Storage Protocols

Table of contents

There are several storage protocols that use the advantage of InfiniBand and RDMA for performance reasons (high throughput, low latency and low CPU utilization). In this chapter we will discuss the following protocols:

- **SCSI RDMA Protocol (SRP)** is designed to take full advantage of the protocol off-load and RDMA features provided by the InfiniBand architecture.
- **iSCSI Extensions for RDMA (iSER)** is an extension of the data transfer model of iSCSI, a storage networking standard for TCP/IP. It uses the iSCSI components while taking the advantage of the RDMA protocol suite. ISER is implemented on various stor- age targets such as TGT, LIO, SCST and out of scope of this manual.

  For various ISER targets configuration steps, troubleshooting and debugging, as well as other implementation of storage protocols over RDMA (such as Ceph over RDMA, nbdX and more) refer to Storage Solutions on the Community website.
- **Lustre** is an open-source, parallel distributed file system, generally used for large-scale cluster computing that supports many requirements of leadership class HPC simulation environments.
- **NVM Express™ over Fabrics (NVME-oF)**
  - NVME-oF is a technology specification for networking storage designed to enable NVMe message-based commands to transfer data between a host computer and a target solid-state storage device or system over a network such as Ethernet, Fibre Channel, and InfiniBand. Tunneling NVMe commands through an RDMA fabric provides a high throughput and a low latency. This is an alternative to the SCSi based storage networking protocols.
  - NVME-oF Target Offload is an implementation of the new NVME-oF standard Target (server) side in hardware. Starting from ConnectX-5 family cards, all regular IO requests can be processed by the HCA, with the HCA sending IO requests directly to a real NVMe PCI device, using peer-to-peer PCI communications. This means that excluding connection management and error flows, no CPU utilization will be observed during NVME-oF traffic.

    For further information, please refer to Storage Solutions on the Community website (enterprise-support.nvidia.com/s/).

SRP - SCSI RDMA Protocol

The SCSI RDMA Protocol (SRP) is designed to take full advantage of the protocol offload and RDMA features provided by the InfiniBand architecture. SRP allows a large body of SCSI software to be readily used on InfiniBand architecture. The SRP Initiator controls the connection to an SRP Target in order to provide access to remote storage devices across an InfiniBand fabric. The kSRP Target resides in an IO unit and provides storage services.

SRP Initiator

This SRP Initiator is based on open source from OpenFabrics (www.openfabrics.org) that implements the SCSI RDMA Protocol-2 (SRP-2). SRP-2 is described in Document # T10/1524-D available from http://www.t10.org.

The SRP Initiator supports

- Basic SCSI Primary Commands -3 (SPC-3)

  (www.t10.org/ftp/t10/drafts/spc3/spc3r21b.pdf)

- Basic SCSI Block Commands -2 (SBC-2)

  (www.t10.org/ftp/t10/drafts/sbc2/sbc2r16.pdf)

- Basic functionality, task management and limited error handling

ⓘ **Note**

This package, however, does not include an SRP Target.

Loading SRP Initiator

➢ ***To load the SRP module either:***

- Execute the `modprobe ib_srp` command after the OFED driver is up.

or

1. Change the value of `SRP_LOAD` in `/etc/infiniband/openib.conf` to “`yes`”.
2. Run `/etc/init.d/openibd restart` for the changes to take effect.

> ⓘ **Note**
>
> When loading the `ib_srp` module, it is possible to set the module parameter `srp_sg_tablesize.` This is the maximum number of gather/scatter entries per I/O (default: 12).

SRP Module Parameters

When loading the SRP module, the following parameters can be set (viewable by the "modinfo ib_srp" command):

`cmd_sg_entries`	Default number of gather/scatter entries in the SRP command (default is 12, max 255)
`allow_ext_sg`	Default behavior when there are more than cmd_sg_entries S/G entries after mapping; fails the request when false (default false)
`topspin_workarounds`	Enable workarounds for Topspin/Cisco SRP target bugs
`reconnect_delay`	Time between successive reconnect attempts. Time between successive reconnect attempts of SRP initiator to a disconnected target until dev_loss_tmo timer expires (if enabled), after that the SCSI target will be removed

