Technical Report

NetApp SolidFire for Linux Configuration Guide

SolidFire Engineering Team, NetApp

October 2017 | TR-4639

TABLE OF CONTENTS

LIST OF FIGURES

1 Introduction

This document describes how to configure a NetApp® SolidFire® storage system and Linux hosts that can be used to connect to a SolidFire volume. It also provides best practices and implementation recommendations for this type of configuration.

The most common Linux operating systems are Ubuntu, Debian, CentOS, SuSE, and Red Hat Enterprise Linux. The examples and commands used in this document are similar across these operating systems.

Before you can set up your Linux host, you must first set up a SolidFire system account and subsequent volumes.

If you are using a database, see our best practices guides for the database that you are using. NetApp recommends tuning the iSCSI timeout values to avoid any connection errors in the event of a SolidFire node failure.

Note: This document assumes that you have access to the SolidFire Element® OS web UI.

1.1 Creating a SolidFire Account

Each SolidFire account represents a unique volume owner and receives its own set of Challenge-Handshake Authentication Protocol (CHAP) credentials. You can access volumes assigned to an account either by using the account name and the relative CHAP credentials or through a volume access group.

Procedure

1. Log in to the Element OS web UI.
2. Go to Management > Accounts.
3. Click Create Account.
4. Enter the CHAP user name to be used with the Windows host in the Username field.
5. In the CHAP Settings section:
   a. Enter the initiator secret for CHAP node session authentication.
   b. Enter the target secret for CHAP node session authentication.

   Note: Leave these fields blank to autogenerate the secrets. To view them, click Actions > View Details.
6. Click Create Account.

1.2 Creating a SolidFire Volume

After provisioning an account, you must create volumes and associate them with the account. This enables iSCSI initiators by using the provided CHAP credentials to discover and mount iSCSI devices that are associated with that account in Element OS. In the case of volumes that are connected by Fibre Channel (FC), the account serves as a container for the volumes providing individual account statistics and segmentation.

Procedure

1. Go to Management > Volumes.
2. Click Create Volume.
3. In the Create a New Volume dialog box, enter the volume name (1 to 64 characters in length).
4. Enter the total size of the volume.

   Note: The default volume size selection is in GB. Volumes can be created with GB or GiB:

   – 1GB = 1,000,000,000 bytes.

- 1GiB = 1,073,741,824 bytes.

5. Select a block size for the volume.

   This option is necessary to support operating systems and applications that do not recognize native 4K drives, such as VMware ESXi.

   A SolidFire storage system operates on a native 4K block size, yet for compatibility with certain hypervisors and operating systems, it can emulate as a 512-byte block size volume by wrapping up eight 512-byte blocks and writing them as a single 4K block. When creating volumes for use with VMware, Citrix XenServer, and Microsoft Hyper-V prior to 2012, you must enable 512-byte emulation. You should determine whether your specific OS or application supports the 4K block size and set this option accordingly. In some cases, the OS supports 4K blocks but the application does not. In this case, you should use 512-byte blocks to allow support for the application.

6. Click Account and select the account from the list that should have access to the volume. If an account does not exist, click the Create Account link, enter a new account name, and click Create. The account is created and associated with the new volume.

   Note: If there are more than 50 names, the list does not appear. Begin typing, and the autocomplete function displays possible values for you to choose from.

7. Set the Quality of Service values or accept the default values.

   Use the spin box in each column to select the desired IOPS values.

   Caution: Volumes with MaxIOPS and BurstIOPS greater than 20,000 are specifically allowed to accommodate higher bandwidths. Achieving greater than 20,000 small block IOPS on a single volume requires a high queue depth and might require special multipath input/output (MPIO) configuration.

8. Click Create Volume.

1.3 OS Distributions

This guide uses generalized names for the various OS distributions to make common commands easier to identify. The included architectures are noted in the following list. For information about supported versions, see the SolidFire Element OS release notes.

- Debian: Debian and Ubuntu Linux
- SUSE Linux Enterprise Server (SLES)
- Fedora: Red Hat Enterprise Linux, CentOS, and Oracle Linux

1.4 Installing the iSCSI Initiator Software

To connect directly to a SolidFire volume, you must install the iSCSI initiator. After it is installed, you have three options to authenticate your connection with the SolidFire volumes. For more details, see the section “Access Control and Authentication for SolidFire Volumes.”

Procedure

Run the following commands for the OS you are running:

- Debian#:

```
sudo apt-get install –y open-iscsi
```

- SUSE Linux Enterprise Server#:

```
sudo zypper install –y open-iscsi
```

- SUSE Linux Enterprise Server 11 SP4#:

```
1.  sudo zypper install –y open-iscsi
2.  sudo /sbin/chkconfig open-iscsi on
```

```
3.  sudo /etc/init.d/open-iscsi start
```

- Fedora#:

```
sudo yum install –y iscsi-initiator-utils
```

1.5 Enabling Network Interfaces at Boot

You must make sure that all required network interfaces for your client are enabled at boot time. The `/etc/network/interfaces` file controls which interfaces become active during boot and what IP address information is assigned to them. If this is not done automatically by your operating system, use the following procedure to manually make the change.

Procedure

1. Start a command line interface on the Linux iSCSI client.
2. Edit the `/etc/network/interfaces` file with a text editor.
3. Add any required interfaces (`eth1`, `eth2`, and so on) to the auto line.
4. Create an iface definition section for each additional interfaces defined on the auto line.
5. Save the file.
6. Reboot the client system to test the changes.

For more information about the syntax of the `/etc/network/interfaces` file, see the interfaces man page for your distribution. You can also use the following example for guidance:

```
auto eth0 eth1
iface eth0 inet static address 192.0.2.7 netmask 255.255.255.0 gateway 192.0.2.254
iface eth1 inet static address 192.0.2.8 netmask 255.255.255.0 gateway 192.0.2.254
```

1.6 Installing Multipath I/O Software

MPIO software is required for connections to SolidFire series FC volumes because it offers a way to consolidate the multiple paths available to an FC volume into a single usable device. MPIO is also useful when connecting multiple sessions to iSCSI volumes if you need a queue depth greater than the single iSCSI connection queue depth of 32. For example, if your application can drive a queue depth of 128, you could connect four iSCSI sessions and leverage MPIO for an aggregate queue depth of 128.

Procedure

Run the following commands for the OS you are running:

- Debian#:

```
sudo apt-get install -y multipath-tools
```

- SUSE Linux Enterprise Server

1. `sudo chkconfig multipathd on`
2. `sudo chkconfig boot.multipath on`
3. `sudo /etc/init.d/boot.multipath start`
4. `sudo /etc/init.d/multipathd start`

- SUSE Linux Enterprise Server 11 SP4

1. `sudo /sbin/chkconfig multipathd on`
2. `sudo /sbin/chkconfig boot.multipath on`

- Fedora

1. `sudo yum install –y device-mapper-multipath`

```
2.  sudo modprobe dm-multipath
3.  sudo service multipathd start
4.  sudo chkconfig multipathd on
```

- Red Hat Enterprise Linux 7+

```
1.  sudo yum install –y device-mapper-multipath
2.  sudo modprobe dm-multipath
3.  sudo systemctl enable multipathd.service
```

2 Access Control and Authentication for SolidFire Volumes

Access control determines which SolidFire volumes a given iSCSI initiator can access. SolidFire offers two forms of access control to hosted volumes, accounts, and volume access groups:

- Account-based access control
- volume access group access control

If the iSCSI initiator is configured to use CHAP authentication, account-based access control is used. If the iSCSI initiator is not configured to use CHAP authentication, then volume access group access control is used. You can use CHAP authentication (verification that the initiator is the intended volume user) only with account-based access control.

