Enable multipath SAN booting with DS8000 and DMMP

Learn how to boot from storage area networks and improve your overall system performance

Lei Jiang

May 27, 2010
(First published September 23, 2009)

Booting servers from storage area networks with multiple paths can provide significant benefits for the complex, modern data center environment -- benefits such as improving availability and increasing your reach when it comes to easier management and maintenance and in providing reliability. To boot your Linux® system from a SAN device, you'll need to set up a SAN boot operating system to your storage with multipath driver support; that's where this article comes in. The author will walk you through setting up a SAN-boot Linux system on the IBM® System Storage DS8000 with the multipath function supported by the Device-Mapper MultiPath (DMMP) feature. [Editor's note: The article is updated per the author's comment below.]

A key requirement of today's complex data center environments is to be able to deliver mission-critical data quickly, at any time, without interruptions or delays. Booting servers from a storage area network (SAN) can provide significant benefits in this kind of situation.

A server traditionally boots its operating system from an internal hard disk, but this introduces a lot of effort when a physical server goes down, including replacing the server, reloading the operating system, reinstalling applications, and so on. Having the operating system boot from SAN storage simplifies this, because you only have to replace the server; your OS and applications are saved on the SAN.

Booting from SAN can also simplify data center administration, improve disaster tolerance, and reduce total cost of ownership (TCO) through diskless servers.

In this article, I illustrate how to set up Linux to boot from SAN storage, in this case the IBM System Storage DS8000 with the Device-Mapper MultiPath (DMMP) feature to manage multiple paths. I cover procedures for Red Hat Enterprise Linux (RHEL) 4 and 5, and SUSE Linux Enterprise Server (SLES) 9 and 10.

In this article, I focus on Fibre Channel-based SAN architecture, because it's a typical setup for data centers and the DS8000 supports it.
What is multipath SAN storage?

Back in the 1980s, disk or tape storage was connected directly to the server. Since then, however, the amount of data we depend on has grown, but directly attached storage has not kept up with our storage needs. We have become increasingly reliant on data sharing, large storage capacities, and the need to keep data safe.

One aspect of this changing need is for the data to be more available. We've addressed this by adding more paths between host and storage, hence the term multipath. With multiple paths for I/O, we improve fault tolerance—if a subset of the paths fail, the remaining paths will still function—and we improve performance, as I/O can go through multiple routes at the same time.

A simple multipath example and its benefits

A simple example is a server connecting directly to two SAN storage ports. If a single storage port or server port fails, the server still has another path for data transfer. This physical I/O channel configuration make multipath SAN possible at the physical layer.

In addition, we need a multipath software layer to manage the multiple storage paths. This layer should make the multipath solution transparent to users by showing them only one device and path. Without it, the system will recognize each path as a different device, and applications will not be able to use multiple paths at the same time.

Multipath software layers can use redundant paths to deliver performance enhancements such as dynamic load balancing, traffic shaping, automatic path management, and dynamic reconfiguration.

The structure of DMMP

DMMP is a multipath I/O implementation that provides failover function in an active/passive configuration: only half of the paths in this configuration are used for I/O at the same time. If any element of an I/O path (the cable, switch, or controller) fails, DMMP switches I/O to an alternate path.

DMMP can also be configured in active/active mode in which I/O is spread over all the paths in a round-robin fashion. In some configurations, DMMP can detect loading on the I/O paths and dynamically rebalance the load. For all these features, you'll need your storage system's implicit support.

Figure 1 shows a multipath configuration with two I/O paths from a server to storage (RAID array). There are two host bus adapters (HBAs) on the server, two SAN switches, and two RAID controllers on the RAID array.
**Figure 1. Multipath configuration with one RAID device**

In this configuration, there is one I/O path that goes through \textit{hba1, SAN1 (Switch X)} and \textit{controller 1} and a second I/O path that goes through \textit{hba2, SAN2 (Switch Y)} and \textit{controller 2}. With DMMP configured, any single point failure among these devices will cause DMMP to switch the I/O to an alternate path. So, your system is still usable even though there is a point failure; you can repair a failed point device without affecting system usage.

For an active/passive storage RAID array, your host can only reach one single LUN from this array by one of these two paths at any time. It normally uses a preferred path for I/O; DMMP will switch to a standby path if the preferred path meets with failure. DMMP will also spread I/O between these two paths to improve I/O performance for active/active storage.

