MirrorDisk/UX

Feature *by Galen Scalone*

MirrorDisk/UX is a product developed by HP that enables us to store multiple copies of data on separate disks so that, in the event of hardware failure, the data is still accessible, and the system will continue to run. MirrorDisk/UX (HP part number B2491BA) does not come with the standard HP-UX operating system. Please note that you will need double your amount of disk space if you decide to mirror your data. Besides being a mirroring tool, it also offers us a tool to move data, load balance disks or SCSI channels, and do backups with little or no downtime. This article is written for administrators who have an understanding of Logical Volume Manager, as MirrorDisk/UX will add functionality to two LVM commands, *lvextend*, and *lvreduce*, and add the commands *lvsplit*, *lvmerge*, *lvsync*, and *vgsync*. We wil cover the following topics:

- how to mirror a non-root volume group and a root volume group
- how to recover a volume group from hardware failure
- using mirroring to move data within a volume group
- load balancing your disks and/or SCSI channels using mirrors
- splitting mirrors to back up your data

This will all be done from the command line.

Background

Let's recall the *logical* extent to *physical* extent relationship. When LVM is working with physical disk drives, it divides those drives into sections called *physical extents*. By default, a physical extent is 4 megabytes, but the size is configurable at volume group creation. LVM allocates an address for each extent starting at the beginning of the disk, excluding a reserved header area. The LVM software builds internal tables that hold these maps of physical extents.

When we create a logical volume, LVM creates *logical extents* that map to physical extents on disk. In a non-mirrored LVM environment, every *logical extent* will be mapped to one *physical extent.* The LVM software keeps an internal table of logical to physical extent mappings.

When we use MirrorDisk/UX, we map one logical extent to *two* or *three* physical extents, depending on the number of mirrors we would like to have.

Mirroring a Non-Root Volume Group

In the following examples we will assume that a volume group named */dev/test* contains one disk, device file */dev/dsk/c0t0d0*, with a hardware path of *8/0.0*. There is one other disk on the system, on another SCSI controller or *channel*, and its

device file is */dev/dsk/c1t0d0*, hardware path *16/0.0*; it is not being used. The disk types we will be using are the 2-GB hot swappable disks.

The volume group contains five logical volumes that are each 200 MB in size These logical volumes contain filesystems that house an Oracle database (see *Listing 1*). We have decided that we want this data protected from hardware failure, and want to use MirrorDisk/UX to accomplish this. The first task is to find an unused disk that is at least the same size as the disk currently in the volume group. This disk would preferably be on a different SCSI *channel*, as this will provide for redundancy not only in case of disk failure, but for channel, and/or, cable failure. We will add this disk to the */dev/test* volume group, then create mirror copies of the existing logical volumes on the new disk. Once we have chosen a disk we do the following:

`pvcreate /dev/rdsk/c1t0d0` This makes the disk an LVM disk.

`vgextend /dev/test /dev/dsk/c1t0d0` This puts the new disk into the */dev/test* volume group

`lvextend m 1 /dev/test/data1 /dev/dsk/c1t00` This makes one mirror copy of */dev/test/data1* on the disk */dev/dsk/c1t0d0*, and should be repeated for all logical volumes in the volume group.

Now that the mirrors are in place, use `lvdisplay v /dev/test/data1` to look at the logical and physical extents of the logical volume `/dev/test/data1` (see *Listing 2*). Notice one logical extent being mapped to two physical extents, and that all extents are current, or fully synced. At this point all logical volumes in */dev/test* are mirrored with disks on different channels and protected from disk failure, SCSI card failure, and cable failure.

Backups Using Mirrors

MirrorDisk/UX offers system administrators the ability to split a logical volume into two logical volumes and then mount the mirror copy elsewhere on the system for backup. The original volume stays online, while the split volume will go offline, its extents not being updated. Once you have ***split the mirror***, you should check the mirror's filesystem for errors, make a mount point for it, and mount the filesystem to the new mount point.

We will use the */dev/test* volume group from the previous section, since it is now a fully mirrored volume group.

For example, the Oracle database needs to be backed up every night, but cannot be down for backup for an extended period of time. So we need to shut the database down and have it started again in the least amount of time possible. The way to accomplish this is to shut the database down, split the mirror copies of the logical volumes, then restart the database. Once the database is restarted, we will mount, and back up the mirror. This requires only a few minutes of downtime.

After the database is down, and there is no activity on the disks in the volume group, we do the following:

`lvsplit /dev/test/data1` This command splits the */dev/test/data1* logical volume, and creates a logical volume called */dev/test/data1**b*** using the extents from the mirror copy of the volume (see *Listing 3*).We will run this command for all logical volumes in the volume group. After all volumes are split, we restart the Oracle database.