- **Option 1: volume access groups.** Volume access groups provide access control between a list of iSCSI initiator iSCSI qualified names (IQNs) or FC worldwide port names (WWPNs) and an associated group of volumes. Note that this method does not provide account name or secret authentication. Therefore, if you enter an initiator incorrectly, the host might have access to more volumes than intended. Volume access groups might contain volumes from more than one account.
- **Option 2: unidirectional CHAP.** CHAP leverages an account name and secrets to authenticate an initiator to a target. The use of CHAP provides for both access control (limiting access to specific volumes) and authentication (verifying that the specific initiator presents the correct credentials for access to the volume). With unidirectional CHAP, the initiator authenticates with the target.
- **Option 3: bidirectional CHAP.** With bidirectional CHAP, the initiator authenticates with the target, and the target authenticates with the initiator.

Note: This section applies only to server deployments connecting directly to a SolidFire volume. For virtual servers, refer to the Configuring VMware vSphere for Element OS guide for connecting through a hypervisor layer.

2.1 Option 1: Volume Access Groups

A SolidFire volume access group contains a list of IQNs or FC WWPNs that can access the volume without CHAP user ID and password authentication.

Finding the Initiator IQNs

You need the server initiator IQN to associate the server to the volumes by using a SolidFire volume access group.

Procedure

Run the following command to show the initiator names:

```
sudo cat /etc/iscsi/initiatorname.iscsi
```

Example

The IQN (initiator name) is noted in red.

```
## DO NOT EDIT OR REMOVE THIS FILE!
## If you remove this file, the iSCSI daemon will not start.
## If you change the InitiatorName, existing access control lists
## may reject this initiator.  The InitiatorName must be unique
## for each iSCSI initiator.  Do NOT duplicate iSCSI InitiatorNames. InitiatorName=iqn.1993-
08.org.debian:01:c2d8345a1425
```

Note: The IQN is required when Creating an Access Group or Adding Volumes to an Access Group.

Finding the Initiator WWPNs

You need the server initiator WWPN to associate the server to the volumes by using a SolidFire volume access group.

Procedure

Run the following command (this command is used by all Linux distributions):

```
sudo cat /sys/class/scsi_host/host*/device/fc_host/host*/port_name
```

Example

The WWPN is noted in red.

```
[admin1@test  ~]# sudo cat /sys/class/scsi_host/host*/device/fc_host/ host*/port_name
0x2100000e1e11f8c0
0x2100000e1e11f8c1
```

Note: The WWPN is required when Creating an Access Group or Adding Volumes to an Access Group.

Creating an Access Group

Clients can discover volumes through volume access groups, which allow initiator access to volumes without requiring CHAP authentication. You can set up a volume access group by using an IQN or a WWPN. After you create the volume access group, you can assign volumes to the volume access group to create a volume collection.

When creating an access group, note the following:

- An access group can contain a maximum of 64 initiator IQNs or WWPNs. Any initiator in the volume access group can access any volume in the volume access group.
- An IQN or WWPN can belong to only one access group.
- A single volume can belong to a maximum of four access groups.

Procedure

1. Open the SolidFire Element OS web UI.
2. Go to Management > Access Groups.
3. Click Create Access Group.
4. Enter a name for the volume access group in the Name field.
5. To add an FC initiator to the volume access group, complete the following steps:
    a. Under Add Initiators, select an existing FC initiator from the Unbound FC Initiators list.
    b. Click Add FC Initiator.

    Note: SolidFire systems only display WWPNs that are properly zoned, are visible on the FC network, and have not been previously added to another volume access group.

6. To add an iSCSI initiator to the volume access group, select an existing initiator from the Initiators list under Add Initiators.

   Note: You can create an initiator during this step by clicking the Create Initiator link, entering an initiator name, and clicking Create. The system automatically adds the initiator to the Initiators list after you create it.

The accepted format of an initiator IQN is `iqn.yyyy-mm`, where y and m are digits, followed by text, which must only contain digits, lowercase alphabetic characters, periods (.), colons (:), and dashes (-).

Example

```
iqn.1993-08.org.debian:01:6d8652abb76
```

Tip: You can find the initiator IQN for each volume by selecting View Details in the Actions menu for the volume on the Management > Volumes > Active list.

7. (Optional) Add more initiators as needed.
8. Under Attach Volumes, select a volume from the Volumes list. The volume appears in the Attached Volumes list.
9. (Optional) Add more volumes as needed.
10. Click Create Access Group.

Adding Initiators to an Access Group

You can add an initiator to an access group to allow access to volumes in the volume access group without requiring CHAP authentication. When you add an initiator to a volume access group, the initiator has access to all volumes in that volume access group.

Procedure

1. Go to Management > Access Groups.
2. Click the Actions button (⚙) for the access group you want to edit.
3. Click the Edit button (✎).
4. To add an FC initiator to the volume access group, complete the following steps:
   a. Under Add Initiators, select an existing FC initiator from the Unbound FC Initiators list.
   b. Click Add FC Initiator.
5. To add an iSCSI initiator to the volume access group, complete the following steps:
   a. Under Add Initiators, select an existing initiator from the Initiators list.
   b. Click Add Initiator.

   Note: You can create an initiator during this step by clicking the Create Initiator link, entering an initiator name, and clicking Create. The system automatically adds the initiator to the Initiators list after you create it.

Tip: You can find the initiator IQN for each volume by selecting View Details in the Actions menu for the volume on the Management > Volumes > Active list.

6. (Optional) Repeat steps 4 and 5 to add more initiators as needed.
7. Click Save Changes.

Adding Volumes to an Access Group

You can add volumes to a volume access group. Volumes can belong to more than one volume access group. You can see the groups to which each volume belongs in the Active Volumes window.

Note: You can also follow these steps to add volumes to an FC volume access group.

Procedure

1. Go to Management > Access Groups.
2. Choose an access group and click the Actions button (⚙).
3. In the resulting menu, click the Edit button (✏).
4. Under Add Volumes, select a volume from the Volumes list.
5. Click Attach Volume.
6. Repeat steps 5 and 6 to add more volumes as needed.
7. Click Save Changes.

Configuring iSCSI to Use the Volume Access Group

Now that you have a volume access group with associated volumes, you must configure iSCSI on the Linux system to access those volumes.

Prerequisites

- Make sure that a SolidFire account and associated volumes exist.
- Note the existing volume access group containing the IQNs of the servers in question and the volumes to be accessed.
- Open iSCSI must be installed and running on the host operating system in accordance with the guidelines in the section “Installing the iSCSI Initiator Software.”

Procedure

1. Open the `/etc/iscsi/iscsid.conf` file for editing and change the following lines to make sure that CHAP authentication is not used by prepending a hash sign (#) before each line.

   Note: Edit this file as the root user or obtain root permissions by using the "`sudo`" command.

```
# node.startup = manual

# node.session.auth.authmethod = CHAP
# node.session.auth.username = username
# node.session.auth.password = password

# discovery.sendtargets.auth.authmethod = CHAP
# discovery.sendtargets.auth.username = username
# discovery.sendtargets.auth.password = password
```

2. When editing the `/etc/iscsi/iscsid.conf` file, uncomment the following lines by removing the hash sign (#) from each line:

```
node.startup = automatic
```

3. Depending on the Linux distribution, restart the iSCSI service by running the following command:

- Debian/SuSE#:

```
sudo /etc/init.d/open-iscsi restart
```

- Fedora#:

```
sudo /etc/init.d/iscsi restart
```

- Systemd#:

```
sudo systemctl restart iscsi
```

2.2 Option 2: Unidirectional CHAP

The unidirectional CHAP option authenticates volume access by using the SolidFire account name and initiator secret.

Prerequisites

- Make sure that a SolidFire account and associated volumes exist.
- Open iSCSI must be installed and running on the host operating system in accordance with the guidelines in the section "Installing the iSCSI Initiator Software."

