Red Hat Enterprise Linux 9

Configuring InfiniBand and RDMA networks

A guide to configuring InfiniBand and RDMA networks on Red Hat Enterprise Linux 9

Last Updated: 2022-05-18
Red Hat Enterprise Linux 9 Configuring InfiniBand and RDMA networks

A guide to configuring InfiniBand and RDMA networks on Red Hat Enterprise Linux 9
Abstract

This document describes what InfiniBand and remote direct memory access (RDMA) are and how to configure InfiniBand hardware. Additionally, this documentation explains how to configure InfiniBand-related services.
Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE .......................................................... 3
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION .................................. 4
CHAPTER 1. UNDERSTANDING INFINIBAND AND RDMA .................................. 5

CHAPTER 2. CONFIGURING ROCE ................................................................. 6
2.1. OVERVIEW OF ROCE PROTOCOL VERSIONS ........................................... 6
2.2. TEMPORARILY CHANGING THE DEFAULT ROCE VERSION ......................... 6

CHAPTER 3. CONFIGURING AN INFINIBAND SUBNET MANAGER ....................... 8

CHAPTER 4. CONFIGURING IPOIB ................................................................. 9
4.1. THE IPOIB COMMUNICATION MODES .................................................... 9
4.2. UNDERSTANDING IPOIB HARDWARE ADDRESSES ................................ 9
4.3. CONFIGURING AN IPOIB CONNECTION USING NMCLI COMMANDS ........ 10
4.4. CONFIGURING AN IPOIB CONNECTION USING NM-CONNECTION-EDITOR 11

CHAPTER 5. TESTING INFINIBAND NETWORKS .............................................. 13
5.1. TESTING EARLY INFINIBAND RDMA OPERATIONS .................................. 13
5.2. TESTING AN IPOIB USING THE PING UTILITY ...................................... 15
Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message.
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Please let us know how we could make it better.

- For simple comments on specific passages:
  1. Make sure you are viewing the documentation in the *Multi-page HTML* format. In addition, ensure you see the **Feedback** button in the upper right corner of the document.
  2. Use your mouse cursor to highlight the part of text that you want to comment on.
  3. Click the **Add Feedback** pop-up that appears below the highlighted text.
  4. Follow the displayed instructions.

- For submitting feedback via Bugzilla, create a new ticket:
  1. Go to the [Bugzilla](http://bugzilla.redhat.com) website.
  2. As the Component, use **Documentation**.
  3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
  4. Click **Submit Bug**.
CHAPTER 1. UNDERSTANDING INFINIBAND AND RDMA

InfiniBand refers to two distinct things:

- The physical link-layer protocol for InfiniBand networks
- The InfiniBand Verbs API, which is an implementation of the remote direct memory access (RDMA) technology

RDMA provides access to the memory from one computer to the memory of another computer without involving either computer’s operating system. This technology enables high-throughput and low-latency networking with low CPU utilization.

In a typical IP data transfer, when an application on one machine sends data to an application on another machine, the following happens on the receiving side:

1. The kernel must receive the data.
2. The kernel must determine that the data belongs to the application.
3. The kernel wakes up the application.
4. The kernel waits for the application to perform a system call into the kernel.
5. The application copies the data from the kernel’s own internal memory space into the buffer provided by the application.

This process means that most network traffic is copied across the main memory of the system if the host adapter uses direct memory access (DMA), or otherwise at least twice. Additionally, the computer executes a number of context switches to switch between the kernel and application context. Both context switches can cause a high CPU load at high traffic rates and slow down other tasks.

RDMA communication bypasses the kernel intervention in the communication process, unlike the normal IP communication. This reduces the CPU overhead. The RDMA protocol enables the host adapter to know when a packet comes in from the network, which application should receive it, and where in the application’s memory space, the packet should be stored. Instead of sending the packet to the kernel to be processed and then copied into the user application’s memory, with InfiniBand, the host adapter places the contents of the packet directly in the application’s buffer. This process requires a separate API, the InfiniBand Verbs API, and applications must support this API before they can use RDMA.

Red Hat Enterprise Linux supports both the InfiniBand hardware and the InfiniBand Verbs API. Additionally, it supports the following technologies that allow to use the InfiniBand Verbs API on non-InfiniBand hardware:

- Internet Wide Area RDMA Protocol (iWARP): A network protocol that implements RDMA over IP networks.
- RDMA over Converged Ethernet (RoCE), which is also known as InfiniBand over Ethernet (IBoE): A network protocol that implements RDMA over Ethernet networks.

Additional resources

- Configuring RoCE
CHAPTER 2. CONFIGURING ROCE

This section explains background information about RDMA over Converged Ethernet (RoCE), as well as how to change the default RoCE version.

