Create a KVM-based virtual server

Three steps to build a virtual server on a Linux VM hypervisor with full virtualization

Skill Level: Intermediate

Da Shuang He (hedas@cn.ibm.com)
Software Engineer
IBM

19 Jan 2010

In three relatively simple steps, you can create a virtual server on the Linux® KVM hypervisor host using full virtualization. The Kernel-based Virtual Machine (KVM) is free, open source virtualization software for Linux that is based on hardware virtualization extensions (Intel VT-X and AMD-V) and a modified version of QEMU.

Join the green groups on My developerWorks
Discuss topics and share resources about energy, efficiency, and the environment on the GReen IT Report space and the Green computing group on My developerWorks.

The Linux Kernel-based Virtual Machine (KVM) is free, open source virtualization software for Linux based on the Intel VT-X and AMD-V hardware virtualization extensions and a modified version of QEMU (work is underway to get the required changes upstream). KVM—in the form of kvm.ko, a loadable kernel module that provides the core virtualization infrastructure and processor-specific modules kvm-intel.ko and kvm-amd.ko—is designed to enable full hardware emulation as far as needed to boot many PC operating systems in unmodified form.

Using KVM, you can run multiple virtual machines that themselves are running unmodified Linux or Windows® or Mac OS® X images. Each virtual machine has private virtualized hardware such as a network card, disk, graphics adapter, etc.

For this article, I used an IBM® Blade Server HS21 with SUSE 11 as the operating system. The HS21 supports the Intel VT extension and the kernel version of SUSE

11 is 2.6.27.13, which already contains KVM (KVM is included in Linux kernel versions from 2.6.20).

The three main steps to get your virtual server going are:

1. Install the operating system and required software.
2. Create the virtual server.
3. Configure the virtual server network.

Step 1. Install the OS and required software

This section covers:

1. Installing the operating system and required software
2. Determining whether the CPU supports KVM
3. Making sure the software is installed correctly

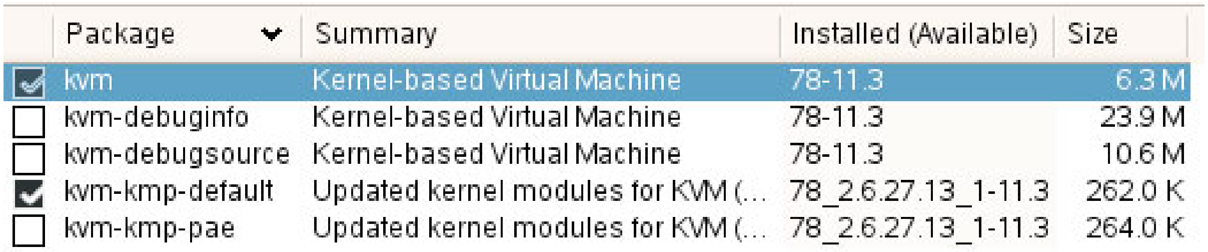
Install operating system and software

After you install the operating system, you can find the installed version of the Linux kernel with the following command:

```
kvm:~ # uname -a
Linux kvm 2.6.27.13-1-pae #1 SMP 2009-01-27 13:41:16 +0100 i686 i686 i386 GNU/Linux
```

Then select and install the kvm and kvm-kmp-default packages (which already include a modified QEMU for I/O hardware emulation). As shown in Figure 1, select these packages and click **Accept** to install them:

Figure 1. Finding the installed kernel



	Package	Summary	Installed (Available)	Size
✓	kvm	Kernel-based Virtual Machine	78-11.3	6.3 M
☐	kvm-debuginfo	Kernel-based Virtual Machine	78-11.3	23.9 M
☐	kvm-debugsource	Kernel-based Virtual Machine	78-11.3	10.6 M
✓	kvm-kmp-default	Updated kernel modules for KVM (...	78_2.6.27.13_1-11.3	262.0 K
☐	kvm-kmp-pae	Updated kernel modules for KVM (...	78_2.6.27.13_1-11.3	264.0 K

Now, the kvm-kmp-default version should be 78_2.6.27.13_1-11.3. (78 is the KVM version, and the rest of the information indicates the kernel version.)

