By Falko Timme

Published: 2009-03-12 20:02

Using ATA Over Ethernet (AoE) On Debian Lenny (Initiator And Target)

Version 1.0
Author: Falko Timme <ft [at] falkotimme [dot] com>
Last edited 02/25/2009

This guide explains how you can set up an AoE target and an AoE initiator (client), both running Debian Lenny. **AoE** stands for "ATA over Ethernet" and is a storage area network (SAN) protocol which allows AoE initiators to use storage devices on the (remote) AoE target using normal ethernet cabling. "Remote" in this case means "inside the same LAN" because AoE is not routable outside a LAN (this is a major difference compared to iSCSI). To the AoE initiator, the remote storage looks like a normal, locally-attached hard drive.

I do not issue any guarantee that this will work for you!

1 Preliminary Note

I'm using two Debian Lenny servers here:

- *server1.example.com* (Initiator): IP address *192.168.0.100*
- *server2.example.com* (Target): IP address *192.168.0.101*

2 Loading The aoe Kernel Module On Both Systems

server1/server2:

Before we start, we must make sure that the the kernel supports AoE:

```
grep ATA_OVER /boot/config-`uname -r`
```

This should display something like this:

```
server2:~# grep ATA_OVER /boot/config-`uname -r`
  CONFIG_ATA_OVER_ETH=m
  server2:~#
```

This means that AoE was built as a kernel module. Let's check if the module is already loaded:

```
lsmod | grep aoe
```

If you get nothing back, this means it's not loaded. In this case we can load it as follows:

```
modprobe aoe
```

Let's check again if the module is loaded:

```
lsmod | grep aoe
```

```
server2:~# lsmod | grep aoe
aoe                    22112  0
server2:~#
```

To have the module loaded automatically when the system boots, we add the `aoe` module to `/etc/modules`:

```
vi /etc/modules
```

```
# /etc/modules: kernel modules to load at boot time.
#
```

```
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.

aoe
loop
```

3 Setting Up The Target (server2)

server2:

First we set up the target (*server2*):

```
aptitude install vblade
```

We can use unused logical volumes, image files, hard drives (e.g. */dev/sdb*), hard drive partitions (e.g. */dev/sdb1*) or RAID devices (e.g. */dev/md0*) for the storage. In this example I will create a logical volume of 20GB named *storage1* in the volume group *vg0*:

```
lvcreate -L20G -n storage1 vg0
```

(If you want to use an image file, you can create it as follows:

```
mkdir /storage

dd if=/dev/zero of=/storage/storage1.img bs=1024k count=20000
```

This creates the image file */storage/storage1.img* with a size of 20GB.

)

Now we export our storage device as follows:

```
vbladed 0 1 eth0 /dev/vg0/storage1
```

The first number (`0`) is the shelf number (major), the second (`1`) the slot number (minor), change these numbers to your liking. Each AoE device is identified by a couple major/minor which must be unique (if you are exporting multiple devices), with major between 0-65535 and minor between 0-255. The `eth0` part tells `vbladed` which ethernet device to use (if you ethernet device is `eth1`, then use `eth1` - you can find out about your ethernet devices by running

```
ifconfig
```

).

To start the export automatically whenever you boot the target, open `/etc/rc.local`...

```
vi /etc/rc.local
```

... and add the following line to it (before the `exit 0` line):

```
[...]
vbladed 0 1 eth0 /dev/vg0/storage1
[...]
```

4 Setting Up The Initiator (server1)

server1:

On `server1`, we install the initiator:

```
aptitude install aoetools
```

Now we check what AoE storage devices are available:

```
aoe-discover
```

The command

```
aoe-stat
```

should now show the storage devices:

```
server1:~# aoe-stat
      e0.1        21.474GB   eth0 up
server1:~#
```

At this point we have a new block device available on the client box named `/dev/etherd/e0.1`. If we have a look at the `/dev` tree a new node appears:

```
ls -la /dev/etherd/
```

```
server1:~# ls -la /dev/etherd/
total 0
drwxr-xr-x  2 root root     160 2009-02-25 14:47 .
drwxr-xr-x 12 root root    3180 2009-02-25 14:47 ..
c-w--w----  1 root disk 152,  3 2009-02-25 14:06 discover
brw-rw----  1 root disk 152, 16 2009-02-25 14:47 e0.1
cr--r-----  1 root disk 152,  2 2009-02-25 14:06 err
c-w--w----  1 root disk 152,  6 2009-02-25 14:06 flush
c-w--w----  1 root disk 152,  4 2009-02-25 14:06 interfaces
```

```
c-w--w----  1 root disk 152,  5 2009-02-25 14:06 revalidate
server1:~#
```

