Virtualization Assignment 1

Sen Li, Irving Gan

Introduction

QEMU is a fast processor emulator using dynamic translation to achieve good emulation speed. It is a free open-source alternative to VMware.

This document will walk through how to:

1. Use QEMU to create virtual disk images for DomU guests
2. Installing an operating system on the image.
3. Configure Xen guest to use the QEMU image file.
4. Launch a guest VM and manage its life cycle using Xen utilities

This document will also explain the differences between QEMU, KVM, Xen and the Advantages/Disadvantages of using QEMU. This document also assumes that you have Xen already installed in your machine.

Preparation

Before we proceed, we want to install a desktop environment. We are going to install a desktop environment. We are going to use a desktop environment called LXDE (Lightweight X11 Desktop Environment). It is free and it requires fewer resources. We also need to install XORG together with LXDE.

Install XORG :

$ sudo apt-get install xorg

Install LXDE:

$ sudo apt-get install lubuntu-desktop

After installing you can reboot the machine and there will be a GUI. The reason we install a GUI is because when booting the image we run into a problem of "*Could not initialize SDL(No available video device) – exiting "*. Therefore, using the GUI will remove this error for us.
We also want to install QEMU. QEMU comes default in Ubuntu. But if your system does not have it you can run this command:

$ sudo apt-get install qemu-kvm qemu

QEMU has two operating modes:

- User mode emulation: Launches Linux processes compiled for one CPU on another CPU.

- Full system emulation: Emulate a full system, including the processor and other peripherals like hard disk, CD-ROM, network adapters and other more.

For this document, we will just use QEMU to emulate our hard disk to put our operating system image in.

1. Use QEMU to provide create virtual disk images

We want to create a hard disk image with

$ qemu-img create <name_of_image>.img <size>

The size can be specified with EG: 1024k, 1024m, 1G, 1T. We can also specify the format using *-f*. However, only the formats "qcow2", "qed" and "vdi" support consistency checks. Right now, we will just use the default format instead which is the "RAW" format. This format has the advantage of being simple and easily exportable to all other emulators.

More information on other commands can be found on their man page:
https://linux.die.net/man/1/qemu-img

2. Installing an operating system on the image

First, we need to create a file to store our operating system. For this document, we will use the Ubuntu 14.04.6 LTS (Trusty Tahr) operating system.

Start by making a new folder for the operating system in the /var/lib/xen/images directory

$ mkdir -p /var/lib/xen/images/ubuntu-netboot/trusty

Now change into the directory using cd

$ cd /var/lib/xen/images/ubuntu-netboot/trusty

We want to download the iso for Ubuntu 16.04

$ wget http://releases.ubuntu.com/16.04/ubuntu-16.04.6-server-i386.iso

After we have downloaded the image, we can install the system on the image we created in step 1.

We can install the operating system on the disk image.

$ sudo qemu-system-x86_64 –m 1024 –hda <image_name>.img –cdrom /var/lib/xen/images/ubuntu-netboot//trusty/ubuntu-16.04.6-server-i386.iso

Once booted, follow the instructions to install the guest operating system into the QEMU disk image

3. Configuring Xen to use the QEMU image file

Once the installation is completed, we can try to boot the operating system again using QEMU using the QEMU image we created. Instead of booting it from the cdrom, we boot the image from the disk image we created using the *-boot c* option and removing the *-cdrom* option.

$ sudo qemu-system-x86_64 –m 1024 –hda <image_name>.img -boot c

Once the system has successfully booted, we can continue to install xen support into the guest OS.

Go into the /etc/xen directory and use the xlexample.pvlinux configuration template to configure our guest OS.

$ cd /etc/xen

$ cp xlexample.pvlinux <DomU-name>.cfg

Now edit the <DomU-name>.cfg

- Set the name to <whatever you want to call the vm>
- Comment out the kernel=”/boot/vmlnux”
- Comment out the extra=”root/dev/xvda1”
- Set the memory to 1024
- Set the vcpus to 1
- Added a line
    - *Bootloader = “/usr/lib/xen-4.4/bin/pygrub”*
- Set the disk to
    - *[‘file:<path-to-the-qemu-img>.img,had,rw’]*

Now save the file.