`fsck F vxfs /dev/test/rdata1b` This command checks the filesystem on the split volume for errors, and will also be run for all newly created logical volumes.

`mkdir -p /oracle.backup/data1` This creates a directory where we will mount the mirror volumes. We will make a directory for each logical volume in the volume group.

`mount /dev/test/data1b /oracle.backup/data1` This mounts the newly created */dev/test/data1b* filesystem to a mount point called */oracle.backup/data1*. We will mount all mirror volumes to their respective mount points (see *Listing 4*). Then we start a backup.

When the backup is complete, we unmount and merge the volumes:

`umount /oracle.backup/data1` This unmounts the split volume.

`lvmerge /dev/test/data1b /dev/test/data1` This command will merge the mirror copy of the logical volume with the original, removing the separate mirror volume and doubling the number of extents in the original volume.

Mirroring the Root Volume Group

When mirroring the **root** logical volumes, what you are trying to achieve is a type of high availability. You want the **system** to continue to run in the event that a root disk, SCSI controller, or SCSI cable fails. In order to accomplish this, the system will need two copies of the logical volumes in */dev/vg00*, on two separate disks. These disks will be on separate SCSI channels. So we'll add a bootable disk to the volume group, put the boot utilities on that disk, then mirror the logical volumes to that disk:

`pvcreate B /dev/rdsk/c1t0d0` Make the disk a bootable, LVM disk.

`vgextend /dev/vg00 /dev/dsk/c1t0d0` Add the disk to the volume group.

`mkboot /dev/rdsk/c1t0d0` Place LIF area on the disk.

`mkboot a "hpux (16/0.0;0)/stand/vmunix" /dev/rdsk/c1t0d0` Place Auto file in LIF.

`lvextend m 1 /dev/vg00/lvol1 /dev/dsk/c1t0d0` Create a mirror copy of */dev/vg00/lvol1*, and put it on disk */dev/dsk/c1t0d0*. You will need to repeat this command for each logical volume in vg00.

The following is a loop to mirror all of the logical volumes in vg00 consecutively, with minimal keystrokes:


```
for a in 1 2 3 4 5 6 7 8

do

lvextend m 1 /dev/vg00/lvol${a} /dev/dsk/c1t0d0

done
```

At this point, all of the logical volumes in vg00 are mirrored to separate disks. This means that each logical extent maps to two physical extents, one set of extents on one disk, and the other set of extents on the other disk. LVM will write to both sets of extents at the same time (default). Should you lose one disk, the system still has a full copy of the logical volumes on the other disk to read from, and to write to; therefore the machine continues to run. The **system** is now protected against a disk failure, a SCSI card failure, or SCSI cable failure.

Recovery of a Failed Disk in vg00

As system admins, we should expect disks to fail every now and then. The idea behind using MirrorDisk/UX is that if you do have a disk failure or SCSI card failure, the system still has one copy of data that it needs to operate and will continue to run. It still has one set of extents it can read from and write to, and it does not matter which set it has--the original or the mirror copy.

There are a few ways to find out whether a disk has failed, although `vgdisplay v` is probably the quickest. When `vgdisplay v /dev/vg00` is run, it will show the logical volumes in the vg00 volume group and the disks in the vg00 volume group. When one of the disks in the volume group has failed, *vgdisplay* will show the disk as unavailable, and will also show some or all of the logical volumes as being in a stale state (see *Listing 5*).

Once you determine which disk has failed, you will need to replace it. Since these examples are using hot swappable HP disks, replacement should be pretty simple--just remove the bad disk and insert a new disk. We will use the same disks as the previous example, *c0t0d0* and *c1t0d0*, but we'll have them in the */dev/vg00* volume group, and say that *c0t0d0* has failed. Remove the disk from the cabinet, and replace with the same size disk. Next is the software portion. Once the new disk is in place:

`vgcfgrestore n /dev/vg00 /dev/rdsk/c0t0d0` This command restores LVM configuration to the reserved header area on the new disk from the file */etc/lvmconf/vg00.conf*, the volume group configuration backup file.

`vgchange a y /dev/vg00` This command tries to activate any physical volumes in the volume group that had previously been unavailable, or missing.

`vgsync /dev/vg00` This command will re-sync the physical extents of all logical volumes in the volume group.

Alternatively:

`lvsync /dev/vg00/lvol1` This command will re-sync one logical volume at a time; repeat for all volumes.

At this point your mirror copies are valid again. Notice that in the previous example, the original disk is the disk that failed, not the mirror. If for some reason you cannot immediately replace the disk, you will need to make sure the machine, at boot time, knows which disk to boot from, in this case the mirror. We would `setboot p 16/0.0` to set the primary boot path to point to the mirror disk.

