System emulation with QEMU

The machine within the machine

M. Tim Jones (mtj@mtjones.com), Consultant Engineer, Emulex

Summary: QEMU is an open source emulator for complete PC systems. In addition to emulating a processor, QEMU permits emulation of all necessary subsystems, such as networking and video hardware. It also permits emulation of advanced concepts, such as symmetric multiprocessing systems (up to 255 CPUs) and other processor architectures, such as ARM or PowerPC. This article explores QEMU and its architecture and shows how to emulate a guest operating system on a Linux® host.

Date: 25 Sep 2007
Level: Introductory
Also available in: Russian Japanese Portuguese

Activity: 47355 views
Comments: 1 (View or add comments)

Average rating (based on 28 votes)

What is QEMU?

Saying that virtualization is a hot technology today is an understatement. Today, googling the word virtualization results in around 22 million hits. For example, in a single month, EMC Corporation announced an IPO for VMware, Citrix Systems announced plans to purchase XenSource, and new virtualization start-ups appeared out of nowhere. New niches are continually being found in what turns out to be an astronomically massive market. But with all the talk of initial public offerings and acquisitions in virtualization these days, it's easy to forget about some of the other virtualization technologies that are out there.

In this article, get acquainted with one of the more interesting virtualization applications not headlining the news today, QEMU. QEMU is an application that you can use in a number of settings. You can use it for guest operating system virtualization or as a full machine emulator running operating systems targeted to the host CPU or other CPU architectures.

Quick introduction to virtualization

Let's start with a quick introduction to virtualization to set the stage for QEMU.

Virtualization in the context of this article is really platform virtualization. On the physical hardware, a control program can be the host operating system or a hypervisor (see Figure 1). In some cases, the host operating system is the hypervisor. The guest operating systems reside on the hypervisor. In some cases, the guest operating system is targeted to the same CPU as the control program, but other cases may be different (for example, a PowerPC guest running on x86 hardware).

Figure 1. Basic architecture of platform virtualization
You can achieve virtualization in lots of ways, but three important methods are seen most often. The first is called *native (or full) virtualization*. In this variation, a hypervisor implements the basic elements of isolation, separating the physical hardware from the guest operating systems. This was first demonstrated in 1966 in the IBM® CP-40 virtual machine/virtual memory operating system and is also used by the VMware ESX Server.

Another popular virtualization technique is called *paravirtualization*. In paravirtualization, the control program implements a hypervisor application program interface (API) that is used by the guest operating system. Paravirtualization is used by both Xen and the Linux Kernel-based Virtual Machine (KVM).

A third useful technique is called *emulation*. Emulation, as the name implies, virtualizes the guest platform by simulating the complete hardware environment. Emulation is implemented in a variety of ways, even within the same solution. Examples of virtualization through emulation include QEMU and Bochs.

QEMU architecture

Let's look at how QEMU achieves emulation. This section describes the two QEMU modes of operation and also some of the interesting aspects of the QEMU dynamic translator.

Basic QEMU operation

QEMU supports two operating modes: user-mode emulation and system-mode emulation. *User-mode emulation* allows a process built for one CPU to be executed on another (performing dynamic translation of the instructions for the host CPU and converting Linux system calls appropriately). *System-mode emulation* allows emulation of a full system, including processor and assorted peripherals.

When x86 code is being emulated on an x86 host system, near native performance can be achieved using what is called the *QEMU accelerator*. This permits execution of the emulated code directly on the host CPU (on Linux through a kernel module).

But what makes QEMU interesting from a technical perspective is its fast and portable dynamic translator. The *dynamic translator* allows runtime conversion of instructions for a target (guest) CPU to the host CPU to provide emulation. This can be done in a brute force way (mapping instructions from one CPU to another), but it's not always that simple, and in some cases it can require multiple instructions or changes in behavior based on the architectures being translated.

QEMU achieves dynamic translation by first converting target instructions into *micro operations*. These micro operations are bits of C code that are compiled into objects. The core translator is then built. It maps the target instructions to micro operations for dynamic translation. This is not only efficient, but also portable.