Note that there are different vendors, such as Mellanox, Broadcom, and QLogic, who provide RoCE hardware.

2.1. OVERVIEW OF ROCE PROTOCOL VERSIONS

RoCE is a network protocol that enables remote direct memory access (RDMA) over Ethernet.

The following are the different RoCE versions:

**RoCE v1**

The RoCE version 1 protocol is an Ethernet link layer protocol with ethertype 0x8915 that enables communication between any two hosts in the same Ethernet broadcast domain.

**RoCE v2**

The RoCE version 2 protocol exists on top of either the UDP over IPv4 or the UDP over IPv6 protocol. The UDP destination port number 4791 is reserved for RoCE v2.

The RDMA_CM sets up a reliable connection between a client and a server for transferring data. RDMA_CM provides an RDMA transport-neutral interface for establishing connections. The communication uses a specific RDMA device, and data transfers are message-based.

**IMPORTANT**

Using RoCE v2 on the client and RoCE v1 on the server is not supported. In this case, configure both the server and client to communicate over RoCE v1.

RoCE v1 works at link layer (layer 2). It only supports communication of two machines in the same network. While in case of RoCE v2, it works at network layer (layer 3). RoCE v2 supports packets routing that provides connection with multiple Ethernet. By default, RoCE v2 is used.

Additional resources

- Temporarily changing the default RoCE version

2.2. TEMPORARILY CHANGING THE DEFAULT ROCE VERSION

Using the RoCE v2 protocol on the client and RoCE v1 on the server is not supported. If the hardware in your server only supports RoCE v1, configure your clients to communicate with the server using RoCE v1. This section describes how to enforce RoCE v1 on the client that uses the mlx5_0 driver for the Mellanox ConnectX-5 Infiniband device. Note that the changes described in this section are only temporary until you reboot the host.

Prerequisites

- The client uses an InfiniBand device that uses, by default, the RoCE v2 protocol.
- The InfiniBand device in the server only supports RoCE v1.
Procedure

1. Create the `/sys/kernel/config/rdma_cm/mlx5_0` directory:
   
   ```
   # mkdir /sys/kernel/config/rdma_cm/mlx5_0
   ```

2. Display the default RoCE mode. For example, to display the mode for port 1:

   ```
   # cat /sys/kernel/config/rdma_cm/mlx5_0/ports/1/default_roce_mode
   ```

   RoCE v2

3. Change the default RoCE mode to version 1:

   ```
   # echo "IB/RoCE v1" > /sys/kernel/config/rdma_cm/mlx5_0/ports/1/default_roce_mode
   ```
CHAPTER 3. CONFIGURING AN INFINIBAND SUBNET MANAGER

All InfiniBand networks must have a subnet manager running for the network to function. This is true even if two machines are connected directly with no switch involved.

It is possible to have more than one subnet manager. In that case, one acts as a master and another subnet manager acts as a slave that will take over in case the master subnet manager fails.

Most InfiniBand switches contain an embedded subnet manager. However, if you need a more up-to-date subnet manager or if you require more control, use the OpenSM subnet manager provided by Red Hat Enterprise Linux.

For details, see Installing the OpenSM subnet manager
CHAPTER 4. CONFIGURING IPOIB

By default, InfiniBand does not use the internet protocol (IP) for communication. However, IP over InfiniBand (IPoIB) provides an IP network emulation layer on top of InfiniBand remote direct memory access (RDMA) networks. This allows existing unmodified applications to transmit data over InfiniBand networks, but the performance is lower than if the application would use RDMA natively.

NOTE

The Mellanox devices starting from ConnectX-4 and above on RHEL 8 use Enhanced IPoIB mode by default (datagram only). Connected mode is not supported on these devices.

4.1. THE IPOIB COMMUNICATION MODES

You can configure an IPoIB device either in Datagram or Connected mode. The difference is the type of queue pair the IPoIB layer attempts to open with the machine at the other end of the communication:

- In the **Datagram** mode, the system opens an unreliable, disconnected queue pair. This mode does not support packages larger than the InfiniBand link-layer's Maximum Transmission Unit (MTU). During transmission of data, the IPoIB layer adds a 4 byte IPoIB header on top of the IP packet. As a result, the IPoIB MTU is 4 bytes less than the InfiniBand link-layer MTU. As 2048 is a common InfiniBand link-layer MTU, the common IPoIB device MTU in **Datagram** mode is 2044.