`fast_io_fail_tmo`	Number of seconds between the observation of a transport layer error and failing all I/O. Increasing this timeout allows more tolerance to transport errors, however, doing so increases the total failover time in case of serious transport failure. Note: fast_io_fail_tmo value must be smaller than the value of reconnect_delay
`dev_loss_tmo`	Maximum number of seconds that the SRP transport should insulate transport layer errors. After this time has been exceeded the SCSI target is removed. Normally it is advised to set this to -1 (disabled) which will never remove the scsi_host. In deployments where different SRP targets are connected and disconnected frequently, it may be required to enable this timeout in order to clean old scsi_hosts representing targets that no longer exists

Constraints between parameters:

- dev_loss_tmo, fast_io_fail_tmo, reconnect_delay cannot be all disabled or negative values.
- reconnect_delay must be positive number.
- fast_io_fail_tmo must be smaller than SCSI block device timeout.
- fast_io_fail_tmo must be smaller than dev_loss_tmo.

SRP Remote Ports Parameters

Several SRP remote ports parameters are modifiable online on existing connection.

➢ ***To modify dev_loss_tmo to 600 seconds:***

```
echo 600 > /sys/class/srp_remote_ports/port-xxx/dev_loss_tmo
```

➢ ***To modify fast_io_fail_tmo to 15 seconds:***

```
echo 15 > /sys/class/srp_remote_ports/port-xxx/fast_io_fail_tmo
```

➤ ***To modify reconnect_delay to 10 seconds:***

```
echo 20 > /sys/class/srp_remote_ports/port-xxx/reconnect_delay
```

Manually Establishing an SRP Connection

The following steps describe how to manually load an SRP connection between the Initiator and an SRP Target. “Automatic Discovery and Connection to Targets” section explains how to do this automatically.

- Make sure that the ib_srp module is loaded, the SRP Initiator is reachable by the SRP Target, and that an SM is running.
- To establish a connection with an SRP Target and create an SRP (SCSI) device for that target under /dev, use the following command:

  ```
  echo -n id_ext=[GUID value],ioc_guid=[GUID value],dgid=[port GID value],\
  pkey=ffff,service_id=[service[0] value] > \
  /sys/class/infiniband_srp/srp-mlx[hca number]-[port number]/add_target
  ```

See **“**SRP Tools - ibsrpdm, srp_daemon and srpd Service Script**”** section for instructions on how the parameters in this echo command may be obtained.

Notes:

- Execution of the above “echo” command may take some time
- The SM must be running while the command executes

- It is possible to include additional parameters in the echo command:
    - max_cmd_per_lun - Default: 62
    - max_sect (short for max_sectors) - sets the request size of a command
    - io_class - Default: 0x100 as in rev 16A of the specification (In rev 10 the default was 0xff00)
    - tl_retry_count - a number in the range 2..7 specifying the IB RC retry count. Default: 2
    - comp_vector, a number in the range 0..n-1 specifying the MSI-X completion vector. Some HCA's allocate multiple (n) MSI-X vectors per HCA port. If the IRQ affinity masks of these interrupts have been configured such that each MSI-X interrupt is handled by a different CPU then the comp_vector parameter can be used to spread the SRP completion workload over multiple CPU's.
    - cmd_sg_entries, a number in the range 1..255 that specifies the maximum number of data buffer descriptors stored in the SRP_CMD information unit itself. With allow_ext_sg=0 the parameter cmd_sg_entries defines the maximum S/G list length for a single SRP_CMD, and commands whose S/G list length exceeds this limit after S/G list collapsing will fail.
    - initiator_ext - see "Multiple Connections from Initiator InfiniBand Port to the Target" section.
- To list the new SCSI devices that have been added by the echo command, you may use either of the following two methods:
    - Execute “fdisk -l”. This command lists all devices; the new devices are included in this listing.
    - Execute “dmesg” or look at /var/log/messages to find messages with the names of the new devices.

