Booting Palacios/Kitten and Palacios/Linux
Over the Network Using PXE

Yuan Tang Lei Xia

November 11, 2011

PXE allows us to boot operating systems over the network. This is incredibly useful for debugging Palacios in the context of different hosts on real hardware. In this document, we focus on the following common Palacios configurations:

- *Palacios/Kitten*: Kitten with Palacios built in, and a test guest built into the Kitten init task.
- *Palacios/Linux*: Linux with the Palacios kernel module, userspace tools, and one or more test guests residing on an initial RAM disk.

The following gives instructions on how to set up the PXE server and client. Although we explain this specifically in the context of the above use cases in Kitten and Linux, PXE can be used to network boot most operating systems, and the configuration can be more complex than we describe.

We assume there are two machines:

- *Server:* This machine will serve a Kitten or Linux kernel over the network to client machines.
- *Client:* This machine will request the Kitten or Linux kernels from the server machine.

1 How does it work?

When the client machine starts up, PXE will first make a DHCP request to retrieve a temporary IP configuration. The server will respond to this request. Part of the response will indicate that the client should reconnect back to the server (this time using TCP or UDP) to retrieve more data using FTP. The client will do so. The server will respond by sending a small bootloader, the kernel, and additional files. The client copies this content into relevant memory locations and jumps to it. The bootloader then boots the kernel. In the typical case of Palacios/Kitten, the kernel contains Kitten and Palacios, while the additional file contains the init task, which in turn has the guest image that's being tested embedded in it. Similarly, in the case of Palacios/Linux, the kernel is host Linux kernel, while the additional file is the archived host initramfs, which contains Palacios kernel module file, one or more guest images, and user level tools for controlling Palacios.

2 What will we boot?

PXE can be used to build quite complex boot paths. For testing Palacios on either Kitten or Linux host OSes, there are typically only two files involved, however:

- *bzImage*: When you boot Palacios/Kitten, this is the Kitten kernel with Palacios built in, typically located at `your_kitten_path/arch/x86_64/boot/bzImage`. If you boot Palacios/Linux,then this is your host kernel image file.
- *init_task*: If you boot Palacios/Kitten, then this should be Kitten init task, which also contains the guest image that will be used for testing. It is typically located at `your_kitten_path/init_task`. If you boot Palacios/Linux, then this is the cpio archive file of your host initramfs[1].

For booting, we copy these files into `/tftpboot` on the server.

3 Client requirements

PXE is a service that is implemented in the client machine's BIOS, the network card's BIOS extensions, or possibly in a boot ROM or CD created using a tool such as ROM-O-Matic. Most modern machines, even inexpensive desktops and laptops, support it but have it turned off by default. To turn it on, power cycle the client and enter the BIOS. There may be an option to enable it here, probably under local devices/networking. If you can't find an option there, power cycle again and wait for the BIOS bootstrap message about your network card. There should be an option to hit a key to go into its configuration screen, in which you should be able to turn on PXE.

4 Server requirements

- DHCP server. DHCP enables the client get an IP address, and other configuration data automatically from the server.
- TFTP server. After the client has an IP address, it will connect back to the TFTP server it to download the kernel and other files from the server.
- Firewall rules that allow access from the client to the DHCP server and TFTP server. DHCP typically requires ports 67 and 68, while TFTP typically uses port 69.
- PXELINUX boot loader (`pxelinux.0`) installed in `/tftpboot`. You may also want to have `memdisk` in the same directory if you need to boot more ancient things.

5 DHCP setup on the server

Install the DHCP service, and configure the file */etc/dhcpd.conf*. You want to add the client's Ethernet MAC address to config file, allow the client to get an IP address/config from DHCP server at boot time. Here is an example configuration:

```
option domain-name "cs.northwestern.edu";
default-lease-time 600;
max-lease-time 7200;
```

[1] You can refer the document "An Introduction to the Palacios Virtual Machine Monitor-Version 1.3" to learn how you can build your bzImage and initramfs archive file for your host Linux.

```
allow booting;
allow bootp;

<-- Change below section according
    to your network configuration --->
subnet 165.124.184.0 netmask 255.255.254.0 {
    range 165.124.184.90 165.124.184.95;
    option broadcast-address 165.124.184.255;
    option routers 165.124.184.1;
    option domain-name-servers 165.124.180.10;
}

group {
  <--- In which IP address the booting machine can
       find the booting image -->
  next-server 165.124.184.209;
  host v-test-intel-lab {
    <-- Replace below with the IP address assigned
        to the booting machine -->
    fixed-address 165.124.184.92;
    <--- Replace below with the MAC address of the booting machine -->
    hardware ethernet  00:1B:21:41:CF:4C;
    filename "pxelinux.0";
  }
}
```

Notice that you can readily add additional client machines by adding new `host` entries.