Procedure

1. Open the `/etc/iscsi/iscsid.conf` file for editing and uncomment the following lines by removing the hash sign (#) from each line:

```
node.startup  = automatic

node.session.auth.uthmethod = CHAP
node.session.auth.username = <SolidFire Account Name>
node.session.auth.password = <SolidFire Account Initiator Secret>

discovery.sendtargets.auth.authmethod = CHAP
discovery.sendtargets.auth.username = <SolidFire Account Name>
discovery.sendtargets.auth.password = <SolidFire Account Initiator Secret>
```

2. When editing the `/etc/iscsi/iscsid.conf` file, change the following lines to comments by prepending a hash sign (#) before each line:

```
#  node.startup = manual
```

3. Depending on the Linux distribution, restart the iSCSI service by running the following command:

- Debian/SuSE#:

```
sudo /etc/init.d/open-iscsi  restart
```

- Fedora#:

```
sudo /etc/init.d/iscsi  restart
```

- Systemd#:

```
sudo systemctl  restart  iscsi
```

2.3 Option 3: Bidirectional CHAP

The bidirectional CHAP option provides the most secure way of authenticating the volume, but it also requires the most configuration. With this method, the volume authenticates the host through the account name and the initiator secret, and then the host authenticates the volume through the account name and the target secret.

Note: For the best security, the initiator secret and target secret should be different.

Prerequisites

- Make sure that a SolidFire account and associated volumes exist.
- Open iSCSI must be installed and running on the host operating system in accordance with the guidelines in the section "Installing the iSCSI Initiator Software."

Procedure

1. Open the `/etc/iscsi/iscsid.conf file` for editing and uncomment the following lines by removing the hash sign (#) from each line:

```
node.startup = automatic
node.session.auth.authmethod = CHAP
node.session.auth.username = <SolidFire Account Name>
node.session.auth.password = <SolidFire Account Initiator Secret>
node.session.auth.username_in = <SolidFire Account Name>
node.session.auth.password_in = <SolidFire Account Target Secret>

discovery.sendtargets.auth.authmethod = CHAP
discovery.sendtargets.auth.username = <SolidFire Account Name>
discovery.sendtargets.auth.password = <SolidFire Account Initiator Secret>
discovery.sendtargets.auth.username_in = <SolidFire Account Name>
discovery.sendtargets.auth.password_in = <SolidFire Account Target Secret>
```

2. While editing the `/etc/iscsi/iscsid.conf` file, change the following lines to comments by prepending a hash sign (#) before each line:

```
# node.startup = manual
```

3. Depending on the Linux distribution, restart the iSCSI service by running the following command:

- Debian/SuSE#:

```
sudo /etc/init.d/open-iscsi restart
```

- Fedora#:

```
sudo /etc/init.d/iscsi restart
```

2.4 Discovering and Logging in to the iSCSI Volume

This section details the steps necessary to discover the SolidFire volumes and log in to them by using a single iSCSI session.

Prerequisites

- You have added at least one volume to the configured account or volume access group.
- You have configured authentication through volume access groups, unidirectional CHAP, or bidirectional CHAP.

Procedure

1. After the volume and initiator are set up to authenticate each other, run the following command to discover the target volumes:

```
sudo iscsiadm –m discovery –t sendtargets –p <SolidFire SVIP>
```

For SLES 11 SP4#:

```
sudo /sbin/iscsiadm -m discovery -t sendtargets -p <SolidFire SVIP>
```

2. Run the following command to log in to the discovered target volumes:

```
sudo iscsiadm –m node –L all
```

For SLES 11 SP4#:

```
sudo /sbin/iscsiadm -m node -L all
```

2.5 Connecting Multiple iSCSI Sessions to a Single Volume

This section details the steps required to connect multiple iSCSI sessions to a single SolidFire iSCSI volume. Multiple iSCSI sessions are useful in two scenarios. In scenario one, you want to leverage two physical network interface controllers (NICs) for your iSCSI traffic. In scenario two, you want to increase the aggregate queue depth to a single volume. SolidFire iSCSI sessions support a queue depth of 32 per session. You can use one of three different methods:

- **Method 1** leverages functionality built into open iSCSI to create multiple sessions automatically. NetApp recommends this method if it works on your system because it is simpler to set up and use.
- **Method 2** manually creates iSCSI iface entries corresponding to the physical (or virtual) NICs on your system. This method should be used for systems that do not support the first method (for example, Ubuntu) or when you need to tie iSCSI traffic to specific NICs. For example, two NICs are connected to two switches for high availability (HA).
- **Method 3** manually defines the number of sessions per iSCSI target. This method is useful in cases in which you want more sessions for specific iSCSI volumes, but not all the volumes on the system.

Note: In some situations, the use of MPIO in conjunction with other system utilities such as Logical Volume Manager (LVM) can result in unexpected behavior and performance.

Note: Procedures 1 and 3 are not supported by Ubuntu 12.04.

Procedure 1: Open iSCSI nr_sessions

Note: Edit this file with the root user or a user with root permission by using the "`sudo`" command.

1. Edit the `/etc/iscsi/iscsid.conf` file and update the following entry:

```
node.session.nr_sessions = 1
```

Enter the number of iSCSI sessions needed per volume.

```
node.session.nr_sessions = <desired  sessions>
```

2. Depending on the Linux distribution, restart the iSCSI service by running the following command:
   – Debian/SuSE#:

```
sudo /etc/init.d/open-iscsi restart
```

   – Fedora#:

```
sudo /etc/init.d/iscsi restart
```

3. If the targets have already been attached, discover the volumes again.
4. Log in to the volumes again.

Procedure 2: Manually Create iSCSI iface Entries

Note: See the section “Creating Routing Tables.”

1. Use iscsiadm to create iSCSI iface entries.

The general format is as follows:

```
sudo iscsiadm -m iface -I ifaceN -o new
sudo iscsiadm -m iface -I ifaceN --op=update -n iface.net_ifacename -v ethXYZ
```

The SUSE Linux Enterprise Server 11 SP4# format is as follows:

```
sudo /sbin/iscsiadm -m iface -I ifaceN -o new
sudo /sbin/iscsiadm -m iface -I ifaceN --op=update -n iface.net_ifacename -v ethXYZ
```

Note: The `N` in `ifaceN` should be replaced with an identifier of your choice. "`ethXYZ`" should be replaced with the appropriate NIC identifier, not just the XYZ portion.

The following example is for most distributions. For SLES11 SP4#, change the sudo prefix to "`sudo /sbin/`."

Example

```
sudo iscsiadm -m iface -I iface0 -o new
sudo iscsiadm -m iface -I iface0 --op=update -n iface.net_ifacename -v eth1
sudo iscsiadm -m iface -I iface1 -o new
sudo iscsiadm -m iface -I iface1 --op=update -n iface.net_ifacename -v eth2
sudo iscsiadm -m iface -I iface2 -o new
sudo iscsiadm -m iface -I iface2 --op=update -n iface.net_ifacename -v eth1
sudo iscsiadm -m iface -I iface3 -o new
sudo iscsiadm -m iface -I iface3 --op=update -n iface.net_ifacename -v eth2
```

2. Depending on the Linux distribution, restart the iSCSI service by running the following command:
   – Debian/SuSE#:

```
sudo /etc/init.d/open-iscsi restart
```

   – Fedora#:

```
sudo /etc/init.d/iscsi restart
```

After you have completed either of these procedures, you should rediscover and relog in to the iSCSI volumes with the steps outlined previously. You see one login per iSCSI session.

Procedure 3: Manually Define Sessions for Each iSCSI Target: Method 1

Note: The following procedure is for most distributions. For SLES11 SP4#, change the sudo prefix to "sudo /sbin/.”