If you like, you can add more paths between your server and your RAID array to increase reliability, such as adding one path between \textit{controller 1} and \textit{SAN 2} or one path between \textit{controller 2} and \textit{SAN 1}. You would then have four paths between your server and storage raid array as in Figure 2:
Booting from a multipath-enabled SAN

As mentioned above, the operating system is traditionally installed on the local hard disk of server, but you would need to reload the OS and redeploy the applications if this server fails, increasing maintenance time. Figure 3 shows this topology.

Figure 4, on the other hand, shows the benefit of setting up a system where you can boot the OS from a SAN.
Figure 4. Simple boot from SAN topology

If the active server fails, your business services will be quickly switched to the passive server, as shown in figure 5.

Figure 5. OS failover in boot from SAN environment

You can also use the DS8000’s remote copy service to back up your OS and application to a remote site, as Figure 6 shows.
I've focused on setting up an OS booting from a SAN active server, but the process can be used for a replacement or passive server, too. I won't discuss automatic OS failover in this article.

## Strategy for booting from a SAN

I'll start by introducing the setup strategy.

- On the storage side, there is no difference between a local boot and a SAN boot.
- On the host side, you need to change a few things to differentiate the SAN boot from the local boot:
  - Set up HBA to enable the BIOS and select the boot device from DS8000.
  - Disable the local hard disk through the server BIOS, or physically remove it.
  - Set the server BIOS to keep the SAN OS volume as the first boot candidate.
  - Enable DMMP in Linux.

After doing these steps, Linux on the SAN will be set as the primary candidate disk for booting up when the server powers on and runs through its self check. In addition, as DMMP loads during the OS boot right after the HBA driver, your Linux will have multipath enabled.

## Setting up the storage device

The IBM System Storage DS8000 is designed to support multipath I/O with dual storage processors in the active/active way. This means you don't have to do anything special to set up for the OS boot from SAN DS8000. The procedure goes like so:

- Zone your server with DS8000.
- Add a volume group.
- Create a host connect.
- Make a new volume for the OS.

Here's more detail on these steps:
Zone your server with DS8000
You need to make a switch zone for each path so your server HBA port can reach the DS8000 IO port. As in Figure 1, you need zone hba1 and controller 1 (DS8000 port 1) together on SAN1; and zone hba2 and controller2 (DS8000 port 2) together on SAN2. It looks like the following if you use a soft zone:

Zone 1:
50:05:07:63:07:03:07:7d (DS8000 port 1 WWPN);
21:00:00:1b:32:04:82:b7 (Your server HBA port 1 WWPN)

Zone 2:
50:05:07:63:07:08:07:7d (DS8000 port 2 WWPN);
21:01:00:1b:32:24:82:b7 (Your server HBA port 2 WWPN)

Add a volume group
Let's assume your storage administrator already helped you partition your DS8000 and create an extent pool. You can add a volume group for your server with the following command executed on your DS8000 by CLI:

dscli>mkvolgrp -hosttype LinuxSUSE myvolgrp

Here, you're setting up a volume group named myvolgrp with host type LinuxSUSE, which defines attaching an Intel-based SUSE Linux OS.

Create a host connect
Host connect is used to map your server's HBA WWPN to one dedicated volume group on the DS8000 so that your server can access all volumes in this dedicated volume group. The command goes like this:

dscli>mkhostconnect -wwname 2100001b3204b7ba -hosttype LinuxSUSE -volgrp v35 -ioport I0001 mysusehost

Here, I've created a host connection for one path of my server with WWPN 2100001b3204b7ba, which connects to the DS8000's I/O port I0001 and maps volume group v35 (myvolgrp) to this host connect. This allows this server port access to all volumes in volume group myvolgrp.

Make a new volume for the OS
You need to make a volume to keep your OS data along with enough space for a Linux installation. It's always a good idea to make it big enough for future use:

dscli>mkfbvol -extpool p2 -cap 10 -volgrp v35 -name mysuseboot 9802

In this step, I create a volume named mysuseboot in extent pool p2 and assign it to volume group v35. This volume will be used for loading Linux.
You can also use this command to add more volumes for data or applications, but it’s better to keep the volume you want to use as an OS disk in first place with the lowest ID. And it’s always a good idea to assign more volumes for data or applications later, after you finish configuring Linux.

Now that the storage device is set up, you can set up your server to get ready to install Linux on the SAN.