4. Launch a guest vm and manage its life cycle using Xen utilities

we can create the DomU using

$ sudo xl create -c /etc/xen/<DomU-name>.cfg

We can shut down the VM by using

$ sudo xl shutdown <name-of-vm>

Once the VM is started and we want to detach from it we can use *Ctrl +]* to exit. Once we want to connect to it again, we need to use

$ sudo xl console <name-of-vm>

You might need to hit the enter key a few times to show the VM prompt. More information of the xl commands can be found at https://xenbits.xen.org/docs/4.4-testing/man/xl.1.html

Differences between QEMU, KVM, Xen

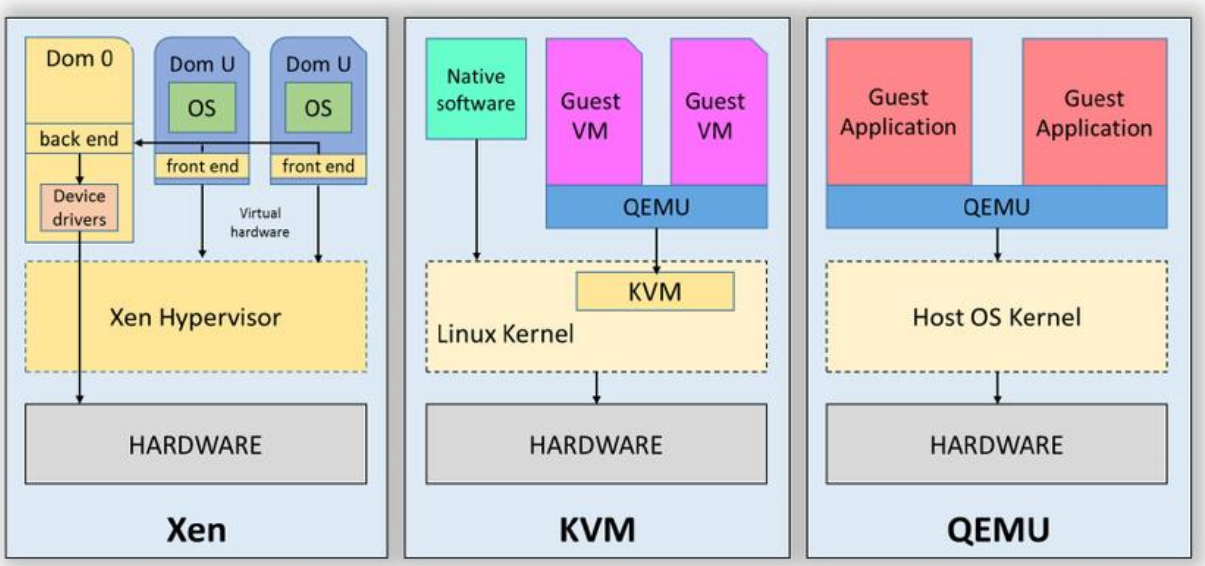

Figure 1: Comparison of Xen, KVM and QEMU

[Song, Yang & Wang, Haoliang & Soyata, Tolga. (2015). Hardware and Software Aspects of VM-Based Mobile-Cloud Offloading. 10.4018/978-1-4666-8662-5.ch008.]

QEMU: is a hardware emulator. It can be used to create a virtual machine. However, it is more often used under Xen or KVM to provide device virtualization to the guest. QEMU can provide simulation for PCI Bridge, VGA card, mouse/keyboard, hard disk, CD-ROM, network adapters, sound card, etc. For our case, we used QEMU to emulate our hard disk.

XEN: is a Type-1 hypervisor running on top of the hardware and is responsible for the resource management of the host machine. It uses para-virtualization, which allows near-native performance level. One big feature of Xen is that it has a specialized VM that has special privileges called Dom0 as seen in *Figure1.* Dom0 can handle resources and I/O access directly and allow control over the guest VM (DomU)

KVM: provides a complete virtualization environment where virtual machines appear as normal Linux Processes and integrate. KVM will be a Type 2 hypervisor when using it on top of the Host OS. lt makes the host Linux OS a type1 hypervisor as seen in Figure1. KVM was merged into Linux Kernel on version 2.6.20. KVM requires CPU virtualization extensions (Intel VT or AMD-V) and is used together with QEMU.