A typical KVM installation consists of these components:

- A device driver for managing the virtualization hardware; this driver exposes its capabilities via a character device /dev/kvm
- A user-space component for emulating PC hardware; currently, this is handled in the user space and is a lightly modified QEMU process
- The I/O model, which is directly derived from QEMU's model with support for copy-on-write disk images and other QEMU features

Determine whether the CPU supports KVM

KVM depends on the x86 virtualization extensions. To check for compatibility, run the command `grep vmx /proc/cpuinfo` (on AMD, run the command `grep svm /proc/cpuinfo`). If the output is similar to Listing 1, then the CPU supports KVM; otherwise, your CPU does not support KVM.

Listing 1. Checking CPU for KVM support

```
kvm:~ # grep vmx /proc/cpuinfo
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe lm constant_tsc arch_perfmon pebs bts pni
monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr dca lahf_lm
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe lm constant_tsc arch_perfmon pebs bts pni
monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr dca lahf_lm
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe lm constant_tsc arch_perfmon pebs bts pni
monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr dca lahf_lm
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe lm constant_tsc arch_perfmon pebs bts pni
monitor ds_cpl vmx est tm2 ssse3 cx16 xtpr dca lahf_lm
```

Determine that the software is successfully installed

Run the command `lsmod | grep kvm` to check whether the KVM module is installed successfully. If the result is similar to this output, then the KVM module is installed successfully:

```
kvm:~ # lsmod | grep kvm
kvm_intel              42604  0
kvm                   150264  1 kvm_intel
```

Step 2. Create the virtual server

This section covers:

1. Creating the raw disk image
2. Installing the OS on this image
3. Firing up the virtual server

Create a raw disk image

To create a raw disk image, use this command:

```
dd if=/dev/zero of=/mnt/kvmtest.img bs=1024 count=0
seek=$[10*1024*1024]
```

With this command, you'll create a 10GB image with the name of kvmtest.img.

Install the operating system on this image

To get the operating system installed on this image, use this command:

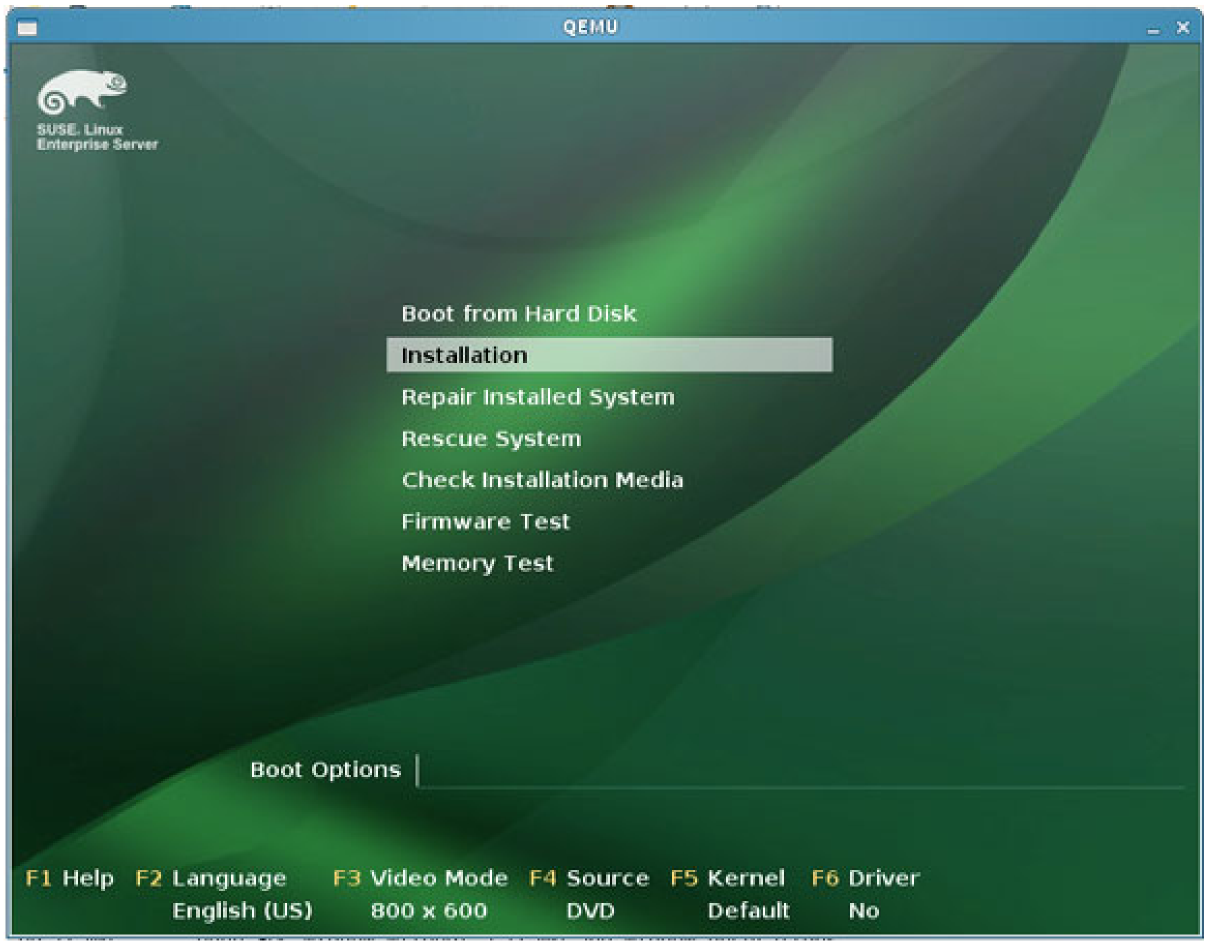
```
/usr/bin/qemu-kvm -hda /mnt/kvmtest.img -boot d -cdrom
/mnt/SLES-11-DVD-i586-RC3-DVD1.iso -m 1024
```