- In the **Connected** mode, the system opens a reliable, connected queue pair. This mode allows messages larger than the InfiniBand link-layer MTU, and the host adapter handles packet segmentation and reassembly. As a result, there is no size limit imposed on the size of IPoIB messages that can be sent by InfiniBand adapters in **Connected** mode. However, IP packets are limited because of the size field and TCP/IP headers. For this reason, the IPoIB MTU in **Connected** mode is 65520 bytes maximum.

The **Connected** mode has a higher performance, but consumes more kernel memory.

If a system is configured to use the **Connected** mode, it still sends multicast traffic in the **Datagram** mode, because InfiniBand switches and fabric cannot pass multicast traffic in **Connected** mode. Additionally, the system falls back to **Datagram** mode, when communicating with any host that is not configured in the **Connected** mode.

While running application that sends multicast data up to the maximum MTU on the interface, you must configure the interface in **Datagram** mode or configure the application to cap the packet send size at a size that will fit in datagram-sized packets.

4.2. UNDERSTANDING IPOIB HARDWARE ADDRESSES

IPoIB devices have a 20 byte hardware address that consists of the following parts:

- The first 4 bytes are flags and queue pair numbers.
- The next 8 bytes are the subnet prefix.

The default subnet prefix is **0xfe:80:00:00:00:00:00:00**. After the device connects to the subnet manager, the device changes this prefix to match the one configured in the subnet manager.
The last 8 bytes are the Globally Unique Identifier (GUID) of the InfiniBand port that the IPoIB device is attached to.

**NOTE**
Because the first 12 bytes can change, don’t use them in udev device manager rules.

Additional resources
- Renaming IPoIB devices

### 4.3. CONFIGURING AN IPOIB CONNECTION USING NMCLI COMMANDS

This procedure describes how to configure an IPoIB connection using `nmcli` commands.

**Prerequisites**
- An InfiniBand device is installed on the server, and the corresponding kernel module is loaded.

**Procedure**

1. Create the InfiniBand connection. For example, to create a connection that uses the mlx4_ib0 interface in the **Connected** transport mode and the maximum MTU of **65520** bytes, enter:

   ```
   # nmcli connection add type infiniband con-name mlx4_ib0 ifname mlx4_ib0 transport-mode Connected mtu 65520
   ```

2. Optional: set a **P_Key** interface. For example, to set **0x8002** as **P_Key** interface of the mlx4_ib0 connection, enter:

   ```
   # nmcli connection modify mlx4_ib0 infiniband.p-key 0x8002
   ```

3. Configure the IPv4 settings. For example, to set a static IPv4 address, network mask, default gateway, and DNS server of the mlx4_ib0 connection, enter:

   ```
   # nmcli connection modify mlx4_ib0 ipv4.addresses 192.0.2.1/24
   # nmcli connection modify mlx4_ib0 ipv4.gateway 192.0.2.254
   # nmcli connection modify mlx4_ib0 ipv4.dns 192.0.2.253
   # nmcli connection modify mlx4_ib0 ipv4.method manual
   ```

4. Configure the IPv6 settings. For example, to set a static IPv6 address, network mask, default gateway, and DNS server of the mlx4_ib0 connection, enter:

   ```
   # nmcli connection modify mlx4_ib0 ipv6.addresses 2001:db8:1::1/32
   # nmcli connection modify mlx4_ib0 ipv6.gateway 2001:db8:1::ffe
   # nmcli connection modify mlx4_ib0 ipv6.dns 2001:db8:1::fffd
   # nmcli connection modify mlx4_ib0 ipv6.method manual
   ```

5. Activate the connection. For example, to activate the mlx4_ib0 connection:

   ```
   # nmcli connection up mlx4_ib0
   ```
4.4. CONFIGURING AN IPOIB CONNECTION USING NM-CONNECTION-EDITOR

This procedure describes how to configure an IPoIB connection using the `nm-connection-editor` application.

**Prerequisites**

- An InfiniBand device is installed in the server, and the corresponding kernel module is loaded.
- The `nm-connection-editor` package is installed.

**Procedure**

1. Open a terminal, and enter:
   ```
   $ nm-connection-editor
   ```
2. Click the + button to add a new connection.
3. Select the InfiniBand connection type, and click Create.
4. On the InfiniBand tab:
   a. Optionally, change the connection name.
   b. Select the transport mode.
   c. Select the device.
   d. Optional: set an MTU.
5. On the IPv4 Settings tab, configure the IPv4 settings. For example, set a static IPv4 address, network mask, default gateway, and DNS server:
6. On the **IPv6 Settings** tab, configure the IPv6 settings. For example, set a static IPv6 address, network mask, default gateway, and DNS server:

<table>
<thead>
<tr>
<th>IPv6 Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Connection name:</strong> mlx4_ib0</td>
</tr>
<tr>
<td><strong>Method:</strong> Manual</td>
</tr>
<tr>
<td><strong>Addresses</strong></td>
</tr>
<tr>
<td><strong>Address</strong></td>
</tr>
<tr>
<td>2001:db8::1</td>
</tr>
<tr>
<td><strong>DNS servers:</strong></td>
</tr>
</tbody>
</table>

7. Click **Save** to save the team connection.

8. Close **nm-connection-editor**.

9. Optional: set a **P_Key** interface. Note that you must set this parameter on the command line, because the setting is not available in **nm-connection-editor**.
   For example, to set **0x8002** as **P_Key** interface of the **mlx4_ib0** connection, enter:

```
# nmcli connection modify mlx4_ib0 infiniband.p-key 0x8002
```
CHAPTER 5. TESTING INFINIBAND NETWORKS

This section provides procedures how to test InfiniBand networks.

5.1. TESTING EARLY INFINIBAND RDMA OPERATIONS

This section describes how to test InfiniBand remote direct memory access (RDMA) operations.

NOTE

This section applies only to InfiniBand devices. If you use iWARP or RoCE/IBoE devices, which are IP-based, see:

- Testing an IPoIB using the ping utility
- Testing an RDMA network using qperf after IPoIB is configured

Prerequisites

- RDMA is configured.
- The `libibverbs-utils` and `infiniband-diags` packages are installed.

Procedure

1. List the available InfiniBand devices:

   ```
   # ibv_devices
   
   device          node GUID
   --------        ----------------
   mlx4_0          0002c903003178f0
   mlx4_1          f4521403007bca0
   ```

2. Display the information for a specific InfiniBand device. For example, to display the information of the `mlx4_1` device, enter:

   ```
   # ibv_devinfo -d mlx4_1
   
   hca_id: mlx4_1
   transport: InfiniBand (0)
   fw_ver: 2.30.8000
   node_guid: f452:1403:007b:cba0
   sys_image_guid: f452:1403:007b:cba3
   vendor_id: 0x02c9
   vendor_part_id: 4099
   hw_ver: 0x0
   board_id: MT_1090120019
   phys_port_cnt: 2
   port: 1
   state: PORT_ACTIVE (4)
   max_mtu: 4096 (5)
   active_mtu: 2048 (4)
   sm_lid: 2
   ```
3. Display the basic status of an InfiniBand device. For example, to display the status of the mlx4_1 device, enter:

```
# ibstat mlx4_1
```

CA 'mlx4_1'
CA type: MT4099
Number of ports: 2
Firmware version: 2.30.8000
Hardware version: 0
Node GUID: 0xf4521403007bcba0
System image GUID: 0xf4521403007bcba3
Port 1:
  State: Active
  Physical state: LinkUp
  Rate: 56
  Base lid: 2
  LMC: 1
  SM lid: 2
  Capability mask: 0x0251486a
  Port GUID: 0xf4521403007bcba1
  Link layer: InfiniBand

Port 2:
  State: Active
  Physical state: LinkUp
  Rate: 40
  Base lid: 0
  LMC: 0
  SM lid: 0
  Capability mask: 0x04010000
  Port GUID: 0xf65214fffe7bcba2
  Link layer: Ethernet

4. Use the ibping utility to ping from a client to a server using InfiniBand:

   a. On the host that acts as a server, start ibping in server mode:

```
# ibping -S -C mlx4_1 -P 1
```

   This command uses the following parameters:

   - **-S**: Enables the server mode.
-C InfiniBand_CA_name: Sets the CA name to use.

-P port_number: Sets the port number to use, if the InfiniBand provides multiple ports.

b. On the host that acts as client, use `ibping` as follows:

```
# ibping -c 50 -C mlx4_0 -P 1 -L 2
```

-c number: Sends these number of packets to the server.

-C InfiniBand_CA_name: Sets the CA name to use.

-P port_number: Sets the port number to use, if the InfiniBand provides multiple ports.

-L port_LID: Sets the Local Identifier (LID) to use.

Additional resources

- `ibping(8)` man page

### 5.2. TESTING AN IPOIB USING THE PING UTILITY

After you configured IPoIB, use the `ping` utility to send ICMP packets to test the IPoIB connection.

**Prerequisites**

- The two RDMA hosts are connected in the same InfiniBand fabric with RDMA ports.
- The IPoIB interfaces in both hosts are configured with IP addresses within the same subnet.

**Procedure**

- Use the `ping` utility to send ICMP packets to the remote host’s InfiniBand adapter:

```
# ping -c5 192.0.2.1
```

This command sends five ICMP packets to the IP address `192.0.2.1`. 