Red Hat Enterprise Linux 8

InfiniBand ネットワークおよび RDMA ネットワークの設定

高速ネットワークプロトコルと RDMA ハードウェアの設定と管理

Last Updated: 2024-09-21

Red Hat Enterprise Linux 8 InfiniBand ネットワークおよび RDMA ネットワークの設定

高速ネットワークプロトコルと RDMA ハードウェアの設定と管理

法律上の通知

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

さまざまなプロトコルを使用して、Remote Directory Memory Access (RDMA) ネットワークと InfiniBand ハードウェアをエンタープライズレベルで設定および管理できます。これには、RDMA over Converged Ethernet (RoCE)、RoCE のソフトウェア実装 (Soft-RoCE)、iWARP などの IP ネットワークプロトコル、iWARP のソフトウェア実装 (Soft-iWARP)、および RDMA 対応ハードウェアのネイティブサポートとしての Network File System over RDMA (NFSoRDMA) プロトコルが含まれます。低レイテンシーで高スループットの接続を実現するために、IP over InfiniBand (IPoIB) および Open Subnet Manager (OpenSM) を設定できます。

目次

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに関するご意見やご感想をお寄せください。また、改善点があればお知らせください。

Jira からのフィードバック送信 (アカウントが必要)

1. Jira の Web サイトにログインします。
2. 上部のナビゲーションバーで **Create** をクリックします。
3. **Summary** フィールドにわかりやすいタイトルを入力します。
4. **Description** フィールドに、ドキュメントの改善に関するご意見を記入してください。ドキュメントの該当部分へのリンクも追加してください。
5. ダイアログの下部にある **Create** をクリックします。

第1章 INFINIBAND および RDMA について

InfiniBand は、次の 2 つの異なるものを指します。

- InfiniBand ネットワーク用の物理リンク層プロトコル
- Remote Direct Memory Access (RDMA) テクノロジーの実装である InfiniBand Verbs API

RDMA は、オペレーティングシステム、キャッシュ、またはストレージを使用せずに、2 台のコンピューターのメインメモリー間のアクセスを提供します。RDMA を使用することで、高スループット、低レイテンシー、低 CPU 使用率でデータ転送が可能になります。

通常の IP データ転送では、あるマシンのアプリケーションが別のマシンのアプリケーションにデータを送信すると、受信側で以下のアクションが起こります。

1. カーネルがデータを受信する必要がある。
2. カーネルは、データがアプリケーションに属するかどうかを判別する必要がある。
3. カーネルは、アプリケーションを起動する。
4. カーネルは、アプリケーションがカーネルへのシステムコールを実行するまで待機する。
5. アプリケーションは、データをカーネルの内部メモリー領域から、アプリケーションが提供するバッファーにコピーする。

このプロセスでは、ホストアダプターが直接メモリーアクセス (DMA) などを使用する場合には、ほとんどのネットワークトラフィックが、システムのメインメモリーに少なくとも 2 回コピーされます。さらに、コンピューターはいくつかのコンテキストスイッチを実行して、カーネルとアプリケーションを切り替えます。これらのコンテキストスイッチは、他のタスクの速度を低下させる一方で、高いトラフィックレートで高い CPU 負荷を引き起こす可能性があります。

従来の IP 通信とは異なり、RDMA 通信は通信プロセスでのカーネルの介入を回避します。これにより、CPU のオーバーヘッドが軽減されます。RDMA プロトコルは、パケットがネットワークに入った後、どのアプリケーションがそれを受信し、そのアプリケーションのメモリー空間のどこに格納するかをホストアダプターが決定することを可能にします。処理のためにパケットをカーネルに送信してユーザーアプリケーションのメモリーにコピーする代わりに、ホストアダプターは、パケットの内容をアプリケーションバッファーに直接配置します。このプロセスには、別個の API である InfiniBand Verbs API が必要であり、アプリケーションは RDMA を使用するために InfiniBand Verbs API を実装する必要があります。

Red Hat Enterprise Linux は、InfiniBand ハードウェアと InfiniBand Verbs API の両方をサポートしています。さらに、InfiniBand 以外のハードウェアで InfiniBand Verbs API を使用するための次のテクノロジーをサポートしています。