Since you're installing a KVM package and not building KVM yourself from source, you can use `qemu-kvm` but not `qemu-system-x86_64`.

`-boot d` means we will make the virtual server boot from the CDROM. `-m 1024` means we specify 1GB memory for the virtual server.

After running this command, the operating system installation screen should look like Figure 2:

Figure 2. The OS installation screen



Next, install the operating system as you normally would.

Now, fire up your virtual server

After the operating system is successfully installed, you can start the virtual server with this command:

```
/usr/bin/qemu-kvm -hda /mnt/kvmtest.img -m 1024
```

Your server should run just like a normal server running on the physical hardware.

Step 3. Configure the virtual server network

You've now seen how to successfully create a virtual server, but not the network for the virtual server. Now I'll show you how to create a bridge network for the virtual server:

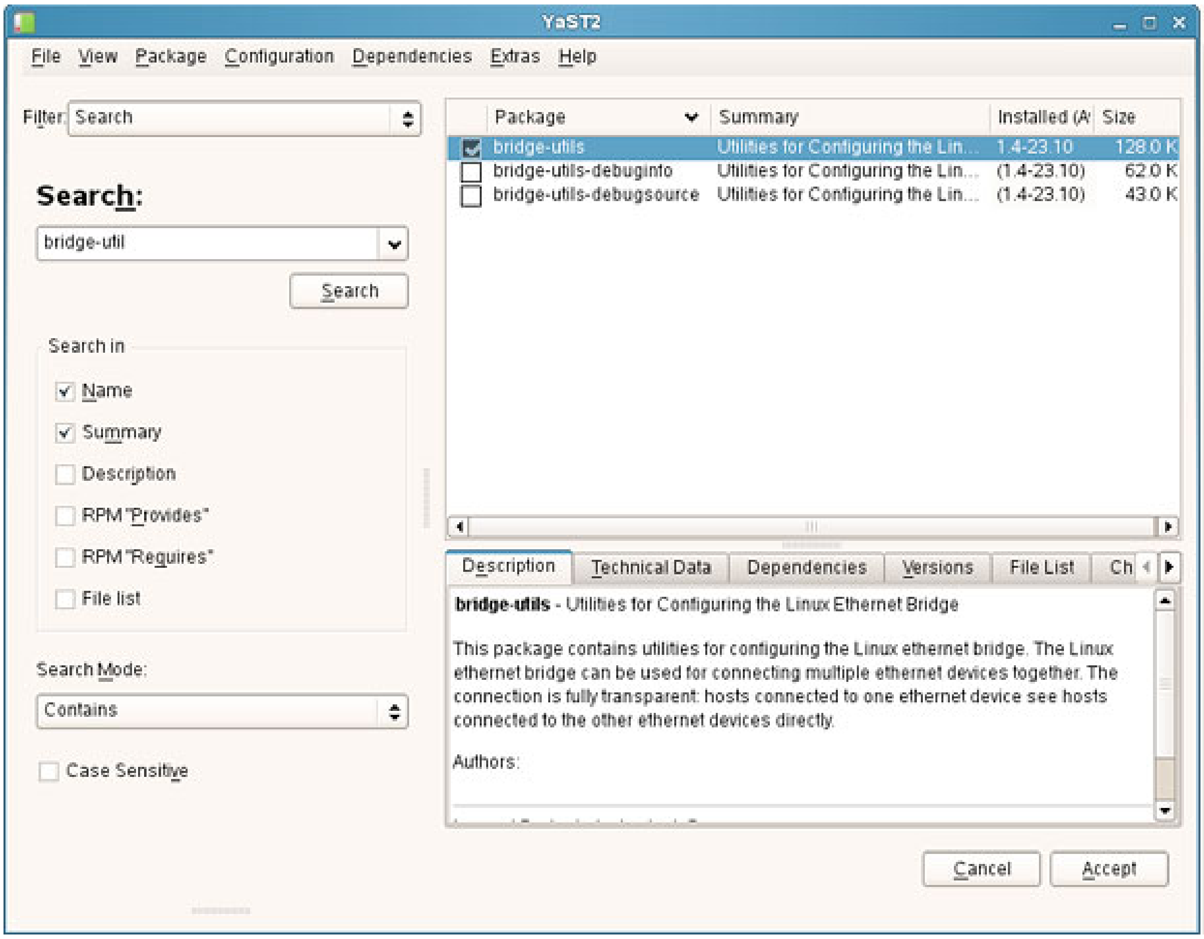
1. Installing the bridge-util package

2. Creating the bridge network interface
3. Creating the script for network control
4. Starting the guest operating system network
5. Configuring the guest operating system network

The bridge-util package install

We need to create a bridge network, so we need to install the bridge-util package in the software management section of the operating system. Figure 3 shows where to select and install the package.

Figure 3. Installing the bridge-util package



	Package	Summary	Installed (A	Size
✓	bridge-utils	Utilities for Configuring the Lin...	1.4-23.10	128.0 K
	bridge-utils-debuginfo	Utilities for Configuring the Lin...	(1.4-23.10)	62.0 K
	bridge-utils-debugsource	Utilities for Configuring the Lin...	(1.4-23.10)	43.0 K

