Root over NFS on User Mode Linux

Giorgos Kappes
Dep. of Computer Science,
University of Ioannina
geokapp@gmail.com

April 8, 2012

Abstract

The boot disk of a UML instance is usually a file in the host's file system known as the disk image. A disk image is a single file containing the complete contents and structure representing a data storage medium or device, such as a hard drive and hence it provides the same block interface as physical hard disks. Although virtual disks that reside on the host system are used widely by virtual machine guests, there are many times when their use is not feasible. For example, if the guests' boot file system does not reside on the host but on a remote server then the guests have no other option than to boot with their root file system on NFS. In this tutorial we will be looking at creating a root file system for UML guests on NFS mounted file system. In many ways this is little different from other techniques such as the use of virtual disks, although similar techniques can be used to populate the root file system. The main difference comes at the point that the UML guest is booted: instead of running on a local file system, the root file system for our UML guests is located on a remote server.

1 Introduction

The root file system is the file system that is directly mounted by the kernel during the boot phase and that holds the system initialization scripts and the most essential system programs. More specifically, the root file system includes the root directory together with a minimal set of subdirectories and files including "/boot", "/dev", "/etc", "/bin", "/sbin" and sometimes "/tmp". Mounting the root file system is a crucial part of system initialization. The Linux kernel allows the root file system to be stored in many different places [1], such as a hard disk partition, a floppy disk, a ramdisk or a remote file system shared via NFS. In any case, the root file system can be specified as a device file in the “/dev” directory either when compiling the kernel or by passing a suitable “root=” option to the initial bootstrap loader. The root file system is mounted in a two-stage procedure:

1. The kernel mounts the special “rootfs” file system, which provides an empty directory.
2. The kernel mounts the real root file system over this empty directory.

The root file system of a UML guest can be stored on a disk image in the host system, on a disk partition of the host system, or in a directory hierarchy in the host system which can be exported to the client via humfs or NFS. This guide follows the last approach. More specifically, the root file system for the UML guest will be located on an exported directory hierarchy stored in a NFS server. This server could be the host system, a remote machine or even another UML guest. Our goal is the UML guest to mount this exported directory hierarchy as its root file system via NFS.

2 Working environment

Our working environment consists of the host system (“jurassic”) with IP address “192.168.1.200” and of a UML guest (“uml-client”) with IP address “192.168.2.2”. The “uml-client” guest can see our home network through a virtual bridge with IP address “192.168.2.1” which resides on the host. The host system will act as an NFS server and will provide the root file system to the guests. Thus, our goal is the “uml-client” to mount the server’s exported root file system hierarchy as its root file system via NFS. Note that the root file system can also be stored on a remote machine or even in a UML guest which acts as sever.

3 Server configuration

At this point we will analyze the necessary steps that need to be done on the server side. Initially, we will focus on the server’s kernel configuration and then we will see how to create the root file system for a guest and how to export it.

3.1 Kernel configuration

On the server side we will need to compile NFS server support into the kernel. More specifically, we have to compile a new kernel with the following options enabled as buildin:

```
Network File Systems ->
  NFS client support - enabled
  NFS client support for NFS version 3 - enabled
  NFS client support for the NFSv3 ACL protocol expansion - enabed
  NFS client support for NFS version 4 - enabled
  Use the new idmapper upcall routine - enabled
  NFS server support - enabled
  NFS server support for NFS version 3 - enabled
  NFS server support for the NFSv3 ACL protocol expansion - enabed
  NFS server support for NFS version 4 - enabled
```

Assuming our server's file system is ext4, we should also select the following options if we wish to use NFSv4's extended attributes and ACL features:

```
File Systems ->
The extended 4 (ext4) filesystem
Ext4 Extended attributes - enabled
Ext4 POSIX Access Control Lists - enabled
```