1. Use iscsiadm to discover iSCSI volumes.

```
sudo iscsiadm –m discovery –t sendtargets –p <SolidFire SVIP>
```

2. Use iscsiadm to set the number of sessions per iSCSI target.

```
sudo iscsiadm -m node -T <target name> -p <IP>,<Port> --op update -n node.session.nr_
sessions -v <num sessions>
```

3. Log in to the volumes.

```
sudo iscsiadm –m node –L all
```

This example creates four sessions on a single volume.

```
sudo iscsiadm -m node -T iqn.2010-01.com.solidfire:wegg.vol01.1 -p 10.10.10.1,3260 --op update -n
node.session.nr_sessions -v 4
```

Manually Define Sessions for Each iSCSI Target: Method 2

Depending on the Linux distribution, restart the iSCSI service by running the following command:

- Debian/SuSE#:

```
sudo /etc/init.d/open-iscsi restart
```

- Fedora#:

```
sudo /etc/init.d/iscsi restart
```

Creating Routing Tables

On all Linux operating systems, using multiple interfaces for paths to a single IP presents some issues that can be overcome by creating routing tables for each interface. These commands should be run (prefixed with "sudo" or run as root) for each interface you want to utilize to access specific array IP addresses.

Note: In this procedure, replace the <subnet with SVIP/CIDR> text with your subnet/CIDR containing your SVIP: for example, 10.30.64.0/19.

Procedure

1. Create a table with a different table ID for each interface.

```
/sbin/ip route flush table 10
```

2. Create routes for the array in each table.

```
/sbin/ip route add table 10 to <subnet with SVIP/CIDR> via <gateway> dev ethXYZ
```

3. Create rules with source IPs of the matching eth interface tied to a specific table.

```
/sbin/ip rule add from <IP of interface> table 10 priority 10
```

After these commands are run, the discovery process should not time out, because the packets tied to each interface should both leave and arrive by the proper interface. These commands must be saved in a file to be processed on boot after network initialization so that a system can come up reliably and connect to targets with each boot.

2.6 Listing Connected (Logged In) iSCSI Volumes

This section provides instructions for listing the iSCSI volumes currently connected (logged in) to the host system.

Prerequisites

You are connected (logged in) to SolidFire iSCSI volumes.

Procedure

1. To list all iSCSI volumes that are currently logged in, run the following command:

```
sudo iscsiadm –m node
```

For SUSE Linux Enterprise Server 11 SP4#, run the following command:

```
sudo /sbin/iscsiadm –m node
```

2. To find the mapping of the iSCSI volumes to kernel devices, run the following command:

```
sudo ls –l /dev/disk/by-path/ | grep <volume_name>
```

Note: The volume mapping is typically `/sda` or `/sdb` and is referred to as sdX throughout the remainder of this document.

Example

```
demo@demo:~# sudo ls -l /dev/disk/by-path/ | grep demo002
lrwxrwxrwx 1 root root  9  May 1 10:16 ip-172.27.21.52:3260-iscsi-iqn.2010-
01.com.solidfire:wupi.demo002.16-lun-0 -> ../../sdc
```

This example shows that the SolidFire volume '`demo002`' (volume ID 16) is mapped to kernel device `/dev/sdc`.

3. To see all SolidFire volume mappings on this system, run the following command:

```
sudo ls –l /dev/disk/by-path/ | grep solidfire
```

2.7 Scanning for FC Volumes

After you have added the host WWPNs to the SolidFire volume access group, the volumes in that volume access group immediately become visible to the host. For the host to recognize the volumes, you must first scan the SCSI bus for the new volumes.

Prerequisites

- At least one volume has been added to the configured volume access group.
- All host WWPNs have been zoned to the SolidFire cluster and added to the configured volume access group.

Procedure

Run the following commands for the OS you are running:

- Debian#:

```
sudo -s
echo "1" > /sys/class/fc_host/hostN/issue_lip
exit
```

- SUSE Linux Enterprise Server#:

```
sudo rescan-scsi-bus.sh
```

- Fedora#:

```
sudo -s
echo "1" > /sys/class/fc_host/hostN/issue_lip
exit
```

Note: For each hostN entry, the N must be replaced with the appropriate number (for example, host8).

2.8 Tune the Kernel for Optimal Performance

To obtain maximum performance from your SolidFire volumes, you can tune some Linux kernel parameters. Each parameter is described in the following procedure, along with its default setting. The parameters suggested here are offered as a starting point. NetApp recommends that you tune the parameters for your specific application to meet your needs.

Prerequisites

- Version 9.x or higher of the hdparm utility is required to set the queue depth.
- To check the hdparm version, run the following command: `sudo /sbin/hdparm -V`.
- On Red Hat Enterprise Linux 7+ distributions, hdparm is not installed by default. To install it, run the following command:

```
sudo yum install -y hdparm
```

Note: Instead of using hdparm, you can manipulate queue depth by managing the file `/sys/block/sdX/device/queue_depth`.

You can tune the SolidFire volumes with one of the following methods:

- udev Rules
- `/etc/rc.local`

Each method sets the tuning parameters, but the viability of each method depends on the operating system and OS version that you are running. You might want to try various methods to find the one that works best for your situation.

- **I/O schedulers.** Four types of I/O schedulers exist for each operating system in Linux:
  - **Noop.** Inserts all incoming I/O requests into a simple FIFO queue and implements request merging.
  - **Anticipatory.** Seeks to increase the efficiency of disk utilization by anticipating synchronous block requests and caching them for use.
  - **Deadline.** Guarantees the start service time for a request by imposing a deadline on all I/O operations to prevent starvation of requests for both reads and writes.
  - **Completely fair queuing (cfq).** Places synchronous requests submitted by processes into a number of per-process queues and then allocates time slices for each of the queues to access the disk. The length of the time slice and number of requests a queue is allowed to submit depend on the I/O priority of the given process.

- **Rotational.** Setting the rotational parameter to `0` indicates to the system (or to other applications that check) that the disk is a nonrotational disk or, in other words, a solid-state disk. (Default = 0 or 1.)
- **nr_requests.** The nr_requests parameter affects the number of read and write I/O requests that are queued by the kernel before an application is requested to sleep. Increasing this parameter allows the kernel to queue more I/O but can have a negative effect on system memory. If you see memory issues on your system, try lowering the nr_requests parameter to find a balance. (Default = 128.)
- **add_random.** The add_random parameter indicates whether the specified device should participate in adding randomness to the entropy pool. There is no need for SolidFire volumes to contribute to the system's entropy pool. Therefore, setting this parameter to `0` prevents the device from contributing to the entropy pool and can increase performance and decrease CPU load. (Default = 1.)
- **Queue depth.** The device queue depth defines the number of I/O requests that are queued for transmission to the physical device. An increased queue depth can allow for higher I/O throughput but should be carefully tied to your volume QoS setting for maximum performance. (Default varies.)
- **rq_affinity.** I/O completions can be processed on a different CPU from the one that issued the I/O. Setting rq_affinity to `1` causes the kernel to deliver completions to the CPU on which the I/O was issued. This can improve CPU data-caching effectiveness.
- **max_sectors_kb.** By default, the maximum request size sent to disk is 512KB. This tunable parameter can be used to either raise or lower that value. The minimum value is limited by the logical block size; the maximum value is limited by the max_hw_sectors_kb parameter. Some SSDs perform worse when I/O sizes exceed the internal erase block size. In such cases, NetApp recommends turning max_hw_sectors_kb down to the erase block size. You can test for this by using an I/O generator such as iozone or aio-stress and varying the record size from, for example, 512 bytes to 1MB.

You can read more about each of these parameters at this Queue sysfs Files page and at this Solid-State Storage Devices and the Block Layer page.

2.9 Checking Tuning Parameters

You can check the current tuning parameters for a volume by running the following commands:

Note: NetApp recommends that you check the parameters prior to tuning the volumes so that you can revert the settings back to defaults if needed.

