

MFT / Linux Setup Documentation

August 31, 2008

1. Loading the MFT software.

The MFT software actually uses a driver called “Fast Block Device” or ‘fbd’. The MFT software is designed to run from /usr/local/fbd. You can load this from the MFT distribution ‘tar file’ as:

```
cd /usr/local
tar xvzf /tmp/fbd-083108.tar.gz (your archive name and location may vary)
```

2. Setup your Flash SSDs.

MFT is very flexible. You can use just about any block device as the MFT “backing device”. Depending on your situation, you may need to pay attention to layouts to achieve the best levels of performance and drive life.

2.A. Single Flash SSD

If you have a single Flash drive, you still have a number of options. You can setup MFT with a single Flash drive using:

- The whole drive without a partition table
- Part of the drive by using a partition table
- Part of the drive by using LVM

It is also possible to mirror a Flash SSDs (or Flash SSD partition or logical volume) with a regular hard drive to create a raid-1 “asymmetric” array.

2.A.1: Use the whole SSD without a partition table

This is the easiest case. You can simply format the Flash SSD with:

```
/usr/local/fbd/fbd-maint format /dev/sdc -wtblksz=2048 -quick
```

This example assumes a 2048K erase block, which is a common size for many Flash SSDs.

2.A.2: Use a partition on the SSD

This is still easy, but you will need to deal with partition alignment. MFT is designed to exactly overlay the erase blocks structure of the SSD. If you miss this alignment, your write performance and drive wearout will degrade about 30%.

The easiest way to do this is to always partition a Flash SSD so that every partition starts on an erase block boundary. With most drives, this means the first partition will start at block 4096 instead of the default of 63. If you partition this way, then sector0 offsets are not necessary. The ‘-u’ option on the fdisk program allows you to enter partitions offsets as sectors instead of tracks. If you don’t map partitions to erase block boundaries, then simple layouts still work, but you need to do some math.

First, you have to find out where the partition starts:

```
fdisk -ul /dev/sdc
```

Look at the starting sector number for the partition your are using for MFT. You then need to tell MFT how many sector to skip so that MFT data starts at an erase block boundary. Lets assume that fdisk shows that the partition starts at sector 123,457:

$$123,457 / 2048 = 60.2861 * 2048 = 124,928 \quad 124,928 - 123,457 = 1,471$$

So in this case we need to skip the first 1,471 sectors before MFT starts storing data. We then format the partition with:

```
/usr/local/fbd/fbd-maint format /dev/sdc3 -wtblksz=2048 -offset0=1471 --quick
```

2.A.3 Use a Logical Volume

If you use LVM then the goal is the same as for a partition. LVM usually resides inside of a partition, so first find the starting sector number for the partition. Then use vgdisplay to see how the volume group is setup:

```
vgdisplay vg0
```

If the “PE Size” is a multiple of the erase blocks size, then LVM is being kind and you can just use the offset0 number for the partition as calculated in 2.A.2. Most LVM setups have a 32 MB PE size which is a multiple of a single Flash SSDs erase block size. If LVM uses another size, then you need to see which physical extents are actually in use with:

```
lvdisplay -maps /dev/vg0/fbd
```

If the physical extents occupy a single contiguous range of blocks, then you can take the 1st extent number and multiple it by the PE size (expressed in sectors) and then add this number to the partitions starting sector number. The math in 2.A.2 will then tell you what to use in offset0:

```
/usr/local/fbd/fbd-maint format /dev/vg0/volname -wtblksz=2048 -offset0=1571 --quick
```

2.B. Single Flash SSD mirrored to a hard disk

You can setup a “poor man’s” RAID-1 array by mirroring a Flash SSD with a regular hard disk. This is done with Linux’s software RAID-1 driver using the ‘—write-mostly’ option. This option will cause all writes to the “device” to mirror to both the SSD and the hard disk but all reads will be serviced by the SSD. This gives you the performance of the SSD with RAID-1 data integrity without having to buy two SSDs. If the SSD crashes, your data is still intact on the HDD and you application will continue to run un-interrupted, albeit a lot slower.

```
mdadm --create /dev/md10 -a yes -level=1 -raid-devices=2 /dev/sdc --write-mostly /dev/sda5  
/usr/local/fbd/fbd-maint format /dev/md10 -wtblksz=2048 --quick
```

This same method works with partitions and logical volumes located on the Flash SSD, although you will need to supply an ‘—offset0=nnn’ to get the erase block to line up in the correct location.