SRP sysfs Parameters

Interface for making ib_srp connect to a new target. One can request ib_srp to connect to a new target by writing a comma-separated list of login parameters to this sysfs attribute. The supported parameters are:

id_ext	A 16-digit hexadecimal number specifying the eight byte identifier extension in the 16-byte SRP target port identifier. The target port identifier is sent by ib_srp to the target in the SRP_LOGIN_REQ request.
ioc_guid	A 16-digit hexadecimal number specifying the eight byte I/O controller GUID portion of the 16-byte target port identifier.
dgid	A 32-digit hexadecimal number specifying the destination GID.
pkey	A four-digit hexadecimal number specifying the InfiniBand partition key.
service_id	A 16-digit hexadecimal number specifying the InfiniBand service ID used to establish communication with the SRP target. How to find out the value of the service ID is specified in the documentation of the SRP target.
max_sect	A decimal number specifying the maximum number of 512-byte sectors to be transferred via a single SCSI command.
max_cmd_per_lun	A decimal number specifying the maximum number of outstanding commands for a single LUN.
io_class	A hexadecimal number specifying the SRP I/O class. Must be either 0xff00 (rev 10) or 0x0100 (rev 16a). The I/O class defines the format of the SRP initiator and target port identifiers.
initiator_ext	A 16-digit hexadecimal number specifying the identifier extension portion of the SRP initiator port identifier. This data is sent by the initiator to the target in the SRP_LOGIN_REQ request.
cmd_sg_entries	A number in the range 1..255 that specifies the maximum number of data buffer descriptors stored in the SRP_CMD information unit itself. With allow_ext_sg=0 the parameter cmd_sg_entries defines the maxi- mum S/G list length for a single SRP_CMD, and commands whose S/G list length exceeds this limit after S/G list collapsing will fail.
allow_ext_sg	Whether ib_srp is allowed to include a partial memory descriptor list in an SRP_CMD instead of the entire list. If a partial memory descriptor list has been included in an SRP_CMD the remaining memory descriptors are communicated from initiator to target via an additional RDMA transfer. Setting allow_ext_sg to 1 increases the maximum amount of data that can be transferred between initiator and target via a single SCSI command. Since not all SRP target implementations support partial memory descriptor lists the default value for this option is 0.
sg_tablesize	A number in the range 1..2048 specifying the maximum S/G list length the SCSI layer is allowed to pass to ib_srp. Specifying a value that exceeds cmd_sg_entries is only safe with partial memory descriptor list support enabled (allow_ext_sg=1).

comp_vector	A number in the range 0..n-1 specifying the MSI-X completion vector. Some HCA's allocate multiple (n) MSI-X vectors per HCA port. If the IRQ affinity masks of these interrupts have been configured such that each MSI-X interrupt is handled by a different CPU then the comp_vector parameter can be used to spread the SRP completion workload over multiple CPU's.
tl_retry_count	A number in the range 2..7 specifying the IB RC retry count.

SRP Tools - ibsrpdm, srp_daemon and srpd Service Script

The OFED distribution provides two utilities: ibsrpdm and srp_daemon:

- They detect targets on the fabric reachable by the Initiator (Step 1)
- Output target attributes in a format suitable for use in the above “echo” command (Step 2)
- A service script srpd which may be started at stack startup

The utilities can be found under /usr/sbin/, and are part of the srptools RPM that may be installed using the OFED installation. Detailed information regarding the various options for these utilities are provided by their man pages.

Below, several usage scenarios for these utilities are presented.

ibsrpdm

ibsrpdm has the following tasks:

1. Detecting reachable targets.
    1. To detect all targets reachable by the SRP initiator via the default umad device (/sys/class/infiniband_mad/umad0), execute the following command:

       ```
       ibsrpdm
       ```

       This command will result into readable output information on each SRP Target detected. Sample:

```
     IO Unit Info:
          port LID:        0103
          port GID:
fe800000000000000002c90200402bd5
          change ID:       0002
          max controllers: 0x10
       controller[  1]
           GUID:      0002c90200402bd4
           vendor ID: 0002c9
           device ID: 005a44
           IO class : 0100
           ID:        LSI Storage Systems SRP Driver
200400a0b81146a1
           service entries: 1
           service[  0]: 200400a0b81146a1 /
SRP.T10:200400A0B81146A1
```

2. To detect all the SRP Targets reachable by the SRP Initiator via another umad device, use the following command:

```
ibsrpdm -d <umad device>
```

2. Assisting in SRP connection creation.

   1. To generate an output suitable for utilization in the “echo” command in “Manually Establishing an SRP Connection” section, add the ‘-c’ option to ibsrpdm:

   ```
   ibsrpdm -c
   ```

   Sample output:

```
id_ext=200400A0B81146A1,ioc_guid=0002c90200402bd4,
dgid=fe800000000000000002c90200402bd5,pkey=ffff,service_id
```

2. To establish a connection with an SRP Target using the output from the 'ibsrpdm -c' example above, execute the following command:

   ```
   echo -n
   id_ext=200400A0B81146A1,ioc_guid=0002c90200402bd4,
   dgid=fe800000000000000002c90200402bd5,pkey=ffff,service_id
   > /sys/
   class/infiniband_srp/srp-mlx5_0-1/add_target
   ```

   The SRP connection should now be up; the newly created SCSI devices should appear in the listing obtained from the '`fdisk -l`' command.