Note: The `add_random` and `rq_affinity` parameters are not present in versions of Red Hat Enterprise Linux prior to 6.0.

Procedure

To check tuning parameters, run the following commands:

```
demo@demo:~# sudo cat /sys/block/sdX/queue/rq_affinity
demo@demo:~# sudo cat /sys/block/sdX/queue/max_sectors_kb
demo@demo:~# sudo cat /sys/block/sdX/queue/rotational
demo@demo:~# sudo cat /sys/block/sdX/queue/scheduler
demo@demo:~# sudo cat /sys/block/sdX/queue/nr_requests
demo@demo:~# sudo cat /sys/block/sdX/queue/add_random
demo@demo:~# sudo /sbin/hdparm -Q /dev/sdX
```

For distributions that do not support queue-depth arguments, you can view queue depth by using the following command:

```
cat  /sys/block/sdX/device/queue_depth
```

Method 1: udev Rules

This method allows you to control multiple SolidFire volumes by using one configuration file. This method also implements the `hdparm` settings for queue depth. For details, see the section “Volume I/O Queue Depth.”

Note: NetApp recommends using udev rules for the tuning method.

Note: This udev rules file does not work with versions of Red Hat Enterprise Linux prior to 6.0.

Procedure

1. Create the following file as root:

```
/etc/udev/rules.d/99-solidfire.rules
```

2. Edit the `/lib/udev/rules.d/99-solidfire.rules` file as root and add the following lines:

```
ACTION=="add|change", \
KERNEL=="sd[a-z]", \
ATTRS{vendor}=="SolidFir ", \
ATTR{queue/scheduler}="noop", \
ATTR{queue/add_random}="0", \
ATTR{queue/rq_affinity}="2", \
ATTR{queue/nr_requests}="1024", \
ATTR{queue/max_sectors_kb}="2048", \
RUN+="/sbin/hdparm -Q <Queue Depth> /dev/%k"
```

3. Trigger the udev rules so that the tunes are executed.

```
demo@demo:~# sudo udevadm trigger
```

For SUSE Linux Enterprise Server 11 SP4#, run the following command:

```
sudo /sbin/udevadm trigger
```

Note: Only perform this step when the rules are created or updated. Afterward, the rules are matched or executed when the OS boots or new devices are connected. For SLES 11 SP4, the udevadm command is sufficient for changing settings. Therefore, you do not need to log in to the volumes again.

4. Recheck the tuning parameters (see the section “Checking Tuning Parameters”). The values should be as follows:

```
rotational 0
scheduler [noop]
nr_requests 1024
add_random
0 max_sectors_kb 2048
dev/sdX queue_depth    = <Queue Depth given to hdparm>
```

Note: For systems that do not have a recent version of hdparm, or if you want to make the udev rules file cleaner, you can replace the RUN line with the following:

```
ATTR{device/queue_depth}="64".
```

Method 2: /etc/rc.local

This method changes the tuning parameters at the level of individual devices. You must add a line for each device.

Note: This method is not valid for systemd-based systems such as Red Hat Enterprise Linux 7+.

Note: Edit this file with the root user or a user with root permissions.

Procedure

1. Edit the following rc bootup file:
   – Debian/Fedora#:

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

   – SUSE Linux Enterprise Server#:

```
/etc/rc.d/boot.local
```

2. Add the following lines to the end of the file before the `'exit 0'` line:

```
echo 0 > /sys/block/sdX/queue/rotational
echo noop > /sys/block/sdX/queue/scheduler
echo 1024 > /sys/block/sdX/queue/nr_requests
echo 0 > /sys/block/sdX/queue/add_random
echo 2048 > /sys/block/sdX/queue/max_sectors_kb
/sbin/hdparm -Q <QD> /dev/sdX
```

   This can also be done with the following command:

```
echo <QD> /sys/block/sdX/device/queue_depth.
```

3. Reboot your system for the changes to take effect.
4. Recheck the tuning parameters (see the section “Checking Tuning Parameters”). The values should be:

```
rotational 0
scheduler [noop]
nr_requests 1024
add_random 0
max_sectors_kb 2048
dev/sdX queue_depth    = <Queue Depth given to hdparm>
```

3 Multipath I/O Configuration

Multipath I/O (MPIO or DM-MPIO) is a method of creating multiple data paths to a single iSCSI volume over a single physical link or multiple physical links (NICs or HBAs) connected to a single storage system. NetApp recommends MPIO for volumes that have (or might in the future have) large amounts of I/O or multiple paths for HA (such as FC).

This section details configuring MPIO for use with SolidFire volumes. For information about installing MPIO, see the section “Installing Multipath I/O Software.”

Note: For systems that use Red Hat Enterprise Linux, see the section “Multipath I/O Configuration.”

Note: Edit this file as the root user or a user with root permissions.

3.1 Procedure

1. Edit the multipath configuration file at `/etc/multipath.conf` to have the following contents:

```
# This is a basic configuration file with some examples, for device mapper
# multipath.

## Use user friendly names, instead of using WWIDs as names. defaults {
user_friendly_names yes
}
##
devices {
device {
vendor "SolidFir"
product "SSD  SAN"
path_grouping_policy multibus
path_selector "round-robin 0"
path_checker tur hardware_handler "0"
failback immediate
rr_weight uniform
rr_min_io 10
rr_min_io_rq 10
features "0"
no_path_retry 24
prio const
}
}
## Device black list
## Enter devices you do NOT want to be controlled by multipathd
## Example: internal drives

#blacklist {

#}
```

Notes:

- The default path_selector recommended for SolidFire series clusters is "`round-robin 0`." However, for FC installations, "`service-time 0`" yields better throughput and failover times.
- `rr_min_io` and `rr_min_io_rq` define the amount of I/O to send to each path before switching to the next. A value of 10 is recommended for both variables as a default for SolidFire series clusters. You might need to tune this value depending on your workload. For high-performance applications, NetApp recommends tuning these values and testing throughput to determine the best balance. As a best practice, both parameters should be set to the same value.

2. Use the following command to force MPIO to reload the configuration:

```
sudo multipath -r
```

For SUSE Linux Enterprise Server 11 SP4#, enter the following command:

```
sudo /sbin/multipath -r
```

3. Use the following command to verify that the changes have taken effect:

```
sudo multipath -ll
```

For SUSE Linux Enterprise Server 11 SP4#, enter the following command:

```
sudo /sbin/multipath -11
```

Note: In some cases, you might need to run "`multipath -r`" more than once for the changes to take effect.

4 Volume I/O Queue Depth

To optimize SolidFire QoS, you must update the volume queue depth to the appropriate value. If the queue depth is too high, then frames remain in the active queue for too long. However, if the queue depth is too low, then the volume cannot reach the desired performance levels.

Table 1 helps you determine the appropriate queue depth.

Table 1) Queue depth.

Minimum IOPS	Queue Depth
100–199	1
200–399	2

Minimum IOPS	Queue Depth
400–799	4
800–1,599	8
1,600–3,199	16
3,200–6,399	32
6,400+	64*

*For more information about using higher queue depth values, see the section “Connecting Multiple iSCSI Sessions to a Single Volume.”

In addition, certain hypervisors and HBAs might throttle queue depth, so see Configuring SolidFire Quality of Service or Defining SolidFire Quality of Service guides for additional details. If the hdparm command cannot complete, then check and update the settings on these devices accordingly.

Note: The queue depth settings listed are suggestions only. They should be used as a starting point for tuning your OS and application performance.

4.1 Updating the Queue Depth

This section explains how to change the queue depth on a specific device with the hdparm tool. You should verify the default queue depth prior to changing it in case you want to revert back.

Note: You can skip this section if you used the udev rules method for tuning your device parameters (see the section “Method 1: udev Rules”). If you find you need to tune the queue depth to a different value, make the change to the udev rule and retrigger for the change to take effect.