QEMU's dynamic translator also caches blocks of translated code to minimize the translator's overhead. When a block of target code is first encountered, the block is translated and stored as a *translated block*. QEMU caches the most recently used translated blocks in a 16 MB block. QEMU can even support self-modifying code by invalidating translated blocks in the cache.
To learn more about the internal details of QEMU and its dynamic translator, check out the Resources section for an interesting paper by Fabrice Bellard (the author of QEMU).

Supported peripherals

Using QEMU as a PC system emulator provides a wide variety of peripherals. Standard peripherals that you expect include a hardware Video Graphics Array (VGA) emulator, PS/2 mouse and keyboard, integrated development environment (IDE) hard disk and CD-ROM interface, and floppy disk emulation. In addition, QEMU includes emulation for an NE2000 Peripheral Controller Interconnect (PCI) network adapter, serial ports, numerous sound cards, and a PCI Universal Host Controller Interface (UHCI) Universal Serial Bus (USB) controller (with a virtual USB hub). Processor symmetric multiprocessing (SMP) support is also provided with support for up to 255 CPUs.

In addition to emulating a standard PC or ISA PC (without a PCI bus), QEMU can also emulate other non-PC hardware such as the ARM Versatile baseboard (using the 926E) and the Malta million instructions per second (MIPS) board. Work is in progress for a variety of other platforms, including the Power Macintosh G3 (Blue & White) and Sun-4u.

Building and installing QEMU

Building and installing QEMU is about as simple as you expect using standard GNU tools. After downloading and untaring the QEMU distribution, configure, make, and then make install, and you're done (see Listing 1).

Listing 1. Building the QEMU emulator

```
$ wget http://fabrice.bellard.free.fr/qemu/qemu-0.9.0.tar.gz
$ tar xfvz qemu-0.9.0.tar.gz
$ cd qemu-0.9.0
$ ./configure
$ make
$ make install
$
```

This process creates not only an executable qemu image for the current target architecture but also a set of images for other architectures, including ARM, MIPS, PowerPC, 68k, and SPARC. With these, you can boot a Linux kernel that was built for a different target architecture.

If your host and guest operating system are targeted to the same processor architecture, then you can speed things up to near native performance using the QEMU accelerator (KQEMU). KQEMU is a driver (kernel module for Linux) that allows user-mode code and kernel code to execute directly on the host CPU. Building the QEMU accelerator is the same as building QEMU itself (see Listing 2).

Listing 2. Building the QEMU accelerator

```
$ wget http://fabrice.bellard.free.fr/qemu/kqemu-1.3.0pre11.tar.gz
$ tar xvfz kqemu-1.3.0pre11.tar.gz
$ cd kqemu-1.3.0pre11
$ ./configure
$ make
```
$ make install

You can compile or install KQEMU on a number of operating systems, including Microsoft® Windows®, FreeBSD®, and Linux. After you build the QEMU accelerator, install it in Linux using the following command:

$ insmod kqemu.ko
$
$ qemu -hda disk.img -cdrom /root/cflinux-1.0.iso -boot d
$

Using `qemu`, you specify your hard disk image with the `hda` option and the `cdrom` (file where your ISO image resides) with the `cdrom` option. The `boot` option specifies to boot from the CD-ROM. Argument `d` specifies to boot from the CD-ROM, where `a` boots from the floppy, `c` boots from the hard disk (the default), and `n` boots from the network. When this command is issued, a new QEMU window that represents the emulated machine appears (see Figure 2).

Figure 2. Preparing to install cfLinux onto an emulated disk with QEMU

Follow the installation instructions, per the CD-ROM install, to complete the installation of the ISO on the emulated hard disk. The install requests that you reboot. At this point, you can end the emulation (Ctrl-C in the `qemu` window). You can now boot your newly-installed operating system with the following command:

$ qemu -hda disk.img
$

This command line simply says to emulate a standard PC (the default option) with a hard disk represented by the `disk.img` image file. The Linux image boots from the emulated hard disk, resulting in the QEMU window, as shown in Figure 3.