3. Discover reachable SRP Targets given an InfiniBand HCA name and port, rather than by just running `/sys/class/infiniband_mad/umad<N>` where `<N>` is a digit.

srpd

The srpd service script allows automatic activation and termination of the srp_daemon utility on all system live InfiniBand ports.

srp_daemon

srp_daemon utility is based on ibsrpdm and extends its functionality. In addition to the ibsrpdm functionality described above, srp_daemon can:

- Establish an SRP connection by itself (without the need to issue the "echo" command described in "Manually Establishing an SRP Connection" section)
- Continue running in background, detecting new targets and establishing SRP connections with them (daemon mode)

- Discover reachable SRP Targets given an infiniband HCA name and port, rather than just by `/dev/umad<N>` where `<N>` is a digit

- Enable High Availability operation (together with Device-Mapper Multipath)

- Have a configuration file that determines the targets to connect to:

1. srp_daemon commands equivalent to ibsrpdm:

```
"srp_daemon -a -o" is equivalent to "ibsrpdm"
"srp_daemon -c -a -o" is equivalent to "ibsrpdm -c"
```

Note: These srp_daemon commands can behave differently than the equivalent ibsrpdm command when /etc/srp_daemon.conf is not empty.

2. srp_daemon extensions to ibsrpdm.

- To discover SRP Targets reachable from the HCA device <InfiniBand HCA name> and the port <port num>, (and to generate output suitable for 'echo'), execute:

```
host1# srp_daemon -c -a -o -i <InfiniBand HCA name> -p <port
number>
```

Note: To obtain the list of InfiniBand HCA device names, you can either use the ibstat tool or run ‘ls /sys/class/infiniband’.

- To both discover the SRP Targets and establish connections with them, just add the -e option to the above command.

- Executing srp_daemon over a port without the -a option will only display the reachable targets via the port and to which the initiator is not connected. If executing with the -e option it is better to omit -a.

- It is recommended to use the -n option. This option adds the initiator_ext to the connecting string (see "Multiple Connections from Initiator InfiniBand Port to the Target" section).

- srp_daemon has a configuration file that can be set, where the default is /etc/srp_daemon.conf. Use the -f to supply a different configuration file that configures the targets srp_daemon is allowed to connect to. The configuration file can also be used to set values for additional parameters (e.g., max_cmd_per_lun, max_sect).
- A continuous background (daemon) operation, providing an automatic ongoing detection and connection capability. See "Automatic Discovery and Connection to Targets" section.

Automatic Discovery and Connection to Targets

- Make sure the ib_srp module is loaded, the SRP Initiator can reach an SRP Target, and that an SM is running.
- To connect to all the existing Targets in the fabric, run "`srp_daemon -e -o`". This utility will scan the fabric once, connect to every Target it detects, and then exit.

> **Note**
>
> srp_daemon will follow the configuration it finds in /etc/srp_daemon.conf. Thus, it will ignore a target that is disallowed in the configuration file.

- To connect to all the existing Targets in the fabric and to connect to new targets that will join the fabric, execute srp_daemon -e. This utility continues to execute until it is either killed by the user or encounters connection errors (such as no SM in the fabric).
- To execute SRP daemon as a daemon on all the ports:
  - srp_daemon.sh (found under /usr/sbin/). srp_daemon.sh sends its log to /var/log/srp_daemon.log.
  - Start the srpd service script, run service srpd start

For the changes in openib.conf to take effect, run:

```
/etc/init.d/openibd restart
```

Multiple Connections from Initiator InfiniBand Port to the Target

Some system configurations may need multiple SRP connections from the SRP Initiator to the same SRP Target: to the same Target IB port, or to different IB ports on the same Target HCA.