Load Balancing

In some cases you may not want, or need, your data mirrored. If you ever have a disk or I/O bottleneck, you can use MirrorDisk to *move* data from one disk to another. There is no need for downtime, since you will create a mirror copy of a logical volume, then remove the original copy. For instance, we had an Oracle database that was in a volume group that contained disks on a disk array. We had only two SCSI channels from the HP to the array. With archive logging turned on, Oracle was writing its archive logs to a filesystem on the array. When the database started to get busy, we were getting I/O bottlenecks on the disk that had the archive log filesystem on it, which in turn slowed the whole channel down. We also had a Jamaica cabinet on the system that was on a separate SCSI channel and contained four 4-GB drives. We added one of these disks to the volume group, made a mirror copy of the archive log logical volume on the new disk, then removed the original copy from the array disks, actually moving the filesystem from the array to the disk in the Jamaica cabinet. Now when the system gets busy writing archive logs, it is writing to a separate disk, on a separate channel, not the array--therefore no more I/O bottleneck.

Moving Data from Disk to Disk Within a Volume Group

At one site, we had an HP K-Class server with a disk array attached. We had two SCSI channels from the HP to the disk array, one of those channels being alternate links (see *note 1*). We had one Oracle instance on the disk array. We needed to move the data from the one disk array to another newer updated array that the company had just purchased. The new array was to be configured the same way as the old array, two SCSI channels, one being an alternate. The following is probably *not* supported by HP, and if you decide to try this, be sure to have a backup done before starting.

What we did was *vgreduce* the alternate links from the volume group, leaving only one path to the disks in the array. After making sure that all disks on the one channel were not being used, we disconnected that SCSI cable from the HP. The system was up and running when we did this. After disconnecting the cable, we ran `rmsf H <hardware path to disk>` *(*see *note 2)* to remove all of the device files that existed on that channel. We connected one of the cables from the new array to the card we just disconnected from, then ran *insf e (*see *note 3)* to install the new device files for the new disks. With the new device files installed, we *pvcreate*d the new disks, and extended the volume group to contain all of the new disks.

The next step was to mirror every logical volume in the volume group to the new set of disks, and then remove the original copy from the disks on the old array. Again this can be done with the system running. After making sure again that none of the disks on the old array were being used, we *vgreduce*d the old disks out of the volume group, and disconnected the cables for those disks from the HP. The next step was to use *rmsf* again to remove the old device files, and then connect the

alternate link cables from the new array to the HP.

Again run `insf e`, and *pvcreate* the disks. Then *vgextend* the volume group to contain the new alternate link disks.

When you add alternate links to a volume group, the order in which you add them is the way they will be used. Example: You have a disk, *c0t0d0*, and its alternate link is *c2t0d0*. When you add *c0t0d0* to a volume group, then add *c2t0d0* to the volume group, *c0t0d0* will be the primary path. If you have 10 disks, and 10 alternate links, you want to alternate primary paths to use both SCSI controllers. This will also help with I/O bottlenecks, as you are using both channels to move data, instead of only one.

`vgreduce /dev/test /dev/dsk/c1t0d0 /dev/dsk/c1t1d0 /dev/dsk/c1t2d0 /dev` This command reduces all of the paths to the disks on one channel, ***c1***, from the test volume group, regardless of whether it is a primary or alternate path, leaving one path down one channel to each disk.

`vgdisplay v |grep /dev/dsk/c1` Shows only disks that are on the ***c1*** channel in any volume group. After ensuring that there are indeed no disks from this channel configured into any volume group, we disconnect the ***c1*** cable from the HP.

`rmsf H 16.0 -H 16.1 -H 16.2 -H 16.3 -H 16.4` This command removes all special files for the disks located at the hardware paths specified, or all disks on the ***c1*** channel (see *ioscan*).

At this point, we connect the cable from the new array to the SCSI card we disconnected the old cable from, on the HP.

`insf ev` This command installs special files for new or existing devices.

`pvcreate /dev/rdsk/c1t0d0 /dev/dsk/c1t1d0...` This command makes all of the new disks on the new array LVM disks.

`vgextend /dev/test /dev/dsk/c1t0d0 /dev/dsk/c1t1d0...` This command takes all of the new disks and adds them to the */dev/test* volume group.

`lvextend m 1 /dev/test/data1 /dev/dsk/c1t0d0` This command creates a mirror copy of the */dev/test/data1* on the newly added disk, */dev/dsk/c1t0d0* (on the new array). This is repeated for all logical volumes in the volume group, and new disks in the volume group.

Now we have a copy of the data on both disk arrays.