Prerequisites

- hdparm version 9.x or higher is required to set the queue depth.

  If you don't meet this requirement, you can use the node at `/sys/block/sdX/device/queue_depth` to read and set the queue depth values. The hdparm tool is not strictly needed.

Procedure

1. To check the current queue depth to a volume, run the following command:

```
sudo /sbin/hdparm -Q /dev/sdX
```

2. When you have a corresponding queue depth value for your volume, run the following command:

```
sudo /sbin/hdparm -Q <Queue Depth> /dev/sdX
```

5 Creating the Partitions

You can create a partition with two methods: fdisk or GNU Parted.

Note: The following method is not valid for creating partitions or file systems on versions of Red Hat Enterprise Linux prior to 6.0.

Note: In a multipath configuration, use the device located in `/dev/mapper` for the following commands. Typically, the devices are named `/dev/dm-x` and are linked to from the devices in `/dev/mapper`.

- The fdisk utility is used to format or partition any volume less than 2TB in size. It does not understand GUID partition tables (GPTs), and it is not designed for large partitions.
- The GNU Parted utility is used to format or partition any volume greater than 2TB in size or any smaller volumes that you want to partition with the GPT. NetApp recommends GNU Parted because it allows for more flexibility, supports both GPT and master boot record (MBR) partition tables, and allows formatting of any partition size.

5.1 Method 1: fdisk

This method creates a partition by using the fdisk command. For volumes that use 512-byte emulation (512e), consult the Configuring Partition Alignment for SolidFire Storage guide at the NetApp Support site.

Note: The fdisk utility cannot create partitions greater than 2TB or utilize the GPT. To create larger partitions or to use the GPT, see the section “Method 2: GNU Parted.”

Prerequisites

- You must be connected (logged in) to SolidFire volumes.

Procedure

Note: This example uses a 10GB volume.

1. Run the following command and respond to the prompts issued:

```
sudo /sbin/fdisk -c -u </dev/sdX>
```

   Note: If the previous multipath recommendations are followed, the device node is not `/dev/sdX`, but rather `dev/dm-#`. You can find the specific device name you need by using the `sudo /sbin/multipath -l` command. This command lists the `/dev/mapper/mpathX` device in use, and you can find the specific `/dev/dm-X` device you need for the fdisk command by examining the `/dev/mapper/mpathX` soft link target.

2. Command (m for help):

   <response> `n`

3. Partition type:

   `p   primary (0 primary, 0 extended, 4 free)`

   `e   extended`

   <response> `p`

4. Partition number (1–4, default 1):

   <response> `1`

5. First sector (256–2441471, default 256).

   <response> `256`

6. Last sector, +sectors or +size {K,M,G} (256–2441471, default 2441471):

   <response> `2441471`

   Note: For volumes of different sizes, the last sector number changes.

7. Command (m for help):

   <response> `p`

```
Disk /dev/sdc: 10.0 GB, 10000269312 bytes
```

```
64 heads, 32 sectors/track, 1192 cylinders, total 2441472 sectors
Units = sectors of 1 * 4096 = 4096 bytes
Sector size (logical/physical): 4096 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disk identifier: 0x73d25965

Device Boot    Start      End        Blocks   Id    System
/dev/sdc1           256  2441471       9764864     83  Linux
```

8. Command (m for help):

   <response> `w`

   Output response from the system:

```
The partition  table  has been altered!
Calling  ioctl()  to  re-read  partition  table.
Syncing  disks.
```

Note: For the remainder of this document, the newly created partitions are referred to as `/dev/sdX#`: for example, `/dev/sdb1`. For multipath volumes, these are the next available numbers after the "`dm`." For example, a disk called `dev/dm-0` has its first partition on `/dev/dm-1`.

5.2 Method 2: GNU Parted

This method creates a partition with the GNU Parted command, which you can use to partition any volume.

Prerequisites

You must be connected (logged in) to SolidFire volumes.

Procedure

Note: This example uses a 10GB volume.

Run the following command and respond to the prompts issued:

```
sudo parted -a optimal </dev/sdX>
```

For SUSE Linux Enterprise Server 11 SP4#, run the following command:

```
sudo /usr/sbin/parted -a optimal </dev/sdX>
```

For multipath SUSE Linux Enterprise Server 11 SP4#, run the following command:

```
sudo /usr/sbin/parted -a optimal </dev/dm-#>
```

Note: Using the `–a optimal` option helps to properly align sector boundaries for 512e and non-512e volumes. This helps maximize volume performance.

a. (parted):

   <response> `mklabel gpt`

b. (parted):

   <response> `print`

```
Model: SolidFir SSD SAN (scsi)
Disk /dev/sdb: 10.0GB
Sector size (logical/physical): 512B/4096B
Partition Table: gpt

Number  Start  End  Size  File system  Name  Flags
```

c. (parted):

   <response> `mkpart`

d. Partition name? [ ]?

<response> `Part1`

e. File system type? [ext2]?

<response> `ext4`

f. Start?

<response> `0GB`

g. End?

<response> `10GB`

h. (parted):

<response> `print`

```
Model: SolidFir SSD SAN (scsi)
Disk /dev/sdb: 10.0GB
Sector size (logical/physical): 512B/4096B
Partition Table: gpt

Number  Start   End     Size   File system  Name   Flags
1      1049kB  9999MB  9998MB              Part1
```

i. (parted):

<response> `quit`

5.3 GNU Parted Examples

Here are two slightly more complex large-volume examples, both of which use an 8TB volume. The `mkpart` subcommand is as follows:

```
mkpart <optional filesystem type> <partition name> <start> <end>
```

The `<start>` and `<end>` offsets are from the beginning of the disk and can be any of the following or a combination:

- Percentage (14%)
- Size (10MB, 47GiB, 1.27TB, and so on)
- Sectors (40s)

You can also use negative numbers to denote the offset from the end of the disk. For example, `mkpart endingpart -500GB 100%` creates a partition on the last 500GB of the volume.

Note: You might see this notice after running parted: “Information: You might need to update /etc/fstab.” Updates to `/etc/fstab` are necessary for volumes to be automatically mounted on a system reboot. See the section “Mounting the File System” for details.

Example 1

Carve one 8TB volume into four 2TB partitions. User input is noted in red.

```
sudo parted -a optimal /dev/sdb
GNU Parted 2.3
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mklabel gpt
(parted) print
Model: SolidFir SSD SAN (scsi) Disk /dev/sdb: 8000GB
Sector size (logical/physical): 4096B/4096B Partition Table: gpt

Number  Start  End  Size  File system  Name  Flags

(parted) mkpart P1 0% 25%
(parted) mkpart P2 25% 50%
```

```
(parted) mkpart P3 50% 75%
(parted) mkpart P4 75% 100%
(parted) print

Model: SolidFir SSD SAN (scsi)
Disk /dev/sdb: 8000GB
Sector size (logical/physical): 4096B/4096B
Partition Table: gpt

Number  Start   End     Size     File system  Name   Flags
1      1049kB  2000GB  2000GB                P1
2      2000GB  4000GB  2000GB                P2
3      4000GB  6000GB  2000GB                P3
4      6000GB  8000GB  2000GB                P4

(parted) quit
Information: You may need to update /etc/fstab.
```

Note: For SUSE Linux Enterprise Server 11 SP4, use the command `sudo /usr/sbin/parted -a optimal /dev/sdb`.