### Ready the server for SAN OS install

Since you want to install the OS to SAN storage, you need disable your server’s local hard disk and enable the HBA BIOS so it will recognize the disk from the DS8000 during the server boot device check.

The steps are as follows:

- Disable the local disk on the server.
- Enable HBA BIOS to boot from the SAN.
- Select the boot disk from DS8000.

Here is more detail on these steps:

#### Disable the local disk on the server

First, you need to enter the system BIOS setup page; I’ll use the IBM System x server BIOS to illustrate this step. Select *Device and I/O Ports* from the page as in Figure 7.

**Figure 7. IBM System x server BIOS main page**

Then, disable the local hard disk on your server (*Planar SAS* in Figure 8).
If your server doesn't support disabling the local disk adapter directly, you can simply pull out your local hard disk physically.

**Enable the HBA BIOS to boot from the SAN**

I'll use a Qlogic HBA BIOS setting as example here. If you use an HBA from another vendor, refer to their documentation for the correct setting.

First, you need press **Ctrl-Q** to select the *Host Adapter* page when your server posts the Qlogic self-test. You should get something like you see in Figure 9.

**Figure 9. Select Host Adapter page**

You can select the first HBA port, which will bring you to the *Fast!UTIL Options* page, shown in Figure 10.
Enable multipath SAN booting with DS8000 and DMMP

Enter the *Configuration Settings* page (Figure 11) and select *Adapter Settings*.

**Figure 11. Configuration settings**

On the *Adapter Settings* page, you need to enable *Host Adapter BIOS* as in Figure 12.

**Figure 12. Adapter settings**

Then go back to the *Adapter Settings* page (Figure 11) and enter the *Selectable Boot Settings* page as shown in Figure 13.
Figure 13. Selectable boot settings

Select the boot disk from DS8000

From the Selectable Boot Settings page (Figure 13), you need to select the Primary Boot device, which will get you the Select Fibre Channel Device page (as in Figure 14).

Figure 14. Select Fibre Channel device

Now, you need to select the boot device on which to install Linux. The Selectable Boot Settings page is shown in Figure 15.

Figure 15. Selectable boot settings after boot device selection

At this stage, you need to set up the boot device in the Qlogic BIOS. It's like setting up another HBA port, so just follow the same steps. You can get back to the Select Host Adapter page when you escape back to Fast/UTIL Options page (Figure 10).

After selecting a boot device for each HBA or HBA port, you're ready to insert the Linux installation media and start the OS installation.
Installing Linux on the storage device

It's always a good idea to keep only one path alive between your server and DS8000 before you start to install Linux onto the DS8000 volume. To do this, disable one HBA port of your server by either:

- Disabling the corresponding switch port
- Pulling out the HBA cable physically, or
- Disabling all other zones between your server and DS8000

It's better if you assign that volume as the OS install volume before installation. That way, you can always find the right volume to install Linux from since there will only be one volume available at this time.

The installation is much the same as the steps you follow to install Linux to a local hard disk. RHEL5, however, is a special case. You'll need the `linux_mpath` option during the initial installation, because RHEL5 supports enabling DMMP during installation (in other words, you can have multiple paths enabled during installation).

I'll illustrate setup procedures on different x86 platforms (verified on RHEL4 U7, RHEL5 U2, SLES9 SP4, and SLES10 SP2). You might need to use the corresponding packages for your server architecture, such as the i586/i386 packages for 32-bit x86 Linux and x86_64 packages for 64-bit x86 Linux).

Installing the DMMP multipath tool

Now that Linux is installed on your DS8000 volume, you need to install the DMMP multipath tool if you didn't do it during the OS installation. DMMP contains two packages:

- device-mapper
- multipath-tool

Normally, the device-mapper package is always installed by default, because it's needed by other tools such as Logical Volume Manager (LVM). On SLES you can install the multipath-tools package from the distribution disk like so:

```bash
<Linux ~>$ rpm -ivh multipath-tools-0.4.7-75.7.i586.rpm
```

On Red Hat:

```bash
<Linux ~>$ rpm -ivh device-mapper-multipath-0.4.5-31.el4.x86_64.rpm
```

Enabling DMMP on system boot

You now have all the necessary software ready, and you can configure your system using the following two steps to enable DMMP and add DS8000 support:

- Enable DMMP startup during boot.
- Configure DMMP to support the storage device.
Enabling DMMP startup during boot
Add the multipath-tools startup scripts to the startup list as follows:

```
<linux:~> # chkconfig --level 35 multipathd on
```

Verify that it's set up correctly:

```
<linux:~> # chkconfig --list | grep multipath
multipathd 0:off 1:off 2:off 3:on 4:off 5:on 6:off
```

Configuring DMMP to support the storage device
Edit the `/etc/multipath.conf` file with following items:

Listing 1. Configuring DMMP to support DS8000

```
defaults {
    polling_interval 30
    failback immediate
    no_path_retry 5
    rr_min_io 100
    path_checker tur
    user_friendly_names yes
}
devices {
    # DS8000
    device {
        vendor "IBM"
        product "2107900"
        path_grouping_policy group_by_serial
    }
}
multipaths {
    multipath {
        wwid 36005076307ffcc77d000000000001050
        alias OSROOT
    }
}
```

This configuration file is available for download.

Now configure an alias for the Linux OS root disk by WWID (World Wide Identifier) so you can use a short and friendly name in the upcoming configuration steps. To get a WWID for your OS disk, use following command:

```
<linux:~> # scsi_id -g -u -s /block/sda 36005076307ffcc77d000000000001050
```

`sda` is your OS disk device name without DMMP.

Building DMMP modules into the initrd image
In this section, I show how to build DMMP modules into the initrd image—the initial ramdisk, a temporary file system commonly used in the boot process of the Linux kernel for making preparations before the real root file system can be mounted—for SLES9, SLES10, RHEL4, and RHEL5. Pick your OS and follow the steps.
The steps for SLES9
By default, DMMP is not built into the SLES9 initrd image to support a multipath-enabled SLES9 boot from a SAN. To make it work, do the following:

Edit the /etc/sysconfig/kernel file
Add `dm_mod dm_multipath dm_round_robin` to the line for `INITRD_MODULES`:

```
INITRD_MODULES="ata_piix qla2400 dm_mod dm_multipath dm_round_robin reiserfs"
```

Make a new initrd image with DMMP support
```bash
<linux:~> # mkinitrd -k /boot/vmlinuz -i initrdmp.gz -x 6000
```
At this point I build all the needed DMMP modules (`dm_mod`, `dm_multipath`, `dm_round_robin`) to the initrd image so that DMMP will load them during SLES9’s bootup. I also add 6,000 bytes to the image for further editing space.

The steps for RHEL4/5
DMMP isn’t build into the initrd image, so you can use following commands to do it:

```bash
<linux:~> # mkinitrd --with=dm-mod --with=dm-multipath --with=dm-round-robin initrdmp.gz `uname -r`
```
Here, initrdmp.gz is the initrd image name. And if you use the `linux mpath` option during the RHEL5 install, you don’t need to run this step.

The steps for SLES10
You can use either of the following two methods to achieve this for SLES10.

Build a new initrd image with DMMP embedded in it
Run the following command to build a new initrd image with DMMP embedded:

```bash
<linux:~> # mkinitrd -f mpath
```

Edit the /etc/sysconfig/kernel file
Edit the /etc/sysconfig/kernel file and add `dm-multipath` to the line for `INITRD_MODULES` as follows:

```
INITRD_MODULES="ata_piix qla2400 reiserfs dm-multipath"
```
Then run the following command to build DMMP into the initrd image:

```bash
<linux:~> # mkinitrd
```

Editing the initrd image
Now you need to edit the initrd image. Choose your OS (SLES9, SLES10, RHEL4, or RHEL5) and follow the steps.
The steps for SLES9

Unzip the initrd image

```
<linux:~> #mkdir tmp; cd tmp
<linux:~/tmp> #cp /boot/initrdmp.gz ./
<linux:~/tmp> #gunzip ./initrdmp.gz
```

This initrd image name is what I used in previous steps; it may differ from what you used.