- Internet Wide Area RDMA Protocol (iWARP): IP ネットワーク上で RDMA を実装するネットワークプロトコル
- RDMA over Converged Ethernet (RoCE)、別名 InfiniBand over Ethernet (IBoE): RDMA over Ethernet ネットワークを実装するネットワークプロトコル

関連情報

- RoCE の設定

第2章 RDMA サービスの設定

Remote Direct Memory Access (RDMA) プロトコルを使用すると、メインメモリーを使用して、ネットワーク経由で RDMA 対応システム間でデータを転送できます。RDMA プロトコルは、低レイテンシーと高スループットを実現します。サポートされているネットワークプロトコルと通信標準を管理するには、**rdma** サービスを設定する必要があります。この設定には、RoCE や iWARP などの高速ネットワークプロトコル、および Soft-RoCE や Soft-iWARP などの通信標準が含まれます。Red Hat Enterprise Linux が、InfiniBand、iWARP、または RoCE デバイスおよびそれらの設定ファイルが **/etc/rdma/modules/*** ディレクトリーに存在することを検出すると、**udev** デバイスマネージャーが **systemd** に **rdma** サービスを起動するように指示します。**/etc/rdma/modules/rdma.conf** ファイル内のモジュールの設定は、再起動後も保持されます。変更を適用するには、**rdma-load-modules@rdma.service** 設定サービスを再起動する必要があります。

手順

1. **rdma-core** パッケージをインストールします。

```
# dnf install rdma-core
```

2. **/etc/rdma/modules/rdma.conf** ファイルを編集し、有効にするモジュールのコメントを解除します。

```
# These modules are loaded by the system if any RDMA devices is installed

# iSCSI over RDMA client support
ib_iser

# iSCSI over RDMA target support
ib_isert

# SCSI RDMA Protocol target driver
ib_srpt

# User access to RDMA verbs (supports libibverbs)
ib_uverbs

# User access to RDMA connection management (supports librdmacm)
rdma_ucm

# RDS over RDMA support
# rds_rdma

# NFS over RDMA client support
xprtrdma

# NFS over RDMA server support
svcrdma
```

3. サービスを再起動して変更を有効にします。

```
# systemctl restart <rdma-load-modules@rdma.service>
```

検証

1. **libibverbs-utils** および **infiniband-diags** パッケージをインストールします。

```
# dnf install libibverbs-utils infiniband-diags
```

2. 利用可能な InfiniBand デバイスのリストを表示します。

```
# ibv_devices

    device                 node GUID
    ------              ----------------
    mlx4_0              0002c903003178f0
    mlx4_1              f4521403007bcba0
```

3. **mlx4_1** デバイスの情報を表示します。

```
# 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
                        port_lid:           2
                        port_lmc:           0x01
                        link_layer:         InfiniBand

                port:   2
                        state:              PORT_ACTIVE (4)
                        max_mtu:            4096 (5)
                        active_mtu:         4096 (5)
                        sm_lid:             0
                        port_lid:           0
                        port_lmc:           0x00
                        link_layer:         Ethernet
```

4. **mlx4_1** デバイスのステータスを表示します。

```
# 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
```

5. **ibping** ユーティリティーは、パラメーターを設定することで InfiniBand アドレスに ping を実行し、クライアント/サーバーとして動作します。

   a. ホスト上の **-C** InfiniBand 認証局 (CA) 名を使用して、ポート番号 **-P** でサーバーモード **-S** を開始します。

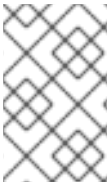
   ```
   # ibping -S -C mlx4_1 -P 1
   ```

   b. クライアントモードを開始し、ホスト上の **-C** InfiniBand 認証局 (CA) 名と **-L** ローカル識別子 (LID) を使用して、ポート番号 **-P** でいくつかのパケット **-c** を送信します。