Example 2

Carve one 8TB volume into four mixed-size partitions by using a variety of start and end values. User input is noted in red.

```
sudo parted -a optimal /dev/sdb
GNU Parted 2.3
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mklabel gpt
(parted) print
Model: SolidFir SSD SAN (scsi)
Disk /dev/sdb: 8000GB
Sector size (logical/physical): 4096B/4096B
Partition Table: gpt

Number Start End Size File system Name Flags

(parted) mkpart Part1 0TB 2TB
(parted) mkpart Part2 2000GB 2000500MB
(parted) mkpart Part3 2000.5GB -500GB
(parted) mkpart Part4 7500GB 100%
(parted) print
Model: SolidFir SSD SAN (scsi) Disk /dev/sdb: 8000GB
Sector size (logical/physical): 4096B/4096B Partition Table: gpt

Number  Start   End     Size   File system  Name  Flags
1      1049kB  2000GB  2000GB              Part1
2      2000GB  2000GB  499MB               Part2
3      2000GB  7500GB  5500GB              Part3
4      7500GB  8000GB  500GB               Part4

(parted) quit
Information: You may need to update /etc/fstab.
```

6 Checking for 512-Byte Offset Misalignment

This procedure only applies to volumes that have 512-byte emulation enabled.

6.1 Primary Procedure

1. To determine if a volume has 512-byte emulation enabled, run the following command:

```
sudo cat /sys/block/sdX/queue/logical_block_size
```

Volumes with 512-byte emulation enabled show a logical block size of 512. Volumes with 512-byte emulation disabled show a logical block size equal to the physical block size (4096).

2. To check the offset of the volume's partitions, run the following command:

```
sudo parted /dev/sdX unit s print
```

For each partition listed, check that the starting sector (start column) is divisible by eight. If the starting sector is not divisible by eight, the volume might have misalignment issues. In this case, refer to the Configuring Partition Alignment for SolidFire Storage guide on the SolidFire Support page at https://mysupport.netapp. This guide assists with correcting unalignment issues.

6.2 Secondary Procedure

To determine if the SolidFire system is receiving misaligned I/O on a specific volume, run the `GetVolumeStats` API command from a web browser:

```
https://<SolidFire MVIP>/json-rpc/6.0?method=GetVolumeStats&volumeID=<SolidFire Volume ID>
```

If the value for `unalignedReads` or `unalignedWrites` increases between subsequent runs of the API command, then your volume might be unaligned.

Example: Aligned Volume

```
demo@demo:~# sudo parted /dev/sdb unit s print
Model: SolidFir SSD SAN (scsi) Disk /dev/sdb: 19531776s
Sector size (logical/physical): 512B/4096B Partition Table: gpt

Number  Start  End        Size       File system  Name   Flags
1       2048s  19529727s  19527680s               Part1
```

Example: Misaligned Volume

```
demo@demo:~# sudo parted /dev/sdd  unit s print
Model: SolidFir SSD SAN (scsi) Disk  /dev/sdd: 19531776s
Sector size (logical/physical):512B/4096B
Partition Table: msdos

Number  Start  End        Size       Type     File system  Flags
1       2052s  19531775s  19529724s  primary
```

Note: The value 2052 is not evenly divisible by 8 (2052/8 = 256.5). Therefore, this volume is likely to show performance issues because of misalignment.

7 Creating the File System

For Linux-based systems, NetApp recommends using the xfs and ext4 file systems to obtain optimal performance with SolidFire volumes. To utilize the xfs file system, you must have the xfsprogs packages installed.

Prerequisites

You need connected (logged in) and partitioned SolidFire volumes.

Procedure

1. Install the xfsprogs packages by running the following command:
   - Debian#:

```
sudo apt-get install -y xfsprogs
```

   - SUSE Linux Enterprise Server#:

```
sudo zypper install -y xfsprogs
```

– Fedora#:

```
sudo yum install -y xfsprogs
```

2. Build the file system by running the following command:

– xfs#:

```
sudo mkfs.xfs -K </dev/sdX#>
```

– ext4#:

```
sudo mkfs.ext4 -E nodiscard </dev/sdX#>
```

– SUSE Linux Enterprise Server 11 SP4# (and other Red Hat–based distributions):

```
sudo /sbin/mkfs.XXXX
```

Note: SUSE Linux Enterprise Server 11 SP4 does not support the `-E` nodiscard option.

Note: The options selected (`-K` and `-E` nodiscard) significantly decrease the provisioning time.

8 Mounting the File System

You need to mount each partition after you have created partitions on the volume. You can then add mount statements to the `/etc/fstab` file to make the mounts persistent across reboots. This section describes the detailed process of mounting the file systems and configuring the `/etc/fstab` file so that the same device is always mounted in the same location. The device is also mounted after the network services have started. See the section “Mount Time Options” for more details.

Prerequisites

The path to mount (`/path/to/mount`) must already be created before you mount your partitions.

Procedure

1. To mount each partition, run the following command:

```
sudo mount </dev/sdX1> </path/to/mount>
```

Example

To mount `/dev/sdb1` to `/mnt/disk1`, run the following commands:

```
sudo mkdir /mnt/disk1
sudo mount -o noatime,discard,nobarrier /dev/sdb1 /mnt/disk1
```

If the "`/mnt`" directory does not exist, run the command `sudo mkdir -p /mnt/disk1`.

2. To list the currently mounted file systems, run the following command:

```
mount
```

3. Add the newly mounted file systems to the `/etc/fstab` file to make sure that they persist across reboots.

Note: During reboots, the device name (such as `/dev/sdb`) associated with each iSCSI volume might change. Therefore, your entries in `/etc/fstab` must reference the drive's universally unique identifier (UUID) rather than its device name.

a. Mount your device at the appropriate mount point.

b. Determine the UUID that corresponds to your device name.

c. Find your device's entry in `/etc/mtab`.

d. Replace the device name with the UUID and add the line to `/etc/fstab`.

e. Add _netdev,noatime,discard,nobarrier to the file system fstab rule.

Note: This tells the operating system to wait until the network services start fully before attempting to mount the file system. Without this, the mount operation can fail if the network is slow to start.

f. Verify that the /etc/fstab entry is correct.

Example

User input is in red.

```
a.   demo@demo:~# sudo mount -o nobarrier,noatime,discard /dev/sdb1
     /mnt/disk1
b.   demo@demo:~# sudo blkid /dev/sdb1
     For SUSE Linux Enterprise Server 11 SP4 and others sudo /sbin/blkid
     /dev/sdb1
/dev/sdb1: UUID="8163881e-7a0d-4d18-a043-c3253800a66a" TYPE="ext4"
c.   demo@demo:~# sudo cat /etc/mtab
     /dev/sda1 / ext4 rw,errors=remount-ro 0 0 proc /proc proc rw,noexec,nosuid,nodev 0 0
       <snip>
     /dev/sdb1 /mnt/disk1 ext4 rw,nobarrier 0 0
d.   demo@demo:~# sudo echo "UUID=8163881e-7a0d-4d18-a043-c3253800a66a
     /mnt/disk1 ext4 rw,_netdev,nobarrier,noatime,discard 0 0" >>
     /etc/fstab
e.   demo@demo:~# sudo cat /etc/fstab
     # /etc/fstab: static file system information.
     #
     # Use 'blkid' to print the universally unique identifier for a
     # device; this may be used with UUID= as a more robust way to name devices
     # that works even if disks are added and removed. See fstab(5).
     #
     # <file system> <mount point>    <type>    <options>    <dump>    <pass>
     UUID=8163881e-7a0d- 4d18-a043-c3253800a66a /mnt/disk1 ext4 rw,_netdev 0 0
```

8.1 Mount Time Options

Mount ext4 and xfs file systems with the noatime, nobarrier, and discard options. These options are described in the following list:

- **noatime.** Do not update node access times on this file system. This improves performance in most workloads (for example, a news spool on a news server).
- **discard.** The discard function issues frequent commands to let the block device reclaim space freed by the file system. This is useful for SSD devices, thinly provisioned LUNs, and virtual machine images, but might have a significant performance impact. The fstrim command is also available to initiate batch trims from userspace.
- **nobarrier.** Write barriers allow certain I/O operations to make it through the device cache to persistent storage. If disabled on a device with a volatile (non-battery-backed) writeback cache, the nobarrier option can lead to file system corruption in the event of a system crash or power loss.

