Using Linux mdadm Multipathing with Sun StorEdge™ Systems

Sun Microsystems, Inc.
www.sun.com

Part No. 819-2452-10
April 2005, Revision A

Submit comments about this document at: `http://www.sun.com/hwdocs/feedback`

Please
Recycle

Adobe PostScript™

Contents

Executive Summary

Linux mdadm (Multiple Device Administration) is a tool for creating, managing and monitoring device arrays using the "md" driver in Linux, also known as Software RAID arrays.

Mdadm was designed for RAID device creation, but also includes limited support for the creation, use and manipulation of multipath storage devices.

The manipulation of multipath storage devices is the focus of this paper.

Overview

The Linux software RAID driver, md, includes the ability to manage multiple paths to the same device. Configuration and management is provided by the mdadm command.

The md device names take the form `/dev/mdN`.

The devices that make up paths to the md device can be whole drives or partitions of drives. SCSI drive names can range from `/dev/sda to /dev/sdzz`.

Partitions are identified by a number appended to the device name. The first partition on /dev/sda would be `/dev/sdb1`.

Device names are assigned during boot and are subject to change if LUN, partition or device configuration is modified between reboots of the host.

The following sample command creates an md device:

```
mdadm -C /dev/md0 -l multipath -n 2 /dev/sdb1 /dev/sdc1
```

This command creates a multipath device accessible as `/dev/md0` using partition 1 on each of the LUNs `/dev/sda` and `/dev/sdc`. Mdadm doesn't check to see if they are paths to the same LUN/partition so it's important to identify them correctly during initial setup.

Multiple Device Administration

Unlike its predecessor, raidtools, mdadm does not require a configuration file (`/etc/raidtab`) making it much simpler to use and maintain.

Mdadm is included in the Linux kernel, and is widely viewed as the replacement solution for raidtools. This also means that the major distributions of Linux, RedHat and Suse, ship with mdadm as part of the base Linux kernel.

Mdadm path manipulation is a fairly manual process. The initial failover occurs when Linux detects a path down event. At that point, failover to the alternate path is initiated, and I/O resumes down the alternate path.

Mdadm does not support automatic failbacks, but does support manual path manipulation. See the "Path Manipulation" section for more detail.

Mdadm supports a simple non-mesh SAN (2 paths presented to the host) and symmetric arrays only. Because of the nature of asymmetric arrays, such as the Sun SE39xx and 6x20's families, only one data path is marked as active the during the initial bus scan. This makes it impossible for mdadm to use the alternate path for failovers. However, devices such as the Sun StorEdge 3510, and SE69xx families can be used with mdadm. Load balancing is not used by mdadm, meaning only one path per controller will be used at a time to pass I/O.