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

第3章 IPOIB の設定

デフォルトでは、InfiniBand は通信にインターネットプロトコル (IP) を使用しません。ただし、IPoIB (IP over InfiniBand) は、InfiniBand Remote Direct Memory Access (RDMA) ネットワーク上に IP ネットワークエミュレーション層を提供します。これにより、変更を加えていない既存のアプリケーションが InfiniBand ネットワーク経由でデータを送信できるようになりますが、アプリケーションが RDMA をネイティブに使用する場合よりもパフォーマンスが低下します。



注記

RHEL 8 以降の Mellanox デバイス (ConnectX-4 以降) は、デフォルトで Enhanced IPoIB モードを使用します (データグラムのみ)。これらのデバイスでは、Connected モードはサポートされていません。

3.1. IPOIB の通信モード

IPoIB デバイスは、**Datagram** モードまたは **Connected** モードのいずれかで設定可能です。違いは、通信の反対側で IPoIB 層がマシンで開こうとするキューペアのタイプです。

- **Datagram** モードでは、システムは信頼できない非接続のキューペアを開きます。
  このモードは、InfiniBand リンク層の Maximum Transmission Unit (MTU) を超えるパッケージには対応していません。IPoIB 層は、データ転送時に IP パケットに 4 バイトの IPoIB ヘッダーを追加します。その結果、IPoIB MTU は InfiniBand リンク層 MTU より 4 バイト少なくなります。一般的な InfiniBand リンク層 MTU は **2048** であるため、**Datagram** モードの一般的な IPoIB デバイス MTU は **2044** になります。

- **Connected** モードでは、システムは信頼できる接続されたキューペアを開きます。
  このモードでは、InfiniBand のリンク層の MTU より大きなメッセージを許可します。ホストアダプターがパケットのセグメンテーションと再構築を処理します。その結果、**Connected** モードでは、Infiniband アダプターから送信されるメッセージのサイズに制限がありません。しかし、**data** フィールドと TCP/IP **header** フィールドにより、IP パケットには制限があります。このため、**Connected** モードの IPoIB MTU は **65520** バイトです。

  Connected モードではパフォーマンスが向上しますが、より多くのカーネルメモリーを消費します。

システムが **Connected** モードを使用するように設定されていても、InfiniBand スイッチとファブリックは **Connected** モードでマルチキャストトラフィックを渡すことができないため、システムは引き続き **Datagram** モードを使用してマルチキャストトラフィックを送信します。また、ホストが **Connected** モードを使用するように設定されていない場合、システムは **Datagram** モードにフォールバックします。

インターフェイス上で MTU までのマルチキャストデータを送信するアプリケーションを実行しながら、インターフェイスを **Datagram** モードに設定するか、データグラムサイズのパケットに収まるように、パケットの送信サイズに上限を設けるようにアプリケーションを設定してください。

3.2. IPOIB ハードウェアアドレスについて

IPoIB デバイスには、以下の部分で構成される **20** バイトのハードウェアアドレスがあります。

- 最初の 4 バイトはフラグとキューペアの番号です。
- 次の 8 バイトはサブネットの接頭辞です。

デフォルトのサブネットの接頭辞は **0xfe:80:00:00:00:00:00:00** です。デバイスがサブネットマネージャーに接続すると、デバイスはこの接頭辞を変更して、設定されたサブネットマネージャーと一致させます。

- 最後の 8 バイトは、IPoIB デバイスに接続する InfiniBand ポートのグローバル一意識別子 (GUID) です。

注記

最初の 12 バイトは変更される可能性があるため、**udev** デバイスマネージャールールでは使用しないでください。

3.3. IPOIB デバイスの名前変更

デフォルトでは、カーネルは Internet Protocol over InfiniBand (IPoIB) デバイスに、**ib0**、**ib1** などの名前を付けます。競合を回避するために、Red Hat では、**udev** デバイスマネージャーでルールを作成し、**mlx4_ib0** などの永続的で意味のある名前を作成することを推奨しています。

前提条件

- InfiniBand デバイスがインストールされている。

手順

1. デバイス **ib0** のハードウェアアドレスを表示します。

```
# ip link show ib0
8: ib0: >BROADCAST,MULTICAST,UP,LOWER_UP< mtu 65520 qdisc pfifo_fast state UP
mode DEFAULT qlen 256
    link/infiniband 80:00:02:00:fe:80:00:00:00:00:00:00:00:00:02:c9:03:00:31:78:f2 brd
00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff
```