9 Translating SCSI IDs into SolidFire Volume IDs

For iSCSI-connected SolidFire volumes, you can easily validate the cluster identifier and volume ID by reading the iSCSI IQN of the volume. However, for FC volumes, you must decode that information from the SCSI IDs. The following procedure provides examples of how to convert the SCSI ID into the SolidFire cluster unique identifier as well as the volume ID.

Note: This procedure is not valid for versions of Red Hat Enterprise Linux prior to 6.0.

Procedure

1. Determine the SCSI ID of the Linux device.

```
/lib/udev/scsi_id -g (For Debian & Red Hat 7+)
```

Example

```
sudo /lib/udev/scsi_id -g /dev/sdf
36f47acc10000000062346665000000d1
```

2. The last eight digits of the SCSI ID are the volume ID in hexadecimal notation. For example:

```
0x000000d1 = 209
```

Thus, the SolidFire volume ID is 209.

The following example combines this into a single command. Note the different quotation marks (single, double, and backtick) when copying and pasting:

```
printf "%d\n" '0x'`scsi_id -g <device> | awk '{ print substr($1,26,8) }'`
```

Example

```
[root@test ~]# printf "%d\n" '0x'`/lib/udev/scsi_id -g /dev/sdf | awk '{ print substr($1,26,8)
}'`
209
```

Note: On all Linux-based systems, the scsi_id tool is located at "`/lib/udev/scsi_id`."

3. The eight digits preceding the volume ID are the ASCII characters of the cluster unique identifier in hexadecimal notation.

```
36f47acc10000000062346665000000d1
```

In this case, 62 34 66 65 converts to b4fe.

The following example combines this into a single command:

```
printf '0x'`scsi_id -g /dev/sdf | awk '{ print substr($1,18,8) }'` | xxd -r
```

Example

```
[root@test ~]# printf '0x'`scsi_id -g /dev/sdf | awk '{ print substr($1,18,8) }'`
| xxd -r
b4fe
```

Appendix A: Multipath I/O Configuration: OS Variants

Depending on your operating system, the configuration procedure detailed in the section "Multipath I/O Configuration" might vary. This section describes the procedures for different variants of Red Hat Enterprise Linux.

Procedure: Red Hat Enterprise Linux 7.x

1. Edit the multipath configuration file at `/etc/multipath.conf` to contain the following text:

```
# This is a basic configuration file with some examples, for device mapper
# multipath.

## Use user friendly names, instead of using WWIDs as names. defaults {
user_friendly_names yes
}
##
devices {
device {
vendor "SolidFir" product "SSD SAN"
path_grouping_policy multibus
path_checker tur hardware_handler "0"
failback immediate
rr_weight uniform
rr_min_io 10
rr_min_io_rq 10
features "0"
no_path_retry 24 prio const
}

}

## Device black list
## Enter devices you do NOT want to be controlled by multipathd
## Example: internal drives

#blacklist {

#}
```

Note: `rr_min_io` and `rr_min_io_rq` define the amount of I/O to send to each path before switching to the next. A value of 10 is recommended for both variables as a default for SolidFire series clusters. You might need to tune this value depending on your workload. For high-performance applications, NetApp recommends tuning these values and testing throughput to determine the best balance. As a best practice, both parameters should be set to the same value.

2. Force MPIO to reload the configuration.

```
sudo multipath -r
```

3. Verify that the changes have taken effect.

```
sudo multipath -ll
```

Note: In some cases, you might need to run “`multipath -r`” more than once for the changes to take effect.

Procedure: Red Hat Enterprise Linux 6.x

1. Edit the multipath configuration file at `/etc/multipath.conf` to contain the following text:

```
#  This is a basic configuration file with some examples, for device mapper
# multipath.

## Use user friendly names, instead of using WWIDs as names. defaults {
user_friendly_names yes
}
##
devices {
device {
vendor "SolidFir" product "SSD SAN"
path_grouping_policy multibus
path_selector "round-robin 0"
path_checker tur hardware_handler "0"
failback immediate
rr_weight uniform
rr_min_io 10
features "0"
no_path_retry 24
}

}

## Blacklist all devices by default. Remove this to enable multipathing on the default devices.

#blacklist { # devnode "*" #}
```

Notes:

  - The default `path_selector` recommended for SolidFire storage is "`round-robin 0`." However, for FC installations, "`service-time 0`" has yielded better throughput and failover times.
  - `rr_min_io` defines the amount of I/O to send to each path before switching to the next. A value of 10 is recommended for both values as a default for SolidFire series clusters. You might need to tune this value depending on your workload. For high-performance applications, NetApp recommends tuning these values and testing throughput to determine the best balance. As a best practice, both parameters should be set to the same value.

2. Force MPIO to reload the configuration.

```
sudo multipath -r
```

3. Verify that the changes have taken effect.

```
sudo multipath -ll
```

Note: In some cases, you might need to run "`multipath -r`" more than once for the changes to take effect.

Procedure: Red Hat Enterprise Linux 5.11

1. Edit the multipath configuration file at `/etc/multipath.conf` to contain the following text:

```
# This is a basic configuration file with some examples, for device mapper
# multipath.
## Use user friendly names, instead of using WWIDs as names. defaults {
user_friendly_names yes
}
##
devices {
device {
vendor "SolidFir" product "SSD SAN" path_grouping_policy multibus
getuid_callout "/lib/udev/scsi_id-g -u -d /dev/%n" path_selector "round-robin 0"
path_checker tur hardware_handler "0" failback immediate rr_weight uniform rr_min_io 10
features "0" no_path_retry 24
}

}

## Blacklist all devices by default. Remove this to enable multipathing on the default devices.

#blacklist { # devnode "*" #}
```

Notes:

- The default `path_selector` recommended for SolidFire storage is "`round-robin 0`." However, for FC installations, "`service-time 0`" has yielded better throughput and failover times.
- `rr_min_io` defines the amount of I/O to send to each path before switching to the next. A value of 10 is recommended for both values as a default for SolidFire series clusters. You might need to tune this value depending on your workload. For high-performance applications, NetApp recommends tuning these values and testing throughput to determine the best balance. As a best practice, both parameters should be set to the same value.

2. Force MPIO to reload the configuration.

```
sudo multipath  -r
```

3. Verify that the changes have taken effect.

```
sudo multipath -ll
```

Note: In some cases, you might need to run "`multipath -r`" more than once for the changes to take effect.

Appendix B: iSCSI Timeouts, iscsid.conf

NetApp recommends using the default open iSCSI timeout values unless you are using a database. In that case, see the best practices guide specific to the database that you are using. In some cases, you must modify the open iSCSI default values to avoid connection errors in the event of a SolidFire node failure.

If you are using a database, see our best practices guides for the database you are using. Depending on your operating system, your default iSCSI values might vary. NetApp recommends tuning the iSCSI timeout values to avoid any connection errors in the event of a SolidFire node failure.

Procedure: Linux

1. Edit the `iscsi.conf` configuration file at `/etc/iscsi/iscsid.conf` to add to or modify the following contents:

```
node.session.timeo.replacement_timeout = 120
node.conn[0].timeo.login_timeout = 15
node.conn[0].timeo.logout_timeout = 15
node.conn[0].timeo.noop_out_timeout = 5
node.session.err_timeo.abort_timeout = 15
node.session.err_timeo.lu_reset_timeout = 20
```

2. Restart open iSCSI services or reboot for changes to take effect.

Where to Find Additional Information

To learn more about the information described in this document, refer to the following websites:

- NetApp Support
  https://mysupport.netapp.com
- NetApp SolidFire Resources
  https://mysupport.netapp.com/info/web/ECMLP2740378.html

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

Copyright Information

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

Trademark Information

TR-4639-1017