`lvreduce m 0 /dev/test/data1 /dev/dsk/c5t0d0` This command will remove the data from the old array, leaving only one copy on the new array. Again repeat for all logical volumes, and all disks on ***c5.***

At this point all of the data is on the new array only.

`vgreduce /dev/test /dev/dsk/c5t0d0 /dev/dsk/c5t1d0...` This removes

the disks on the ***c5*** channel from the test volume group. When this is complete, we disconnect the cable for the **c5** channel from the HP.

`rmsf H 8.0 -H 8.1 -H 8.2 -H 8.3 -H 8.4` This command removes all special files for the disks located at the hardware paths specified, or all disks on the ***c5*** channel. At this point we connect the other cable from the new array to the SCSI card we disconnected the ***c5*** channel from.

`insf ev` This again is used to create the new device files for the newly added disks.

`vgextend /dev/test /dev/dsk/c5t0d0 /dev/dsk/c5t1d0...` this will add the alternate paths to all of the disks in the new array to the */dev/test* volume group.

At this point we have successfully moved the test volume group from one disk array to another disk array, using mirrors.

We then use `pvchange -s /dev/dsk/c5t1d0 /dev/dsk/c5t3d0` to alternate the primary and alternate links for these disks to load balance the SCSI channels.

You will want to run sar reports for disk I/O, and make any primary or alternate link changes, depending on the results of the disk reports.

Moving a Logical Volume from One Volume Group to Another

In some instances we may need to move a logical volume from one volume group to another volume group. This is a little tricky, because you create a mirror copy of a logical volume on a clean disk, remove the mirror--which in fact only removes the headers on the disk and leaves the data intact--then move the disk to another volume group. For example we had a system that had an Informix raw volume in the */dev/vg00* volume group, and needed to move the volume to the Informix volume group. We added a new disk to the */dev/vg00* volume group, created one mirror copy of the logical volume that was 200 MB, on the new disk, *which started from the first physical extent on the disk*, and immediately removed the mirror from the new disk. The removal of the mirror copy only removes the headers on the disk, and not the data contained on the disk. We then removed the disk from the volume group, and added it to the other volume group. We then created the logical volume, exactly the way the mirror was created, *first on the disk*, and the data was still accessible--after changes to the Informix config. to use the new logical volume, in the new volume group. It looked like this:

`pvcreate /dev/rdsk/c3t0d0` Makes the disk an LVM disk.

`vgextend /dev/vg00 /dev/dsk/c3t0d0` Adds the disk to the volume group.

`lvextend m 1 /dev/vg00/rawvol /dev/dsk/c3t0d0` Makes one mirror copy of */dev/vg00/rawvol* on */dev/dsk/c3t0d0* from the first physical extent, and adds data consecutively to the next physical extent.

`lvreduce m 0 /dev/vg00/rawvol /dev/dsk/c3t0d0` Removes the headers, as in all cases, from the disk, leaving the data intact.

`vgextend /dev/informix /dev/dsk/c3t0d0` Adds the disk to the */dev/informix* volume group.

`lvcreate n rawvol /dev/informix` Creates an empty volume called *rawvol* in the */dev/informix* volume group.

`lvextend L 200 /dev/informix/rawvol /dev/dsk/c3t0d0` Extends the volume to the */dev/dsk/c3t0d0* disk. The data is already in the volume--simply point the Informix link to the new volume, in the new volume group.

In Closing

MirrorDisk/UX is a powerful tool that is used to protect our systems from disk failure, channel failure, and cable failure. With redundant hardware, and MirrorDisk/UX, we can make our systems highly available, we can make our systems use multiple channels for better I/O performance, and we can back up our systems and applications with minimal downtime. Some of the more uncommon ways of using mirrors, such as moving volumes around the system to the same or even different volume groups, can easily be accomplished, once you have the knowledge and the experience in MirrorDisk/UX use.

Note 1: Alternate links, or pvlinks, are multiple paths to the same physical disk. Used for redundancy, in case of channel failure, alternate links will automatically change paths to a disk in the event of a card failure, increasing availability.

Note 2: *rmsf* is the command to remove the special files or device files of a given device. In this case, we need to remove the device files because the new disk array attached to that same SCSI card will have the same device file names, although the disks are on a different array and may be of a different size.

Note 3: `insf e` rebuilds all of the device files for the new disks on the system. This needs to be done if you have used *rmsf* to remove device files, or if you are adding a set of disks to a system while the system is online. A form of this command is run every time the system is restarted, in order for the system to see any newly added devices.

Galen Scalone works for Lucent in St. Petersburg, Florida as a Senior System Administrator. Prior to that he was with Symbol Technologies for about four years. He has been working with UNIX since 1993. He can be reached at *scalone@lucent.com* *or* *galen@bbnow.net.*