bridge-utils - Utilities for Configuring the Linux Ethernet Bridge

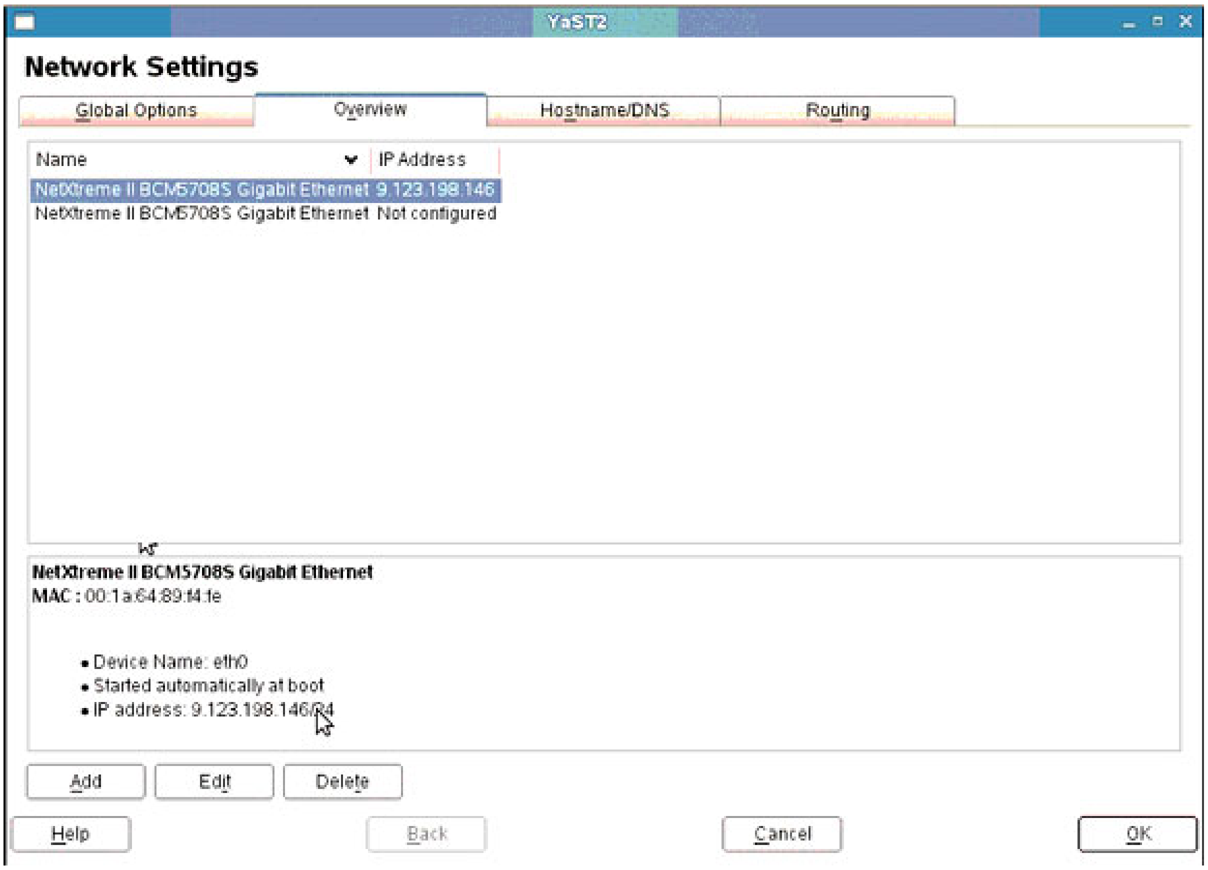
This package contains utilities for configuring the Linux ethernet bridge. The Linux ethernet bridge can be used for connecting multiple ethernet devices together. The connection is fully transparent: hosts connected to one ethernet device see hosts connected to the other ethernet devices directly.