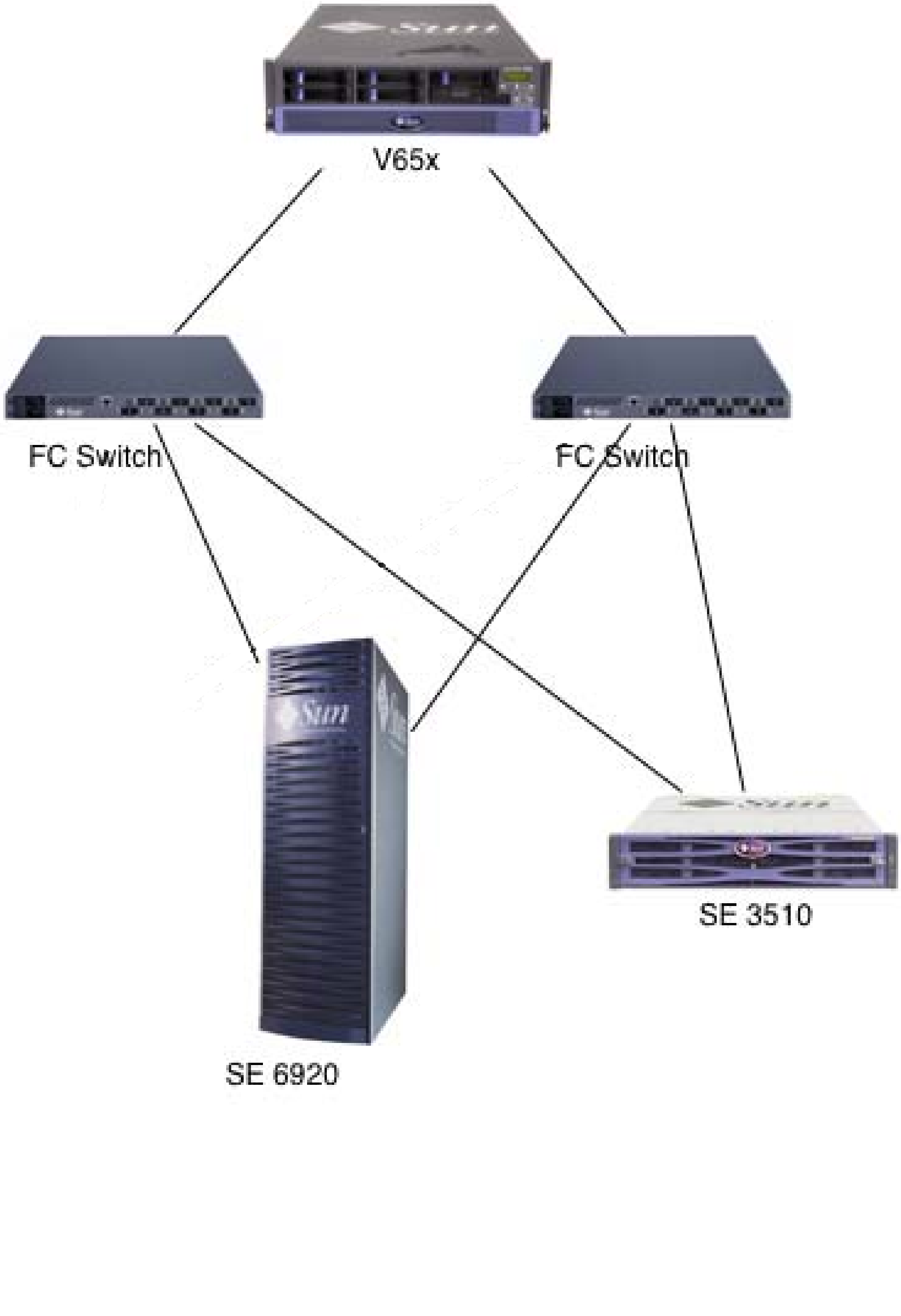
Many HBA drivers (Qlogic QLA2342 - v8.0x.xx for example) have MPIO failover enabled by default upon driver installation. This hides paths/states from mdadm, rendering mdadm unable to properly detect path states. Alternatively, some storage vendor's multipath filter drivers can be used in conjunction with an mdadm solution. One example is the Sun Linux mpp multipath driver, which has been tested in conjunction with mdadm and has been shown to work. See vendor specific documentation for more details and or possible limitations.

Sample Configuration

The following configuration is not the only possible configuration for use with mdadm. The host depicted was configured with a QLA2342DC (Sun Crystal 2a) HBA.

Testing was performed with both Red Hat 3.0, U3 and SuSe 9 SLES. Other HBA, Linux distributions, host and storage types may or may not work.

Consult vendor specific documentation for more details regarding supported components. All arrays depicted are symmetric; asymmetric arrays will not work with mdadm.


V65x
FC Switch
FC Switch
SE 6920
SE 3510

Configuring mdadm

Both RedHat 3.0, U3 and Suse SLES9 are mentioned in the following instructions and examples. Mdadm is included in the Linux kernel, so it may work with any distribution. Refer to vendor documentation for more details.

Note – The following steps and example are not the only way to accomplish successful configuration of mdadm. There are many ways to do the same things in Linux. This is one possible configuration of mdadm that has been proven to work. Feel free to substitute whatever methods you are comfortable with to achieve the same results.

To configure mdadm, perform the following steps:

1. **Configure and Zone the SAN as necessary.**

2. **Permission host LUNs on storage arrays.**

3. **Install the HBA Linux driver module.**

   Upon module installation, the host bus is scanned and any devices present in the SAN which are permissioned for the host are discovered and presented to the host for configuration.

   Note – Ensure that the HBA/Driver you are using does not have native multipathing enabled.

   Example: For Qlogic driver version 8.00.01 or higher, multipath support is enabled by default.

   Disabling it is accomplished by inserting the module using the following syntax:

   Load the driver with the `ql2xfailover` module parameter set to 0.

   ```
   # insmod qla2xxx.ko ql2xfailover=0
   # insmod qla2300.ko
   ```

   Other driver vendor/releases have different mechanisms for achieving this. Many use a configuration file. See vendor specific instructions for more details.