2.C. RAID-0 Flash SSD arrays

All of these examples are using Linux “software raid”. You can use hardware raid as well with MFT. See the “comparison of raid controllers” article for information on hardware vs software raid. If you are going to run RAID-0 (striped), then you need to either stripe the entire hard drive as in:

```
mdadm --create /dev/md10 --a yes --level=0 --raid-devices=2 /dev/sdc /dev/sdd
```

Or else you need to partition the devices so that the partitions start on exact erase block long boundaries.

```
fdisk -ul /dev/sdc
```

/dev/sdc3 starts on 1,822,720 (which is an exact multiple of 4096)

2.D. RAID-1 Flash SSD arrays

You can build simple mirrored arrays using either the entire drive or with partitions. If you use partitions, then the partition layout of both drives should be identical. Setting up partitions at erase block boundaries is the easiest solutions:

```
fdisk -u /dev/sdc  
fdisk -u /dev/sdd
```

Create partitions on both drives that are on 4096 sector multiples. You can then setup raid and format the device:

```
mdadm --create /dev/md10 --a yes --level=1 --raid-devices=2 /dev/sdc3 /dev/sdd3  
/usr/local/fbd/fbd-maint format /dev/md10 --quick
```

2.E. RAID-5 Flash SSD arrays

With RAID-5, you must either use entire drives or setup the partitions on erase block boundaries. It is not possible to use an offset0 value with RAID-5.

```
fdisk -u /dev/sdc  
fdisk -u /dev/sdd  
fdisk -u /dev/sde  
fdisk -u /dev/sdf
```

Create partitions on all drives that are on 4096 sector multiples. You can then setup raid and format the device:

```
mdadm --create /dev/md10 --a yes --level=5 --raid-devices=4 /dev/sdc3 /dev/sdd3 /dev/sde3 /dev/sdf3  
/usr/local/fbd/fbd-maint format /dev/md10 --quick
```

2.F. RAID-6 Flash SSD arrays

RAID-6 is setup just like RAID-5.

2.G. RAID-10 Flash SSD arrays

RAID-10 arrays work just fine with MFT, with small to medium sized arrays, RAID-10 can actually underperform RAID-5. With large arrays, RAID-10 can perform better, especially for high bandwidth arrays, but at a significantly higher cost.

2.H. RAID-50/RAID-60 Flash SSD arrays

These work fine as well, but with Linux software raid drivers, flat RAID-5 and RAID-6 arrays tend to perform just as well.

2.I. Larger asymmetric arrays

Depending on the devices you have, it might make sense to create a hybrid Flash and hard disk array with more than a simple mirror. For example, you might want to:

- RAID-0 4 MLC drives
- RAID-0 2 Hard disks
- RAID-1 the pair together

When you raid hard disks to Flash SSDs, you want to use a hard disk configuration that has enough write throughput to keep up with the SSDs. In this example 4 MLC drives will write at 40 MB/sec each for a maximum bandwidth of 160 MB/sec. One hard drive is not enough to get to 160 MB/sec, but two high performance drives at 80 MB/sec each will get there. The commands for this are:

```
mdadm --create /dev/md10 --yes --level=0 --raid-devices=4 /dev/sdc /dev/sdd /dev/sde /dev/sdf
mdadm --create /dev/md11 --yes --level=0 --raid-devices=2 /dev/sda3 /dev/sdb3
mdadm --create /dev/md12 --yes --level=1 --raid-devices=2 /dev/md10 --write-mostly /dev/md11
```

While you can extend this technique, our suggestion is “Don’t get too cute”. If you have one or two Flash SSDs that are going into an existing server with HDDs, then consider an asymmetric RAID-1 array. If you have more drives, just run RAID-5 and keep it simple.