Mount the initrd image to a temp directory

```
<linux:~/tmp> #mkdir temp
<linux:~/tmp> #mount -t ext2 -o loop ./initrdmp temp
```

Copy multipath.conf to the initrd image

```
<linux:~/tmp> #cd temp
<linux:~/tmp> #cp /etc/multipath.conf temp/etc/
```

Copy the DMMP-related binary and library to the initrd image

```
<linux:~/tmp> # cp /sbin/multipath sbin
<linux:~/tmp> # cp /sbin/dmsetup sbin/
<linux:~/tmp> # cp /sbin/kpartx sbin/
<linux:~/tmp> # cp /sbin/mpath_* sbin/
<linux:~/tmp> # cp /sbin/lvm sbin/
<linux:~/tmp> # cp /sbin/vgscan sbin/
<linux:~/tmp> # cp /sbin/vgchange sbin/
<linux:~/tmp> # cp /sbin/devmap_name sbin/
<linux:~/tmp> # cp /bin/awk bin/
<linux:~/tmp> # cp /lib/libdevmapper.so.1.0.1 lib/
<linux:~/tmp> # cp /lib/libsysfs.so.1 lib/
<linux:~/tmp> # mkdir lib/tls
<linux:~/tmp> # cp /lib/tls/libc.so.6 lib/tls
<linux:~/tmp> # cp /lib/tls/libm.so.6 lib/tls
<linux:~/tmp> # cp /lib/tls/libselinux.so.1 lib/
<linux:~/tmp> # cp /lib/tls/libdl.so.2 lib/tls
<linux:~/tmp> # cp /lib/tls/libc.so.6 lib/tls
<linux:~/tmp> # cp /lib/tls/libsysfs.so.1 lib/
```

In this step, if you don’t want use LVM to manage your root partition, then you can ignore copying lvm, vgscan, vgchange, and related lib. And if you are using X86_64 SLES9, you need to copy related libraries under the lib64 directory.

Edit the linuxrc file

Find the last driver-loading line (like `dm-round-robin.ko`, in my case):
insmod /lib/modules/2.6.5-7.308-bigsmp/kernel/drivers/md/dm-round-robin.ko

Add the following after the last driver-loading line:

Listing 2. Addition to linuxrc

```
multipath -v2
for i in `dmsetup ls | awk '{print $1}'`; do
  kpartx -a -v /dev/mapper/$i
done
```

If you don't like verbose output, you can remove the `-v` parameter for `multipath` or `kpartx`.

Repackage the initrd image

```
<linux:~/>umount temp
<linux:~/>gzip initrdmp
<linux:~/>mv initrdmp.gz /boot/
```

The steps for RHEL4

Extract the initrd image to a temp directory

```
<linux host ~>mkdir temp
<linux host ~>cd temp
<linux host ~>gzip -dc ../initrdmp.gz | cpio -id
```

Copy multipath.conf to the initrd image

```
<linux host ~>cp /etc/multipath.conf temp/etc/
```

Copy the DMMP binaries to the initrd image

```
<linux host ~>cp /sbin/dmsetup.static sbin/dmsetup
<linux host ~>cp /sbin/multipath.static sbin/multipath
<linux host ~>cp /sbin/kpartx.static sbin/kpartx
<linux host ~>cp /sbin/scsi_id.static sbin/scsi_id
<linux host ~>cp -a /etc/udev/rules.d/ etc/udev/
```

Edit the init file

Add following two lines just after all drivers load:

```
/sbin/multipath -v2
/sbin/dmsetup ls --target multipath --exec "/sbin/kpartx -a"
```

Repackage the initrd image

```
<linux host ~>find ./ | cpio -H newc -o > ../initrdmp
```
The steps for RHEL5 and SLES10

Since RHEL5 and SLES10 support DMMP, you only need to add multipath.conf to the initrd image to support DS8000.

Extract the initrd image to a temp directory

```
<linux host ~> # gzip initrdmp
<linux host ~> # cp initrdmp.gz /boot/
```

Copy multipath.conf to the initrd image

```
<linux host ~> # cp /etc/multipath.conf temp/etc/
```

Repackage the initrd image

```
<linux host ~> # find ./ | cpio -H newc -o > ../initrdmp
<linux host ~> # gzip initrdmp
<linux host ~> # cp initrdmp.gz /boot/
```

Adding LVM2 support

If you install Linux with LVM2 to manage your root file system, you will need to configure your LVM configuration file, lvm.conf, to support the DMMP device and copy this lvm.conf to the initrd image.