In case of a single Target IB port, i.e., SRP connections use the same path, the configuration is enabled using a different initiator_ext value for each SRP connection. The initiator_ext value is a 16-hexadecimal-digit value specified in the connection command.

Also in case of two physical connections (i.e., network paths) from a single initiator IB port to two different IB ports on the same Target HCA, there is need for a different initiator_ext value on each path. The conventions is to use the Target port GUID as the initiator_ext value for the relevant path.

If you use srp_daemon with -n flag, it automatically assigns initiator_ext values according to this convention. For example:

```
id_ext=200500A0B81146A1,ioc_guid=0002c90200402bec,\
dgid=fe800000000000000002c90200402bed,pkey=ffff,\
service_id=200500a0b81146a1,initiator_ext=ed2b400002c90200
```

Notes:

- It is recommended to use the -n flag for all srp_daemon invocations.
- ibsrpdm does not have a corresponding option.
- srp_daemon.sh always uses the -n option (whether invoked manually by the user, or automatically at startup by setting SRP_DAEMON_ENABLE to yes).

High Availability (HA)

High Availability works using the Device-Mapper (DM) multipath and the SRP daemon. Each initiator is connected to the same target from several ports/HCAs. The DM multipath is responsible for joining together different paths to the same target and for failover between paths when one of them goes offline. Multipath will be executed on newly joined SCSI devices.

Each initiator should execute several instances of the SRP daemon, one for each port. At startup, each SRP daemon detects the SRP Targets in the fabric and sends requests to the ib_srp module to connect to each of them. These SRP daemons also detect targets that subsequently join the fabric, and send the ib_srp module requests to connect to them as well.

Operation

When a path (from port1) to a target fails, the ib_srp module starts an error recovery process. If this process gets to the reset_host stage and there is no path to the target from this port, ib_srp will remove this scsi_host. After the scsi_host is removed, multipath switches to another path to this target (from another port/HCA).

When the failed path recovers, it will be detected by the SRP daemon. The SRP daemon will then request ib_srp to connect to this target. Once the connection is up, there will be a new scsi_host for this target. Multipath will be executed on the devices of this host, returning to the original state (prior to the failed path).

Manual Activation of High Availability

Initialization - execute after each boot of the driver:

1. Execute `modprobe dm-multipath`
2. Execute `modprobe ib-srp`
3. Make sure you have created file /etc/udev/rules.d/91-srp.rules as described above
4. Execute for each port and each HCA:

```
srp_daemon -c -e -R 300 -i <InfiniBand HCA name> -p <port
```

```
number>
```

This step can be performed by executing srp_daemon.sh, which sends its log to /var/log/srp_daemon.log.

Now it is possible to access the SRP LUNs on /dev/mapper/.

Note

It is possible for regular (non-SRP) LUNs to also be present; the SRP LUNs may be identified by their names. You can configure the /etc/multipath.conf file to change multipath behavior.

Note

It is also possible that the SRP LUNs will not appear under /dev/mapper/. This can occur if the SRP LUNs are in the black-list of multipath. Edit the ‘blacklist’ section in /etc/multipath.conf and make sure the SRP LUNs are not blacklisted.

Automatic Activation of High Availability

- Start srpd service, run:

```
service srpd start
```

- From the next loading of the driver it will be possible to access the SRP LUNs on /dev/mapper/

> **Note**
>
> It is possible that regular (not SRP) LUNs are also present. SRP LUNs may be identified by their name.

- It is possible to see the output of the SRP daemon in /var/log/srp_daemon.log

Shutting Down SRP

SRP can be shutdown by using “rmmod ib_srp”, or by stopping the OFED driver (“/etc/init.d/openibd stop”), or as a by-product of a complete system shutdown.

Prior to shutting down SRP, remove all references to it. The actions you need to take depend on the way SRP was loaded. There are three cases:

1. Without High Availability

   When working without High Availability, you should unmount the SRP partitions that were mounted prior to shutting down SRP.

2. After Manual Activation of High Availability

   If you manually activated SRP High Availability, perform the following steps:

   1. Unmount all SRP partitions that were mounted.

   2. Stop service srpd (Kill the SRP daemon instances).

   3. Make sure there are no multipath instances running. If there are multiple instances, wait for them to end or kill them.