3. fbd-maint documentation

The /usr/local/fbd/fbd-maint program has the following command-line parameters:

```
...fbd-maint format device --wtblksz=nnnn --offset0=nnn --quiet/-q
```

Perform an MFT logical format on the device.

```
...fbd-maint status device
```

Display the MFT logical format on the device.

```
...fbd-maint authorize {SerialNumber Capacity "Enterprise" {demo} }
```

Generate an activation code. You can send this code to EasyCo to activate your software.

```
...fbd-maint activate { AuthCode }
```

Enter the code you get back from EasyCo.

If you do not include an AuthCode, then the current state of the MFT activation is displayed.

```
/etc/mft.lic
```

This is where MFT stored it's license information. Email this to EasyCo if you experience activation problems.

4. The dm-fbd.ko kernel module

In order to actually use MFT, you need to setup the correct version of the dm-fbd.ko loadable kernel module. This version of MFT ships a “partial source” version of MFT. This lets you compile and link “most” of MFT on your server. This can allow MFT to operate with kernels that EasyCo has not specifically targeted.

Compiling the dm-fbd.ko kernel module requires that you have:

- The gnu C compiler
- Kernel headers

With RedHat style installations, this usually involves installing the packages ‘gcc’ and ‘kernel-devel’. Make sure that the kernel-devel package exactly matches your running kernel.

Once you have the compiler and kernel loaded you can build the dm-fbd.ko module with:

```
cd /usr/local/fbd/obj/date/  
vi Makefile  
    set the VER variable to match your configuration  
make install
```

5. The fbd startup script

To facilitate the easy startup of MFT, there is a startup script in /usr/local/fbd/init.d/fbd. You should edit this script to customize it for your particular configuration.

6. Actually start MFT

Once everything is setup and activated, you can actually start MFT with:

```
/usr/local/fbd/init.d/fbd start
```

If this completes without error, you can display the device's size with:

```
fdisk -l /dev/mapper/fbd1
```

7. Create a file system on MFT

Once MFT is started, you can create a filesystem on it and mount it:

```
mkfs.ext3 /dev/mapper/fbd1  
mount /dev/mapper/fbd1 /ssd
```

You can use any type of Linux file system that you wish. While EasyCo does most of it's testing with ext3, we have run MFT in conjunction with ext2, ReiserFS, XFS, FAT32, NTFS, and others. You should also realize that some options that are desirable with standard hard disks are much less important when MFT is in use. For example, the ‘noatime’ option is commonly used to prevent extra random disk writes. While this is still true with MFT, random

disk writes are so inexpensive that the noatime option has little actual impact on an MFT volume.

7.1. Create LVM on MFT

You can also setup LVM on top of MFT with:

```
pvcreate /dev/mapper/fbd1
```

If you setup LVM on top of MFT, you will need to setup the LVM rescan in /usr/local/fbd/init.d/fbd.

8. Setup fbd to startup on boot

The /usr/local/fbd/init.d/fbd script is designed to go into the Unix Sys-V startup infrastructure.

```
cd /etc/rc.d/init.d  
ln -s /usr/local/fbd/init.d/fbd fbd  
chkconfig --add fbd
```

9. Benchmarks

EasyCo's benchmark program 'bm-flash' is included in the /usr/local/fbd directory. You can benchmark regular or MFT devices.

Warning: bm-flash works only with devices and will destroy the contents of the device.

You can use bm-flash as:

```
/usr/local/fbd/bm-flash /dev/mapper/fbd1
```

... or

```
/usr/local/fbd/bm-flash /dev/md10
```

Please consider sharing your benchmarks, along with your hardware configuration with EasyCo.

10. Support and help

For support and help, please contact EasyCo:

EasyCo LLC

<http://easyco.com>

<http://managedflash.com>

support@easyco.com

+1 610 237-2000 x2