Your lvm.conf file should have two lines that look like this:

```
filter = [ "a|^/dev/mapper/.*|", "r|.*|" ]
types = [ "device-mapper", 253 ]
```

Editing the GRUB configuration

Some Linux distributions do not support device names under the /dev/mapper directory, so I use /dev/disk/by-name instead. Devices under this directory are mapped to devices under the /dev/mapper directory. Edit the /boot/grub/menu.lst file and change the kernel booting section like so (all on one line):

```
kernel (hd0,1)/boot/vmlinuz root=/dev/disk/by-name/OSROOT2 vga=0x314 selinux=0
splash=silent resume=/dev/disk/by-name/OSROOT1 elevator=cfq showopts
```

Here, OSROOT2 means the original sda2 and OSROOT1 means the original sda1. If you are using SLES10, it uses names like OSROOT-part2. Since this friendly name may change in different Linux releases, it's better to get its exact name format by assigning one additional LUN from the DS8000 and partitioning it after DMMP adds it to the device map. The reason for the additional LUN is that...
the root file system is in use at this time and DMMP cannot add it to the device map; therefore, you cannot get that name for the root partition.

**Editing fstab**

Edit the `/etc/fstab` file to replace original `sdx` device with a DMMP device, like so:

```
/dev/mapper/OSROOT2 / reiserfs acl,user_xattr 1 1 /dev/mapper/OSROOT1 swap swap pri=42 0 0
```

Here, `/dev/mapper/OSROOT2` replaces `/dev/sda2` and `/dev/mapper/OSROOT1` replaces `/dev/sda1`. Please don't use a label here. Linux will find duplicate labels because you have multiple paths for each file system, which will prevent your file systems from automatically mounting.

**Verifying the DMMP boot**

Almost done! You have finished all the steps needed to set up Linux to boot from a multipath SAN. You can now enable all paths between your server and DS8000 and then reboot Linux. After the server is rebooted, you should verify whether DMMP works using the command `multipath -ll`.

The output should looks like this:

```
OSROOT (36005076307ff77d00000000000001050) dm-0 IBM,2107900
[size=10G][features=1 queue_if_no_path][hwhandler=0]
  \_ round-robin 0 [prio=2][active]
  \_ 2:0:0:0 sda 8:0 [active][ready]
  \_ 3:0:0:0 sdb 8:16 [active][ready]
```

If you disable one path, it will change the status to show one path has failed:

```
OSROOT (36005076307ff77d00000000000001050) dm-0 IBM,2107900
[size=10G][features=1 queue_if_no_path][hwhandler=0]
  \_ round-robin 0 [prio=1][active]
  \_ 2:0:0:0 sda 8:0 [failed][faulty]
  \_ 3:0:0:0 sdb 8:16 [active][ready]
```

The faulty path will go back to ready status if you re-enable it.

You can also try disabling either path and rebooting your server to verify whether it can boot up with only one path alive. If nothing bad happens, you can get your server up and the `multipath -ll` output will look like the following:

```
OSROOT (36005076307ff77d00000000000001050) dm-0 IBM,2107900
[size=10G][features=1 queue_if_no_path][hwhandler=0]
  \_ round-robin 0 [prio=1][active]
  \_ 3:0:0:0 sdb 8:16 [active][ready]
```

After you add back the disabled path, you need to run `multipath -v0` to add it back to the path list after your HBA driver finds a new enabled path.
You can add more volumes to your volume group on the DS8000 for data or application usage. And you can pull a Fibre Channel cable between your servers and switch to simulate a broken FC cable; you can disable one switch port of your server/storage to simulate a switch failure. In both cases, your server I/O will only freeze for a few seconds, because DMMP is taking care of the problem. After you restore them back to their functional state, DMMP will reestablish a connection to the previous broken path and rebalance I/O to it.

Finally, remember that if a dual path does not provide sufficient reliability, availability, and performance, you can add more paths.

**Conclusion**

I've showed you how to set up Linux to boot from DS8000 SAN storage with DMMP multipath support. If you want to apply these techniques to other storage vehicles (XIV, SAN Volume Controller, the DS6000, etc.), remember that the configuration files will be different from the multipath.conf I used in this article.
Related topics

- This online manual shows you all the tricks to administering, configuring, and using Red Hat's DMMP.
- IBM System Storage DS8000 is high-performance disk storage for business-critical enterprise workloads.
- The `/etc/multipath.conf` file is available at the IBM system storage DS8000 Subsystem Device Driver for Linux download page.
- In the developerWorks Linux zone, find more resources for Linux developers, and scan our most popular articles and tutorials.
- See all Linux tips and Linux tutorials on developerWorks.

© Copyright IBM Corporation 2009, 2010
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)