libVirt seems pretty simple, the project has turned out some very mature features and tools, including:

- Local administration tools—including a shell (virsh), a GNOME application, and a GNOME monitoring applet.
- Plenty of control interfaces—shell scripting, Python and Perl bindings, and robust APIs.
- Monitoring interfaces—feeding stats and states to applications, daemons, and the API hooks for other applications to utilize
- A robust policy framework—enabling complex policies to monitor, control, and correct domains running on the node.
- An XML structure for defining domains—portable, easily parsed, and human readable.

It's worth noting that libVirt and its APIs were used to create the various tools associated with the library. It's a case of "eating your own dog food" that actually works.

Compatibility

In the Linux space, [Xen](#) rules the VM roost. Xen supplies the VM framework and hypervisor utilities. The libVirt project built the libraries to interact with Xen, but not to be dependent upon it. In fact, many of the innovations present in the current build of libVirt are being adopted by Xen and will bring the two tools closer together.

A big advantage of libVirt's vendor-neutral stance is that you can define a framework for your VMs, applications, and policies that will run with most of the popular VMMs. Code once—a somewhat unique aspect in the development space.

Availability

The libVirt libraries are distributed in Fedora Core 5+, and will be available in future versions of Red Hat Enterprise Linux, starting with version V in the December timeframe. The [project Website](#) provides direct downloads of packaged RPMs, snapshot TARballs, and CVS access to the latest source.

The libVirt Lexicon

The [libVirt project](#) uses specific and unique terms to relate to the various pieces of the virtualization pie:

- A single, physical machine (or unit of a larger machine that can act as an independent entity) is referred to as a *node*.
- A virtual machine instance running on the node is referred to as a *domain*.
- A virtual machine manager running on the node is referred to as a *hypervisor* (which is another industry term for VMM).