QEMU Advantages/Disadvantages

QEMU can simulate almost any hardware devices. It emulates the machine's processor through dynamic binary translation and provide a set of different hardware and device models for the machine and allow it to run a variety of guest OS. We used QEMU to create a disk image so Xen and use the image to boot up the guest VM.

Advantages

- Supports emulating IA-32 (x86) PCs, x86-64|AMD64 PCs, MIPS R4000, Suns SPARC sun3 and PowerPC (PReP and Power Macintosh) architectures
- Scalable to customize new instruction sets
- Open source, portable, fast simulation —some applications can run in close to real-time
- Emulate network card
- Support for running Linux binaries for other platforms

Disadvantages

- Incomplete support for Microsoft Windows and other host operating systems. However, the emulation of these systems works fine.
- Incomplete support for less frequently used architectures
- Stimulation speed is not as fast as other virtual software such as VMware. Unless using the KQEMU or KVM accelerator.
- Difficult to install compared to other simulation software

Why do we want to use QEMU along with Xen?

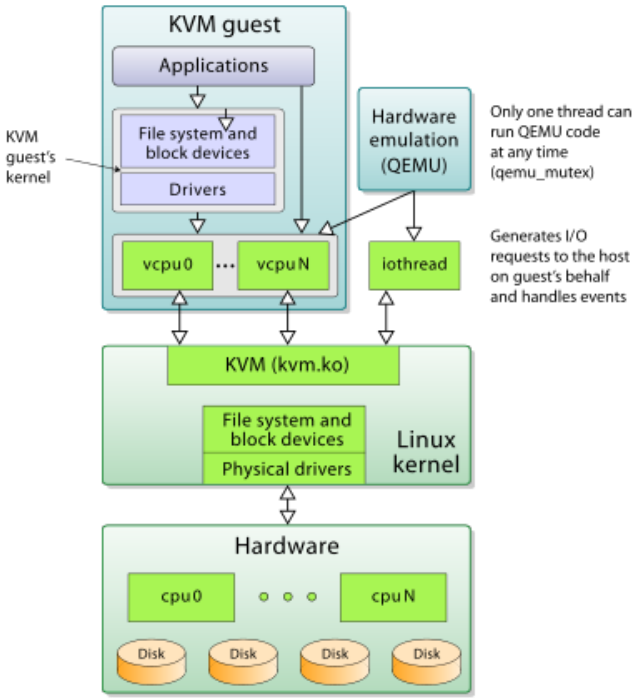

Figure 2 "KERNEL-BASED VIRTUAL MACHINE - WIKIPEDIA." HTTPS://EN.WIKIPEDIA.ORG/WIKI/KERNEL-BASED_VIRTUAL_MACHINE.

As mentioned above in the differences between Xen, QEMU and KVM. Xen offers a special privilege VM called Dom0 which can be useful when managing multiple guest OS. We used QEMU. Once there are multiple guest OS giving each guest OS its own set of hardware will not be cost-effective. Therefore, by using QEMU we can save cost by providing emulation of hardware using QEMU to those guest OS.

The virtual disk image we created can be store in a special format (qcow or qcow2) that only take up almost the same displace as the guest OS uses. Using that format an emulated 120GB disk may only occupy few hundred megabytes on the host. The QCOW2 format also provides the creation of overlay images which record the difference from another (unmodified) base image file. This allows the possibility of reverting the emulated disk's contents to an earlier state. In case of a system being corrupted, the user can revert to an earlier emulated disk image.

Depending on the situation, QEMU might have more functionality compare to Xen. For example, we might want to emulate a different CPU architecture for a specific program. Xen might not have the emulated CPU architecture while QEMU has.

Figure 2 shows how QEMU works together with the KVM. We can assume the Xen is the KVM in the diagram.

References

- Installation/QemuEmulator - https://help.ubuntu.com/community/Installation/QemuEmulator
- QEMU Wiki - http://www.damnsmalllinux.org/wiki/qemu.html
- Differences between QEMU/KVM/Xen - Song, Yang & Wang, Haoliang & Soyata, Tolga. (2015). Hardware and Software Aspects of VM-Based Mobile-Cloud Offloading. 10.4018/978-1-4666-8662-5.ch008.