4. **After the driver module is loaded, discover the device nodes assigned to specific array LUNs/Partitions.**

Examples:

- For RedHat:

  i. Run hwbrowser from the command line.

  ii. Select "Hard Drives" in the left pane.

  All the storage devices connected to the host are displayed.

  Note the device names for the external devices you will be configuring for use. For this example, `/dev/sdb` and `/dev/sdc` are the device names for the SE6920 partition.

- For Suse SLES 9.1:

  i. Run yast2 from the command line.

  ii. Select "System"-> Partitioner.

  This displays the external devices you will be configuring for use.

  The fdisk -l may also be used, but presents less detail regarding the array type. The above examples using hwbrowser and yast2 list dev node names as well as vendor specific inquiry strings, making it much easier to identify what dev node belongs to what device LUN/Partition.

All of the following instruction are the same for both Redhat and Suse Linux distributions, and are specific to the test configuration used at the time of test. Your dev nodes may differ.

5. **Create your md devices.**

For this example, we will assume that the output received from running hwbrowser or yast2 above displayed the SE6920 device nodes as `/dev/sdb/` and `/dev/sdc`. These represent the LUN/partitions that have been permissioned for use by the host.

Run the following command to bind `/dev/sdb` and `/dev/sdc` to `/dev/md1`. Md1 will be the mdadm device node used to access the 6920 LUN:

```
% [root@earp root] mdadm -C /dev/md1 -l multipath -n2 /dev/sdb /dev/sdc
```

6. **Display /dev/md1 device information details for the mdadm device node create in step 5 for the SE6920 LUN:**

```
[root@earp root]# mdadm -D /dev/md1
/dev/md1:
Version : 00.90.00
Creation Time : Thu Oct 21 15:31:24 2004
Raid Level : multipath
Array Size : 10485696 (9.100 GiB 10.74 GB)
Raid Devices : 1
Total Devices : 2
Preferred Minor : 1
Persistence : Superblock is persistent
Update Time : Fri Oct 22 12:28:42 2004
State : clean, no-errors
Active Devices : 1
Working Devices : 1
Failed Devices : 1
Spare Devices : 0
Number Major Minor RaidDevice State
0 8 16 0 active sync /dev/sdb
1 8 48 1 spare /dev/sdc
```

At this point, the md device is ready for use.

Create a filesystem as you normally would for a Linux device and mount it. It is now ready for I/O.

▼ To Enable Devices After Rebooting

The md devices are not automatically enabled on reboot so it is necessary to start them manually, or by using a custom startup script.

When an md device is created using mdadm, information regarding the associated devices is written to the md super block on the disk. The `mdadm` command can be used to scan the superblocks, construct a configuration file, and enable the devices. An example is shown below.

1. **Store a pattern in the `/etc/mdadm.conf` directory that identifies the devices to be scanned.**

```
% echo "DEVICE /dev/sd[a-z]*" > /etc/mdadm.conf
```

2. **Scan the devices and append the results to the** `/etc/mdadm.conf mdadm` **directory.**

```
% mdadm -Es >> /etc/mdadm.conf
```

3. **Start the devices.**

```
% mdadm -As
```

See the mdadm man page for additional information.

Path Manipulation

This section describes some of the functionality of mdadm.

Refer to the mdadm manpages for detailed instructions on using mdadm.

Possible Device/Path states:

Active sync: The currently active path.

Spare: This denotes an available alternate path in a good (usable) state.

Faulty: A path that has experienced some type of failure and is not currently available for failover.

Common uses:

Mark an "active sync" or "spare" path as faulty:

```
% mdadm /dev/md(x) -f /dev/sd(x)
```

`Hot remove a path:`

```
% mdadm /dev/md(x) -r /dev/sd(x)
```

`Hot add a path:`

```
% mdadm /dev/md(x) -a /dev/sd(x)
```

Other Examples:

Repairing a "faulty" or bad path:

The above commands can be strung together to manipulate the path(s). If a path is down (faulty), and you wish to reactivate it, you would issue the following command:

```
% mdadm /dev/md(x) -f /dev/sd(x) -r /dev/sd(x) -a /dev/sd(x)
```

Where x is the currently faulty path. (/dev/sdd for example).This command first marks the path as faulty (-f), hot removes the path (-r), then hot adds the path (-a). While the path may already be marked as faulty, documentation on this procedure recommends using the -f switch anyways. This procedure would re-enable the path and make it accessible as an alternate path. Note: A faulty path MUST be removed before it can be added.