   4. Run: `multipath -F`

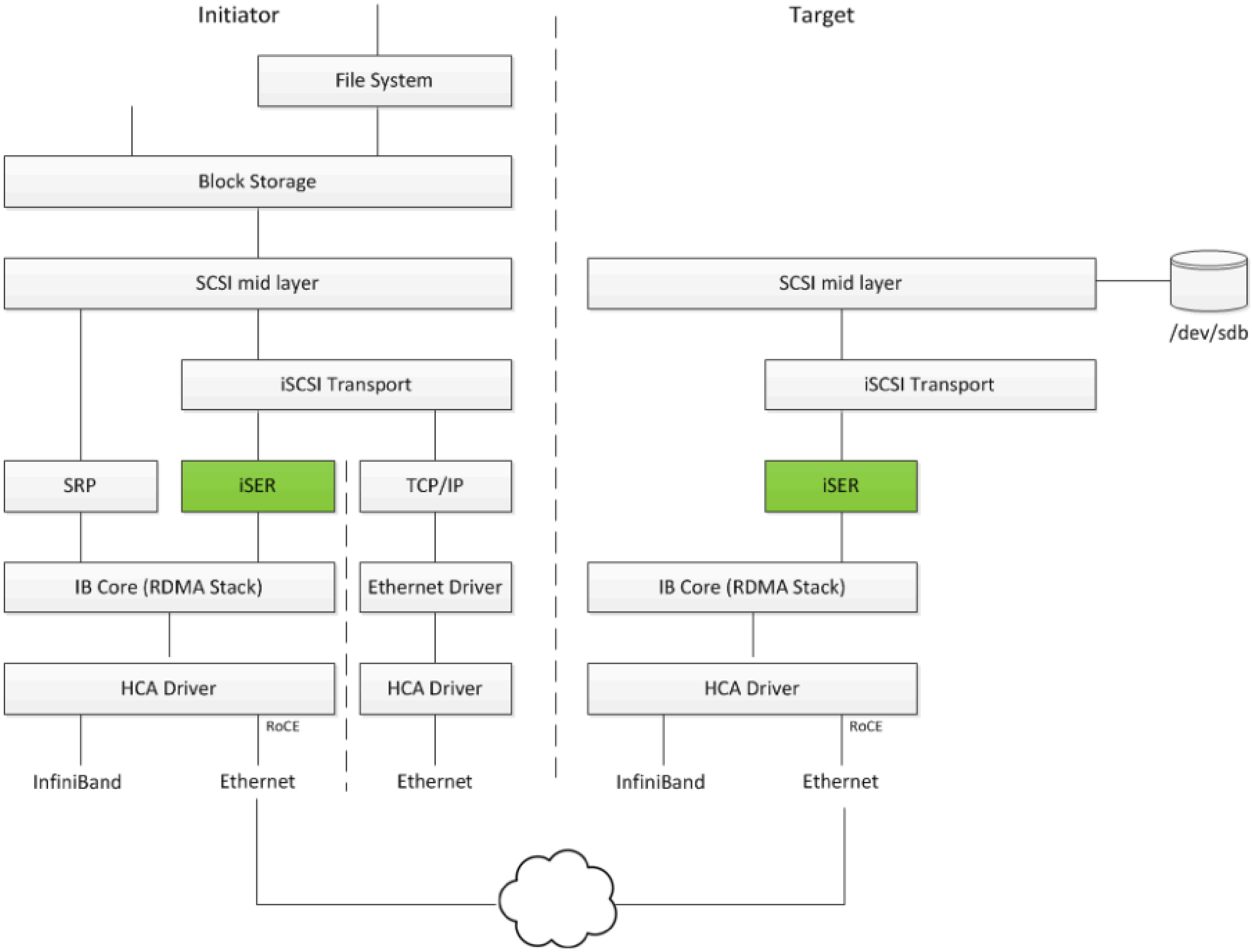
3. After Automatic Activation of High Availability

   If SRP High Availability was automatically activated, SRP shutdown must be part of the driver shutdown ("/etc/init.d/openibd stop") which performs Steps 2-4 of case b above. However, you still have to unmount all SRP partitions that were mounted before driver shutdown.

iSCSI Extensions for RDMA (iSER)

iSCSI Extensions for RDMA (iSER) extends the iSCSI protocol to RDMA. It permits data to be transferred directly into and out of SCSI buffers without intermediate data copies.

iSER uses the RDMA protocol suite to supply higher bandwidth for block storage transfers (zero time copy behavior). To that fact, it eliminates the TCP/IP processing overhead while preserving the compatibility with iSCSI protocol.