   アドレスの最後の 8 バイトは、次のステップで **udev** ルールを作成するために必要です。

2. **00:02:c9:03:00:31:78:f2** のハードウェアアドレスを持つデバイスの名前を **mlx4_ib0** に変更するルールを設定するには、**/etc/udev/rules.d/70-persistent-ipoib.rules** ファイルを編集して **ACTION** ルールを追加してください。

```
ACTION=="add", SUBSYSTEM=="net", DRIVERS=="?*", ATTR{type}=="32",
ATTR{address}=="?*00:02:c9:03:00:31:78:f2", NAME="mlx4_ib0"
```

3. ホストを再起動します。

```
# reboot
```

関連情報

- システムの **udev (7)** man ページ
- IPoIB ハードウェアアドレスについて

3.4. NMCLI コマンドを使用した IPOIB 接続の設定

nmcli コマンドラインユーティリティーは、CLI を使用して NetworkManager を制御し、ネットワークステータスを報告します。

前提条件

- InfiniBand デバイスがサーバーにインストールされている。
- 対応するカーネルモジュールがロードされている。

手順

1. InfiniBand 接続を作成して、**Connected** トランスポートモードで **mlx4_ib0** インターフェイスを使用し、最大 MTU が **65520** バイトになるようにします。

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

2. **P_Key** を設定します。次に例を示します。

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

3. IPv4 を設定します。
   - DHCP を使用するには、次のように入力します。

```
# nmcli connection modify mlx4_ib0 ipv4.method auto
```

     ipv4.method がすでに **auto** (デフォルト) に設定されている場合は、この手順をスキップしてください。

   - 静的 IPv4 アドレス、ネットワークマスク、デフォルトゲートウェイ、DNS サーバー、および検索ドメインを設定するには、次のように入力します。

```
# nmcli connection modify mlx4_ib0 ipv4.method manual ipv4.addresses 192.0.2.1/24 ipv4.gateway 192.0.2.254 ipv4.dns 192.0.2.200 ipv4.dns-search example.com
```

4. IPv6 設定を行います。
   - ステートレスアドレス自動設定 (SLAAC) を使用するには、次のように入力します。

```
# nmcli connection modify mlx4_ib0 ipv6.method auto
```

     ipv6.method がすでに **auto** (デフォルト) に設定されている場合は、この手順をスキップしてください。

   - 静的 IPv6 アドレス、ネットワークマスク、デフォルトゲートウェイ、DNS サーバー、および検索ドメインを設定するには、次のように入力します。

```
# nmcli connection modify mlx4_ib0 ipv6.method manual ipv6.addresses 2001:db8:1::fffe/64 ipv6.gateway 2001:db8:1::fffe ipv6.dns 2001:db8:1::ffbb ipv6.dns-search example.com
```

5. プロファイルの他の設定をカスタマイズするには、次のコマンドを使用します。

```
# nmcli connection modify mlx4_ib0 <setting> <value>
```

値はスペースまたはセミコロンで引用符で囲みます。

6. プロファイルをアクティブ化します。

```
# nmcli connection up mlx4_ib0
```

検証

- **ping** ユーティリティーを使用して、ICMP パケットをリモートホストの InfiniBand アダプターに送信します。次に例を示します。

```
# ping -c5 192.0.2.2
```

3.5. NETWORK RHEL システムロールを使用した IPOIB 接続の設定

IP over InfiniBand (IPoIB) を使用すると、InfiniBand インターフェイス経由で IP パケットを送信できます。IPoIB を設定するには、NetworkManager 接続プロファイルを作成します。Ansible と **network** システムロールを使用すると、このプロセスを自動化し、Playbook で定義されたホスト上の接続プロファイルをリモートで設定できます。

network RHEL システムロールを使用して IPoIB を設定できます。InfiniBand の親デバイスの接続プロファイルが存在しない場合は、このロールによって作成することもできます。