Manual Failover/Failback:

You can also issue the following command against a currently active path (active sync) to force a failover/failback. The paths /dev/sdb and /dev/sdc will be used to illustrate a path manipulation example.

Manual Failover example:

mdadm /dev/md1 -f /dev/sdb - Assuming /dev/sdb was in the "active sync" state, the preceding command would place /dev/sdb into a "faulty" state, causing a failover to the "spare" path (/dev/sdc).

Issuing an mdadm -D /dev/md1 would produce the following device state display after causing the failover:

```
% [root@earp root]# mdadm -D /dev/md1
/dev/md1:
Version : 00.90.00
Creation Time : Thu Oct 21 15:31:24 2004
Raid Level : multipath
Array Size : 10485696 (9.100 GiB 10.74 GB)
Raid Devices : 1
Total Devices : 2
Preferred Minor : 1
Persistence : Superblock is persistent
Update Time : Fri Oct 22 07:50:05 2004
State : dirty, no-errors
Active Devices : 1
Working Devices : 2
Failed Devices : 0
Spare Devices : 1
Number Major Minor RaidDevice State
0 8 16 0 active sync /dev/sdc
1 8 48 1 faulty /dev/sdb
UUID : a204fb79:b61f685c:35db5237:01f7e290
Events : 0.4
```

Manual Failback (Restore a path) example:

```
mdadm /dev/md1 -f /dev/sdb -r /dev/sdb -a /dev/sdb
```

This command would mark `/dev/sdb` faulty and remove it. Next it would hot add the `/dev/sdb` path back, and mark it as "spare".

Result:

```
Number Major Minor RaidDevice State
0 8 6 0 active sync /dev/sdc
1 8 48 1 spare /dev/sdb
```

Identifying Devices LUNs and Partitions

Attach messages are logged whenever a device name is assigned. They can be found in the output of dmesg or if dmesg has rolled, in a messages file in `/var/log`.

Controller, target and LUN information is included.

Sample command to list device names:

```
% dmesg | grep Attach
Attached scsi disk sda at scsi1, channel 0, id 1, lun 1
Attached scsi disk sdb at scsi1, channel 0, id 1, lun 2
Attached scsi disk sdc at scsi1, channel 0, id 1, lun 3
Attached scsi disk sdd at scsi2, channel 0, id 1, lun 1
Attached scsi disk sde at scsi2, channel 0, id 1, lun 2
Attached scsi disk sdf at scsi2, channel 0, id 1, lun 3
```

Note – LUNs sda and sdd are likely two paths to the same LUN, target 1 lun 1 on controllers scs1 and scsi2.

This can be confirmed by running `scsi_unique_id` against both LUNs and comparing the GUID returned in page 83.

```
% scsi_unique_id /dev/sda
model: SUN StorEdge 3510
page83 type3: 600c0ff00000000000043ee5fe6ae6201
page83 type3: 206000c0ff0043ee
page83 type0: 11000000007a21fc00c0ff0043ee0000819a46be
page80: 30303433454535464536414536323031
scsi_unique_id /dev/sdd
model: SUN StorEdge 3510
page83 type3: 600c0ff00000000000043ee5fe6ae6201
page83 type3: 206000c0ff0043ee
page83 type0: 11000000007a21fc00c0ff0043ee0000819a46be
page80: 30303433454535464536414536323031
```

The following output is from a different LUN for comparison. Note differences in the first and last page 83 line and the page 80 line:

```
% scsi_unique_id /dev/sdb
model: SUN StorEdge 3510
page83 type3: 600c0ff0000000000043ee5fe6ae6202
page83 type3: 206000c0ff0043ee
page83 type0: 1100000000 7a21fc00c0ff0043ee0000819a46be
page80: 30303433454535464536414536323032
```

There is an interface into the SCSIdriver for displaying specific information regarding LUNs.

Executing `cat /proc/scsi/scsi` will provide a list of attached devices.

Sample:

```
% cat /proc/scsi/scsi
Host: scsi1 Channel: 00 Id: 01 Lun: 01
Vendor: SUN Model: StorEdge 3510 Rev: 327R
Type: Direct-Access ANSI SCSI revision: 03
Host: scsi1 Channel: 00 Id: 01 Lun: 02
Vendor: SUN Model: StorEdge 3510 Rev: 327R
Type: Direct-Access ANSI SCSI revision: 03
```

Sun Microsystems, Inc.
www.sun.com

For U.S. sales office locations, call:	800-821-4643
In other countries, call:	
Corporate headquarters:	650-960-1300
Intercontinental sales:	650-688-9000