Figure 3. Booting the newly-installed cfLinux from the emulated hard disk
That couldn't be much simpler. In fact, you can use the same sequence to install and boot any variety of operating system (production Linux distribution, Windows, or other).

Other emulators

Even though QEMU is a fantastic emulation environment, others are worth exploring. Wine is an open source implementation of the Windows API that allows you to run Windows programs without the Windows operating system. But as the Wine acronym explains, *Wine Is Not an Emulator*. Instead, Wine implements a set of APIs that permit execution of applications for the x86 architecture. Therefore, applications running on Wine perform well.

An emulator similar to QEMU is Bochs. Bochs is a machine emulator that emulates not only Intel®'s i386™, i486™, Pentium®, Pentium Pro, and Advanced Micro Devices' AMD64 CPUs, but also the common PC peripherals such as disks, memory, display, and networking devices. Bochs has been used to emulate Linux, DOS, and Windows 95/98/XP/2000/NT® operating systems.

Going further

Using QEMU as a machine emulator allows you to experiment with many operating systems for which you may not have a spare machine available. One example is ReactOS, which is an open source Windows XP-compatible operating system (whose emulation is shown in Figure 4). ReactOS aims for binary compatibility with Windows XP, so you can run applications built for Windows XP directly on ReactOS. See the Resources section for details on current application compatibility.

Figure 4. Emulating a standard PC for ReactOS

You'll find QEMU images for ReactOS and many other operating systems at the Free Operating Systems Zoo (see the Resources section for more details). These include live CD images, floppy images, or disk images (in qcow format). QEMU is a great way to try a new operating system without spending time for the install.

**Resources**

**Learn**

- See "QEMU, a Fast and Portable Dynamic Translator" (PDF) by Fabrice Bellard for a look at the gory internal details of QEMU dynamic translation.

- In "Virtual Linux" (developerWorks, December 2006), learn about other Linux virtualization options.

- "Discover the Linux Kernel Virtual Machine" (developerWorks, April 2007) covers the architecture of the Linux KVM as well as why its tight integration with the kernel may change the way you use Linux.

- In the developerWorks Linux zone, find more resources for Linux developers, and scan our most popular articles and tutorials.

- See all Linux tips and Linux tutorials on developerWorks.

- Stay current with developerWorks technical events and Webcasts.

**Get products and technologies**

- Download the QEMU open source processor emulator and QEMU accelerator from Fabrice Bellard's
Web site. There, you'll also find documentation, APIs, and the current status of QEMU.

- Learn more about cfLinux; the author used this small distribution to demonstrate installing an operating system into a QEMU machine.

- Get prepackaged images from the Free Operating Systems Zoo, and save yourself some time. At this site, you'll find many operating system images, from standard Linux distributions to other more exotic operating systems (Plan 9, OpenSolaris, MINUX, ReactOS, Darwin, MenuetOS, and others). After downloading one of these disk images, you'll have a disk image that's ready to boot.

- ReactOS is an open source Windows binary-compatible operating system that permits the execution of many Windows applications.

- Wine permits the execution of Windows applications on non-native Windows operating systems, such as Linux.

- Bochs is similar to QEMU in that it provides full system emulation.

- Order the SEK for Linux, a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

- With IBM trial software, available for download directly from developerWorks, build your next development project on Linux.

Discuss

- Get involved in the developerWorks community through blogs, forums, podcasts, and community topics in our new developerWorks spaces.

About the author

M. Tim Jones is an embedded software architect and the author of GNU/Linux Application Programming, AI Application Programming, and BSD Sockets Programming from a Multilanguage Perspective. His engineering background ranges from the development of kernels for geosynchronous spacecraft to embedded systems architecture and networking protocols development. Tim is a Consultant Engineer for Emulex Corp. in Longmont, Colorado.