Now you can start the DHCP server.

```
[jdoe@newskysaw test]$ service dhcpd restart
```

6 TFTP Setup

After installing the tftp service, enable it by editing the file */etc/xinetd.d/tftp* so that it looks similar to the following. Typically you can just delete the line *disable = yes*.

```
service tftp
{
    socket_type             = dgram
    protocol                = udp
    wait                    = yes
    user                    = root
    server                  = /usr/sbin/in.tftpd
    server_args             = -s /tftpboot
    per_source              = 11
    cps                     = 100 2
```

```
        flags                   = IPv4
    }
```

You can now copy the kernel and init task files to tftp boot directory, which typically is */tftpboot*. For Palacios/Kitten, this looks like:

```
[jdoe@newskysaw test]$ cp your_kitten_path/arch/x86_64/boot/bzImage  \
                             /tftpboot/bzImage
[jdoe@newskysaw test]$ cp your_kitten_path/init_task \
                             /tftpboot/init_task
```

And for Palacios/Linux, typically:

```
[jdoe@newskysaw test]$ cp host_linux_src/arch/x86/boot/bzImage  \
                             /tftpboot/bzImage
[jdoe@newskysaw test]$ cp /path/to/initrd/initramfs.cpio \
                             /tftpboot/initramfs.cpio
```

You will also want to be sure that you have copied `pxelinux.0` and `memdisk` to `/tftpboot/`

You can now start the TFTP service:

```
    service xinetd restart
```

7 PXE Configuration

We will now set up the PXE boot configuration. Create the directory *pxelinux.cfg* in */tftpboot*. In *pxelinux.cfg*, create a new file with the filename as the client's MAC address, prefaced with `01-`:

```
[jdoe@newskysaw test]$ cd /tftpboot
[jdoe@newskysaw test]$ cd pxelinux.cfg
[jdoe@newskysaw test]$ touch 01-00-1b-21-41-cf-4c
```

Edit this new configuration file (*01-00-1b-21-41-cf-4c* here), to set up a boot option for the machine. For Palacios/Kitten testing, the option typically looks like this:

```
    default Palacios_Kitten
    label Palacios_Kitten
        kernel bzImage
        append serial.baud=115200 console=serial initrd=init_task

    prompt 1
    timeout 20
```

For Palacios/Linux the option typically looks like this:[2]:

[2] The "movablecore" parameter specifies the maximum amount of memory that Palacios can allocate for guest physical memory use. This is amount of physical memory that may be to be "hot-removed" from Linux use for Palacios to use later. You can put any Linux boot commands here, for example, "console=ttyS0,115200" if you prefer to use the serial console for your host Linux.

```
    label Palacios_Linux
        kernel bzImage
        append movablecore=262144 initrd=initramfs.cpio console=tty0
```

Notice that your kernel bzImage file, and the initramfs cpio file are both referenced. These are the copies that are /tftpboot. You can have multiple options in the configuration file, each with a different label. You can learn more about these configurations by reading about PXELINUX.

You can readily add additional machines simply by creating files reflecting their MAC addresses. You can also configure PXE to support multiple options for each machine. This allows for one PXE server to support numerous test machines and developers.

8 Testing

At this point things should be ready to go. Power on the client machine, select network booting if prompted. PXE should then go searching for the DHCP server, find our server, configure, and then continue the boot process via TFTP. Within seconds, you will see the Kitten or Linux kernel start to boot.

If the client fails to connect to the PXE server, you should check the firewall settings, making sure they allow the client's connection requests. Also, make sure the DHCP and TFTP services are actually running.

9 Serial

For testing, it is usually a good idea to connect the client machine to some other machine via the serial port. Palacios and its host OS produce debugging and logging output via serial communication. To connect two machines, you will want a null modem RS232 cable. The host OSes typically use 115200 bps communication. If your machine's serial port is /dev/ttyS0, then execute the following to see the output:

```
stty -F /dev/ttyS0 115200
cat /dev/ttyS0
```

Typical machines with a single port on the machine typically actually have a dual UART. If you have such a machine, you may want to try `/dev/ttyS1` if `/dev/ttyS0` doesn't work.

We have also experienced some challenges using SIIG multiport serial interfaces, but these can usually be cleared up by configuring them using:

```
setserial /dev/ttySn uart 16950 baud_base 115200
```