Authors:

Build the bridge network interface

In the network configuration screen of the operating system, click **Add a new network interface** (see Figure 4):

Figure 4. Adding a new network interface



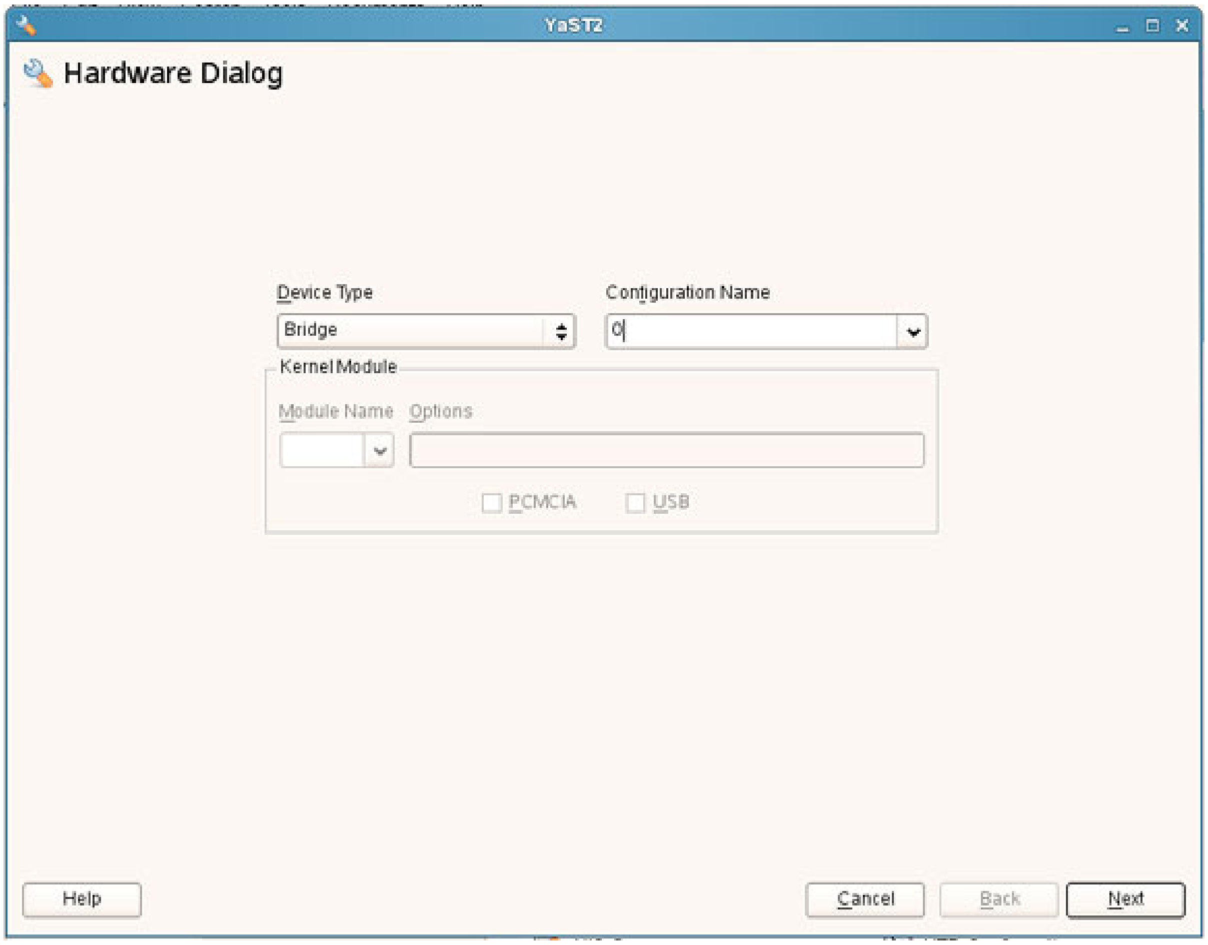
Name	IP Address
NetXtreme II BCM5708S Gigabit Ethernet	9.123.198.146
NetXtreme II BCM5708S Gigabit Ethernet	Not configured