3.2 Creation of the guest's root file system

In this section we will create a root file system for our guest UML. Let 's assume that we have an NFS server already configured and running. For a guide on how to set up an NFS server look on [2]. Firstly, we create the directory "/export" if it doesn't already exist:

```
root@host$ mkdir /export
```

Then, we create a new directory under "/export" to place the guest's root file system:

```
root@host$ mkdir /export/uml-client
```

Now, we can populate the guest's root file system either by copying from the host's root:

```
root@host$ cp -ax /{root,dev,var,etc,usr,bin,sbin,lib}
           /export/uml-client
root@host$ mkdir /export/uml-client/{proc,sys,home,tmp}
```

or by installing a base Linux distribution, for example Debian Squeeze:

```
root@host$ debootstrap --arch i386 squeeze /export/uml-client
http://ftp.us.debian.org/debian
```

Next, we tailor the file system by editing "/etc/fstab", "/etc/hostname", "/etc/ network/ interfaces" etc and adding some user accounts. For a guide on how to set up a root file system for a UML guest look on [3]. Note that "/etc/fstab" and "/dev/" need some caution. In "etc/fstab" we have to specify the coorect mount point of the root file system, which in our case resides on a NFS server. Assuming that our NFS server is on 192.168.2.1, the guest's "/etc/fstab" must look like this:

```
192.168.2.1:/          /        nfs4    rw               0 0
proc                  /proc     proc    defaults         0 1
```

Furthermore, we have to create a dummy device on the guest's "/dev" directory:

```
root@host$ chroot /export/uml-client
root@host$ mknod /dev/nfs b 0 255
root@host$ chown root:disk /dev/nfs
root@host$ exit
```

Also, the guest's "/etc/network/interfaces" file must look like this:

```
# Used by ifup(8) and ifdown(8). See the interfaces(5) manpage or
# /usr/share/doc/ifupdown/examples for more information.
auto lo
iface lo inet loopback4

auto eth0
iface eth0 inet static
        address 192.168.2.2
        netmask 255.255.255.0
        network 192.168.2. 0
        broadcast 192.168.2. 2 5 5
        gateway 192.168.2. 1
up route add -net 192.168.1.0/24 gw 192.168.2. 1
up route add default gw 192.168.2. 1
```

Of course, you have to adjust it properly to suit your needs.

3.3 Exporting the guest's root file system

With the root file system populated on the NFS server and NFS services running, the next step is to export the file system in order for the UML guests to be able to mount it. This is achieved by adding an entry to the "/etc/exports" file on the server:

```
/export/uml-client
192.168.2.2(rw,fsid=0,async,wdelay,no_root_squash,no_subtree_check)
```

In the above example 192.168.2.2 is the IP address of the UML guest which will use the exported root file system. The option "rw" indicates that the UML guest can make changes to the file system, while the "fsid=0" signals that the current export is the root. Also, "async" enables asynchronous writes and "wdelay" tells to the NFS server to delay writing to the disk if it suspects another write request is imminent. The "no_root_squash" option disables root squashing on the server and finally the "no_subtree_check" disables the sub tree check [3]. Once the "/etc/exports" file has been updated the "exportfs" command can be run to update the table of exported NFS file systems:

```
root@host$ exportfs -a
```

Alternatively you can restart the NFS service.

4 Guest kernel configuration

The UML guest's kernel needs the following settings as minimum in order for the guest to be able to mount its root file system via NFS:

```
Networking options ->
  IP: kernel level autoconfiguration - enabled
  IP: DHCP support - enabled
  IP: BOOTP support - enabled
  IP: RARP support - enabled

File Systems ->
  Network File Systems ->
    NFS client support - enabled
    NFS client support for NFS version 3 - enabled
    NFS client support for the NFSv3 ACL protocol expansion - enabed
    NFS client support for NFS version 4 - enabled
    Root filesystem on NFS - enabled
    Use the new idmapper upcall routine - enabled
```

Be careful to compile the above options as build in and not as modules. Modules only work after the kernel is booted, and these things are needed during boot.