There are three target implementation of ISER:

- Linux SCSI target framework (tgt)
- Linux-IO target (LIO)
- Generic SCSI target subsystem for Linux (SCST)

Each one of those targets can work in TCP or iSER transport modes.

iSER also supports RoCE without any additional configuration required. To bond the RoCE interfaces, set the fail_over_mac option in the bonding driver (see "Bonding IPoIB").

RDMA/RoCE is located below the iSER block on the network stack. In order to run iSER, the RDMA layer should be configured and validated (over Ethernet or InfiniBand). For troubleshooting RDMA, please refer to "HowTo Enable, Verify and Troubleshoot RDMA" on the Community website.

iSER Initiator

The iSER initiator is controlled through the iSCSI interface available from the iscsi-initiator-utils package.

To discover and log into iSCSI targets, as well as access and manage the open-iscsi database use the `iscasiadm` utility, a command-line tool.

To enable iSER as a transport protocol use "`I iser`" as a parameter of the `iscasiadm` command.

Example for discovering and connecting targets over iSER:

```
iscsiadm -m discovery -o new -o old -t st -I iser -p <ip:port> -l
```

Note that the target implementation (e.g. LIO, SCST, TGT) does not affect he initiator process and configuration.

iSER Targets

Note

Setting the iSER target is out of scope of this manual. For guidelines of how to do so, please refer to the relevant target documentation (e.g. stgt, targetcli).

Targets settings such as timeouts and retries are set the same as any other iSCSI targets.

ⓘ **Note**

If targets are set to auto connect on boot, and targets are unreachable, it may take a long time to continue the boot process if timeouts and max retries are set too high.

For various configuration, troubleshooting and debugging examples, refer to Storage Solutions on the Community website.

Lustre

Lustre is an open-source, parallel distributed file system, generally used for large-scale cluster computing that supports many requirements of leadership class HPC simulation environments.

Lustre Compilation for MLNX_OFED:

> **Note**
>
> This procedure applies to RHEL/SLES OSs supported by Lustre. For further information, please refer to Lustre Release Notes.

➤ ***To compile Lustre version 2.4.0 and higher:***

```
$ ./configure --with-o2ib=/usr/src/ofa_kernel/default/
$ make rpms
```

```
$ EXTRA_LNET_INCLUDE="-I/usr/src/ofa_kernel/default/include/ -include /usr/src/ofa_kernel/default/include/linux/compat-2.6.h" ./configure --with-o2ib=/usr/src/ofa_kernel/default/
$ EXTRA_LNET_INCLUDE="-I/usr/src/ofa_kernel/default/include/ -include /usr/src/ofa_kernel/default/include/linux/compat-2.6.h" make rpms
```

For full installation example, refer to HowTo Install NVIDIA OFED driver for Lustre Community post.

NVME-oF - NVM Express over Fabrics

NVME-oF

NVME-oF enables NVMe message-based commands to transfer data between a host computer and a target solid-state storage device or system over a network such as Ethernet, Fibre Channel, and InfiniBand. Tunneling NVMe commands through an RDMA fabric provides a high throughput and a low latency.

For information on how to configure NVME-oF, please refer to the HowTo Configure NVMe over Fabrics Community post.

ⓘ **Note**

The --with-nvmf installation option should **not** be specified, if nvme-tcp kernel module is used. In this case, the native Inbox nvme-tcp kernel module will be loaded.

NVME-oF Target Offload

ⓘ **Note**

This feature is only supported for ConnectX-5 adapter cards family and above.

NVME-oF Target Offload is an implementation of the new NVME-oF standard Target (server) side in hardware. Starting from ConnectX-5 family cards, all regular IO requests can be processed by the HCA, with the HCA sending IO requests directly to a real NVMe PCI device, using peer-to-peer PCI communications. This means that excluding connection management and error flows, no CPU utilization will be observed during NVME-oF traffic.

- For instructions on how to configure NVME-oF target offload, refer to HowTo Configure NVME-oF Target Offload Community post.
- For instructions on how to verify that NVME-oF target offload is working properly, refer to Simple NVMe-oF Target Offload Benchmark Community post.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.