NetXtreme II BCM5708S Gigabit Ethernet
MAC : 00:1a:64:89:f4:1e

- Device Name: eth0
- Started automatically at boot
- IP address: 9.123.198.146/24

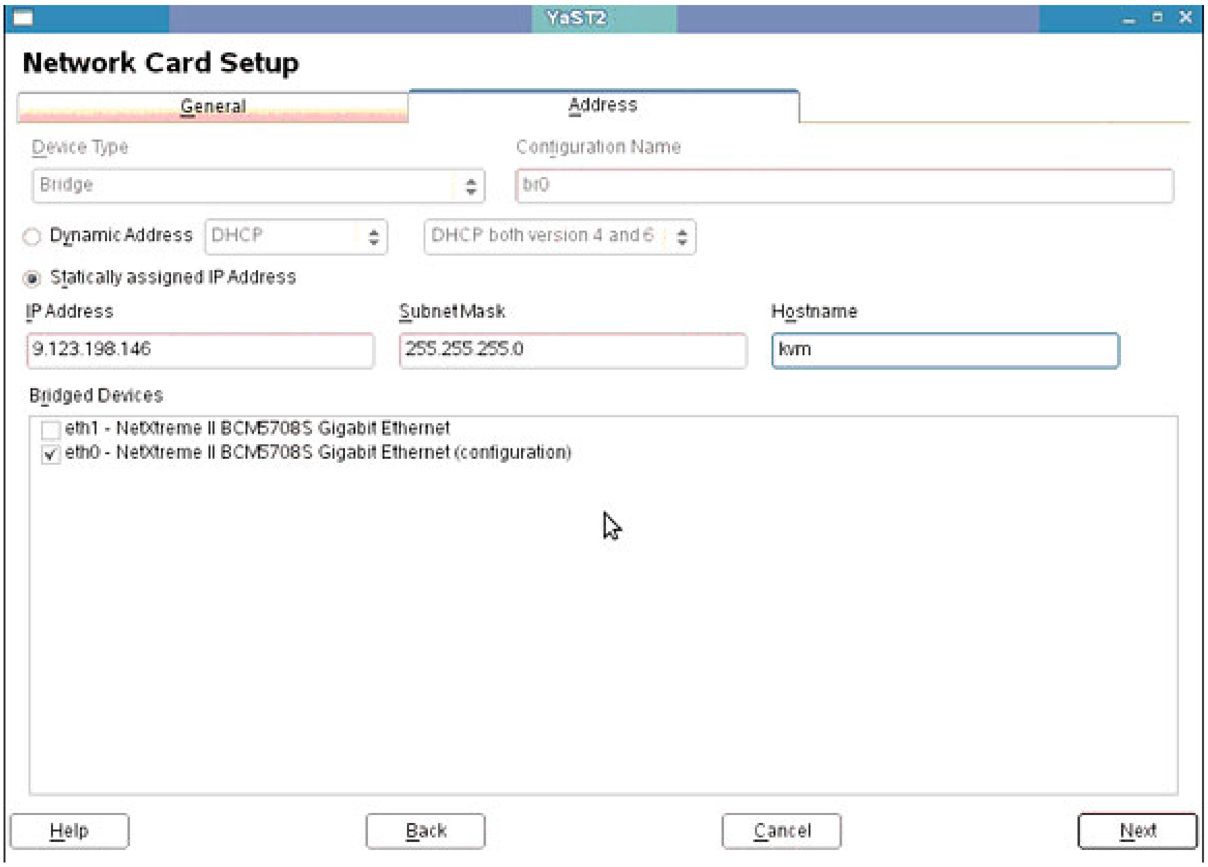
You should get a Hardware Dialog as shown in Figure 5:

Figure 5. The hardware dialog for new network interface



Choose the **Bridge** device type; click **Next** to continue the network configuration. You'll see the configuration screen shown in Figure 6:

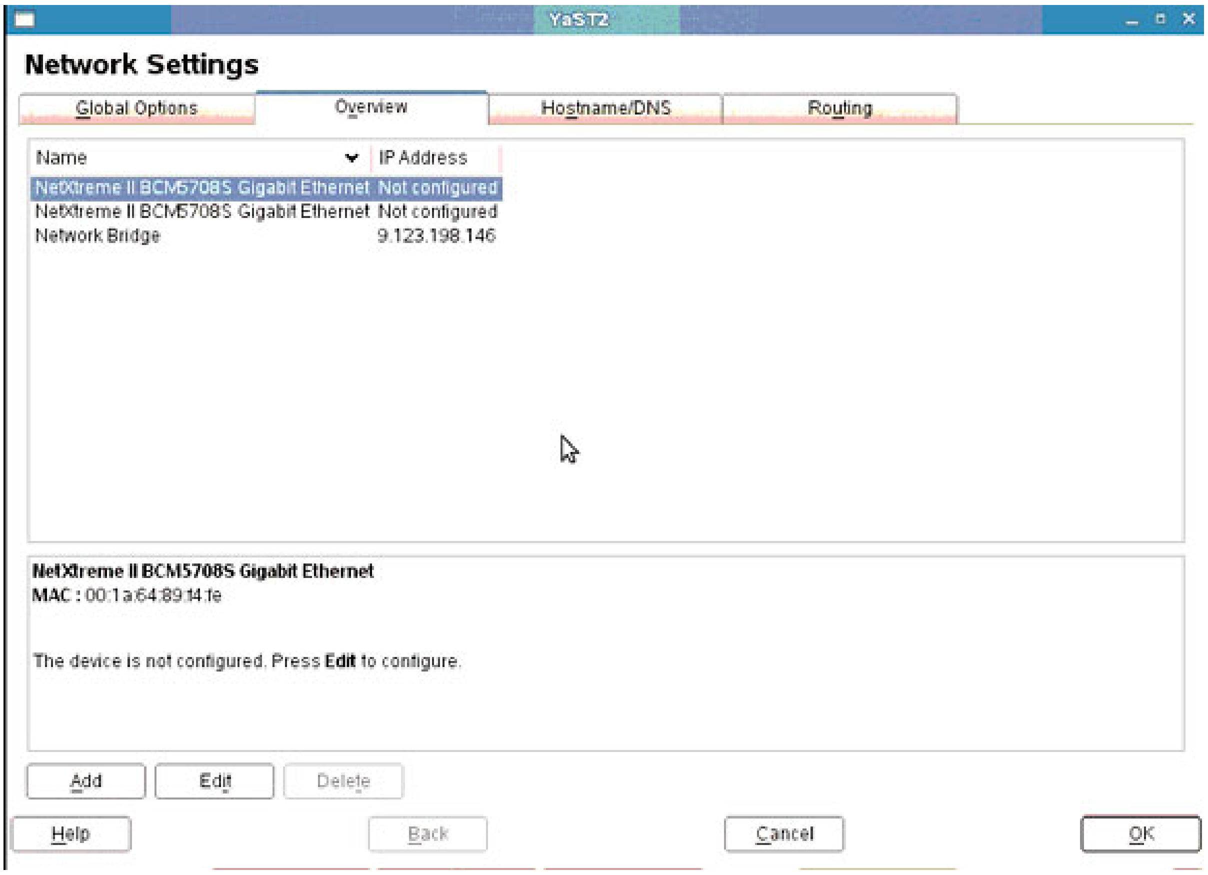
Figure 6. Network configuration screen



Device Type: Bridge
Configuration Name: br0
( ) Dynamic Address: DHCP — DHCP both version 4 and 6
(●) Statically assigned IP Address
IP Address: 9.123.198.146
Subnet Mask: 255.255.255.0
Hostname: kvm
Bridged Devices:
[ ] eth1 - NetXtreme II BCM5708S Gigabit Ethernet
[✓] eth0 - NetXtreme II BCM5708S Gigabit Ethernet (configuration)

Choose **eth0** under "Bridged Devices" and configure a static IP for the bridge interface, which is the same configuration of *eth0* (see Figure 7).

Figure 7. Configuring a static IP for the bridge interface



Name	IP Address
NetXtreme II BCM5708S Gigabit Ethernet	Not configured
NetXtreme II BCM5708S Gigabit Ethernet	Not configured
Network Bridge	9.123.198.146

NetXtreme II BCM5708S Gigabit Ethernet
MAC : 00:1a:64:89:14:fe

The device is not configured. Press **Edit** to configure.

After you do that, the *eth0* configuration will be cleared, so click **Next** to continue, and you are back to the network configuration screen. You can see a new bridge network interface has been created and the *eth0* configuration is cleared.

Build the network control script

The content for a network control script should look something like Listing 2:

Listing 2. Network control script

```
#!/bin/sh
set -x
switch=br0
if [ -n "$1" ];then
        tunctl -u `whoami` -t $1
        ip link set $1 up
        sleep 0.5s
        brctl addif $switch $1
        exit 0
else
        echo "Error: no interface specified"
        exit 1
fi
```

Start the guest operating system

Start the guest operating system with a network interface using the command:

```
/usr/bin/qemu-kvm -hda /mnt/kvmtest.img -m 1024 -net
nic,macaddr=52:54:00:12:34:56 -net tap,script=/etc/qemu-ifup
```