In the output of

```
fdisk -l
```

you should now also find the new hard drive:

```
server1:~# fdisk -l

Disk /dev/sda: 32.2 GB, 32212254720 bytes
255 heads, 63 sectors/track, 3916 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x00031334

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1        3749    30113811   83  Linux
/dev/sda2            3750        3916     1341427+   5  Extended
/dev/sda5            3750        3916     1341396   82  Linux swap / Solaris

Disk /dev/etherd/e0.1: 21.4 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x00000000

Disk /dev/etherd/e0.1 doesn't contain a valid partition table
server1:~#
```

To use that device, we must format it:

```
fdisk /dev/etherd/e0.1
```

```
server1:~# fdisk /dev/etherd/e0.1
  Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
  Building a new DOS disklabel with disk identifier 0xa00b110d.
  Changes will remain in memory only, until you decide to write them.
  After that, of course, the previous content won't be recoverable.


  The number of cylinders for this disk is set to 2610.
  There is nothing wrong with that, but this is larger than 1024,
  and could in certain setups cause problems with:
  1) software that runs at boot time (e.g., old versions of LILO)
  2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)
Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help): <-- n
Command action
   e   extended
   p   primary partition (1-4)
<-- p
Partition number (1-4): <-- 1
First cylinder (1-2610, default 1): <-- ENTER
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-2610, default 2610): <-- ENTER
Using default value 2610

Command (m for help): <-- t
Selected partition 1
Hex code (type L to list codes): <-- 83

Command (m for help): <-- w
The partition table has been altered!
```

```
Calling ioctl() to re-read partition table.
Syncing disks.
server1:~#
```

Afterwards, the output of

```
fdisk -l
```

should look as follows:

```
server1:~# fdisk -l

Disk /dev/sda: 32.2 GB, 32212254720 bytes
255 heads, 63 sectors/track, 3916 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x00031334

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1        3749    30113811   83  Linux
/dev/sda2            3750        3916     1341427+   5  Extended
/dev/sda5            3750        3916     1341396   82  Linux swap / Solaris

Disk /dev/etherd/e0.1: 21.4 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0xa00b110d

            Device Boot      Start         End      Blocks   Id  System
/dev/etherd/e0.1p1               1        2610    20964793+  83  Linux
server1:~#
```

Now we create a filesystem on `/dev/etherd/e0.1p1`...

```
mkfs.ext3 /dev/etherd/e0.1p1
```

... and mount it for test purposes:

```
mount /dev/etherd/e0.1p1 /mnt
```

You should now see the new device in the outputs of...

```
mount
```

```
server1:~# mount
  /dev/sda1 on / type ext3 (rw,errors=remount-ro)
  tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
  proc on /proc type proc (rw,noexec,nosuid,nodev)
  sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
  udev on /dev type tmpfs (rw,mode=0755)
  tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
  devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
  /dev/etherd/e0.1p1 on /mnt type ext3 (rw)
server1:~#
```

... and

```
df -h
```

```
server1:~# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1              29G  683M   27G   3% /
tmpfs                 253M     0  253M   0% /lib/init/rw
```

```
udev                     10M   88K   10M   1% /dev
tmpfs                   253M     0  253M   0% /dev/shm
/dev/etherd/e0.1p1       20G  173M   19G   1% /mnt
server1:~#
```

You can unmount it like this:

```
umount /mnt
```

To have the device mounted automatically at boot time, e.g. in the directory */storage*, we create that directory...

```
mkdir /storage
```

... and add the following line to */etc/fstab*:

```
vi /etc/fstab
```

```
[...]
/dev/etherd/e0.1p1       /storage        ext3    defaults,auto,_netdev 0 0
```

This alone isn't enough to have the device mounted at boot time because the AoE stuff gets loaded after */etc/fstab* is read. Therefore we open */etc/rc.local*...

```
vi /etc/rc.local
```

... and add the following lines to it (before the *exit 0* line):

```
[...]
aoe-discover
sleep 5
mount -a
[...]
```

For test purposes, you can now reboot the system:

```
reboot
```

After the reboot, the device should be mounted:

```
mount
```

```
server1:~# mount
  /dev/sda1 on / type ext3 (rw,errors=remount-ro)
  tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
  proc on /proc type proc (rw,noexec,nosuid,nodev)
  sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
  udev on /dev type tmpfs (rw,mode=0755)
  tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
  devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
  /dev/etherd/e0.1p1 on /storage type ext3 (rw,_netdev)
server1:~#
```

```
df -h
```

```
server1:~# df -h
Filesystem            Size  Used Avail Use% Mounted on
```

```
/dev/sda1              29G  684M   27G   3% /
tmpfs                 253M     0  253M   0% /lib/init/rw
udev                   10M   88K   10M   1% /dev
tmpfs                 253M     0  253M   0% /dev/shm
/dev/etherd/e0.1p1     20G  173M   19G   1% /storage
server1:~#
```

5 Links

- AoE Protocol Definition: **http://www.coraid.com/RESOURCES/AoE-Protocol-Definition**
- Debian: **http://www.debian.org**