5 Running the UML guest

In order to successfully boot the new UML guest we have to pass it some options to inform it how to mount its root file system. There are three ways to pass options to the kernel and thus control its behavior:

- When building the kernel.
- When starting the kernel.
- At runtime, by writing to files in the "/proc" and "/sys" directories.

Here we will use the second approach. We will tell to the UML guest's kernel what root file system device to use, and where to find the server and the name of the directory on the server to mount as root. This can be established by a couple of kernel command line parameters [4]:

- "root=/dev/nfs"
  The first option that we have to pass to the UML guests kernel is its root file

system. Here comes the “/dev/nfs” dummy device file that we created earlier. This is necessary to enable the pseudo-NFS-device. Note that its not a real device but just a synonym to tell the kernel to use NFS instead of a real device.

- “rw”
  With this option we tell to the kernel to mount its root file system as read-write.

- “nfsroot=[<server-ip >:] <root-dir >[, <nfs -options >]”
  The “nfsroot” option tells to the kernel where to find its root file system. The “server-ip” specifies the IP address of the NFS server, the “root-dir” specifies the name of the directory on the server to mount as root and the “nfs-options” specifies some standard NFS options. All options are separated by commas. The default options are the following:

  – port = as given by the rpcbind/portmap daemon.
  – rsize = 1024.
  – wsize = 1024.
  – timeo = 7.
  – retrans = 3.
  – acregmin = 3.
  – acregmax = 60.
  – acdirmin = 30.
  – acdirmax = 60.
  – flags = hard, nointr, noposix, cto, ac.

  An example is given in the following line:

  nfsroot=192.168.2.1:/export/uml-client

  Here, 192.168.2.1 is the IP address of the NFS server and “/export/uml-client” is the directory on the server to mount as root.

- “ip=<client-ip>:<server-ip>:<gwip>:<netmask>:<hostname>:<device>:<autoconf>”
  This parameter provides all the necessary information for the client to set up its Ethernet interface and to contact the NFS server, respectively:

  – “client-ip” is the IP address to be used by the UML guest.
  – “server-ip” is the IP address of the NFS server.

  - “gw-ip” is the IP address of the gateway if the server lies on a different subnet.
  - “netmask” specifies the netmask for the guest’s network interface.
  - “hostname” specifies the hostname for the guest.
  - “device” specifies the name of the guest’s network device to be used.
  - “autoconf” specifies the method to use for autoconfiguration. If this is “dhcp”, “bootp”, or “rarp’, the specified protocol is used.

  An example is given in the following line:

  ip=192.168.2.2:192.168.2.1::255.255.255.0:uml-client:eth0:off

  Here 192.168.2.2 is the IP address of the UML guest, 192.168.2.1 is the IP address of the NFS server, 255.255.255.0 is the netmask, “uml-client” is the guest’s hostname and finally “eth0” is the guest’s network interface to be used. To automatically assign an IP address to the guest using DHCP service you can simply use the option “ip=dhcp”.

- “rootfstype=nfs”
  With this option we specify the type of the root file system.

Now, we can start the UML guest with the following command:

```
user@host$ ./linux rootfstype=nfs4 root=/dev/nfs rw
nfsroot=192.168.2.1:/export/uml-client
ip=192.168.2.2:192.168.2.1::255.255.255.0:uml-client:eth0:off mem=128M
eth0=tuntap,tap1
```

References

[1] Greg Kroah Hartman. *Linux Kernel in a Nutshell.* O’ Reilly, 2006.

[2] User Mofe Linux Installation Guide:
http://www.cs.uoi.gr/~gkappes/tutorials/uml_installation.pdf.

[3] NFSv4 Installation Guide:
http://www.cs.uoi.gr/~gkappes/tutorials/nfs_installation.pdf.

[4] Root over nfs clients & server Howto:
http://tldp.org/HOWTO/Diskless-root-NFS-HOWTO.html.