You'll specify a network interface when you start the virtual server.

Configure the guest operating system network

After the guest operating system is booted, configure its network as you would normally.

Congratulations! You've successfully created a virtual server built on KVM.

Resources

Learn

- Learn more about QEMU, the generic and open source machine emulator and virtualizer.
- Get details on KVM too.
- The developerWorks podcast "David Ashley on a build environment with the Linux KVM" (developerWorks, July 2009) talks about crafting an on-demand software build service using the flexible KVM.
- Two good sources for using KVM are "Discover the Linux Kernel Virtual Machine" (developerWorks, April 2007), which discusses the KVM architecture and how its tight integration with the kernel can change the way you use Linux; and "Create an ooRexx build environment on Linux KVM" (developerWorks, July 2009), a hands-on experience that shows you how to use KVM to build a build system (goes with the podcast mentioned in the previous resource).
- In the developerWorks Linux zone, find more resources for Linux developers, and scan our most popular articles and tutorials.
- See all Linux articles and Linux tutorials on developerWorks.
- Stay current with developerWorks technical events and webcasts.

Get products and technologies

- IBM offers tons of documentation and support for its BladeCenter servers.
- With IBM trial software, available for download directly from developerWorks, build your next development project on Linux.

Discuss

- Get involved in the My developerWorks community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.

About the author

Da Shuang He
Da Shuang He is a software engineer at the IBM Development Lab in Shanghai, China. He is currently working on system management software and focuses on System x and modular management software development. He also has rich experience and knowledge of virtualization, cloud computing, green computing, and Linux high availability technology.