前提条件

- コントロールノードと管理対象ノードの準備が完了している。
- 管理対象ノードで Playbook を実行できるユーザーとしてコントロールノードにログインしている。
- 管理対象ノードへの接続に使用するアカウントに、そのノードに対する **sudo** 権限がある。
- **mlx4_ib0** という名前の InfiniBand デバイスが管理対象ノードにインストールされている。
- 管理対象ノードが NetworkManager を使用してネットワークを設定している。

手順

1. 次の内容を含む Playbook ファイル (例: ~/**playbook.yml**) を作成します。

```
---
- name: Configure the network
  hosts: managed-node-01.example.com
  tasks:
    - name: IPoIB connection profile with static IP address settings
      ansible.builtin.include_role:
        name: rhel-system-roles.network
      vars:
        network_connections:
          # InfiniBand connection mlx4_ib0
          - name: mlx4_ib0
            interface_name: mlx4_ib0
```

```
    type: infiniband

  # IPoIB device mlx4_ib0.8002 on top of mlx4_ib0
  - name: mlx4_ib0.8002
    type: infiniband
    autoconnect: yes
    infiniband:
      p_key: 0x8002
      transport_mode: datagram
    parent: mlx4_ib0
    ip:
      address:
        - 192.0.2.1/24
        - 2001:db8:1::1/64
    state: up
```

サンプル Playbook で指定されている設定は次のとおりです。

type: <profile_type>

作成するプロファイルのタイプを設定します。このサンプル Playbook では、2 つの接続プロファイルを作成します。1 つは InfiniBand 接続用、もう 1 つは IPoIB デバイス用です。

parent: <parent_device>

IPoIB 接続プロファイルの親デバイスを設定します。

p_key: <value>

InfiniBand パーティションキーを設定します。この変数を設定する場合は、IPoIB デバイスに **interface_name** を設定しないでください。

transport_mode: <mode>

IPoIB 接続の動作モードを設定します。この変数は、**datagram** (デフォルト) または **connected** に設定できます。

Playbook で使用されるすべての変数の詳細は、コントロールノードの **/usr/share/ansible/roles/rhel-system-roles.network/README.md** ファイルを参照してください。

2. Playbook の構文を検証します。

```
$ ansible-playbook --syntax-check ~/playbook.yml
```

このコマンドは構文を検証するだけであり、有効だが不適切な設定から保護するものではないことに注意してください。

3. Playbook を実行します。

```
$ ansible-playbook ~/playbook.yml
```

検証

1. **mlx4_ib0.8002** デバイスの IP 設定を表示します。

```
# ansible managed-node-01.example.com -m command -a 'ip address show
mlx4_ib0.8002'
managed-node-01.example.com | CHANGED | rc=0 >>
...
```

```
    inet 192.0.2.1/24 brd 192.0.2.255 scope global noprefixroute ib0.8002
       valid_lft forever preferred_lft forever
    inet6 2001:db8:1::1/64 scope link tentative noprefixroute
       valid_lft forever preferred_lft forever
```

2. **mlx4_ib0.8002** デバイスのパーティションキー (P_Key) を表示します。

```
# ansible managed-node-01.example.com -m command -a 'cat
/sys/class/net/mlx4_ib0.8002/pkey'
managed-node-01.example.com | CHANGED | rc=0 >>
0x8002
```

3. **mlx4_ib0.8002** デバイスのモードを表示します。

```
# ansible managed-node-01.example.com -m command -a 'cat
/sys/class/net/mlx4_ib0.8002/mode'
managed-node-01.example.com | CHANGED | rc=0 >>
datagram
```

関連情報

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** ファイル
- **/usr/share/doc/rhel-system-roles/network/** ディレクトリー

3.6. NM-CONNECTION-EDITOR を使用した IPOIB 接続の設定

nmcli-connection-editor アプリケーションは、管理コンソールを使用して、NetworkManager によって保存されたネットワーク接続を設定および管理します。

前提条件

- InfiniBand デバイスがサーバーに取り付けられている。
- 対応するカーネルモジュールがロードされている。
- **nm-connection-editor** パッケージがインストールされている。

手順

1. コマンドを入力します。

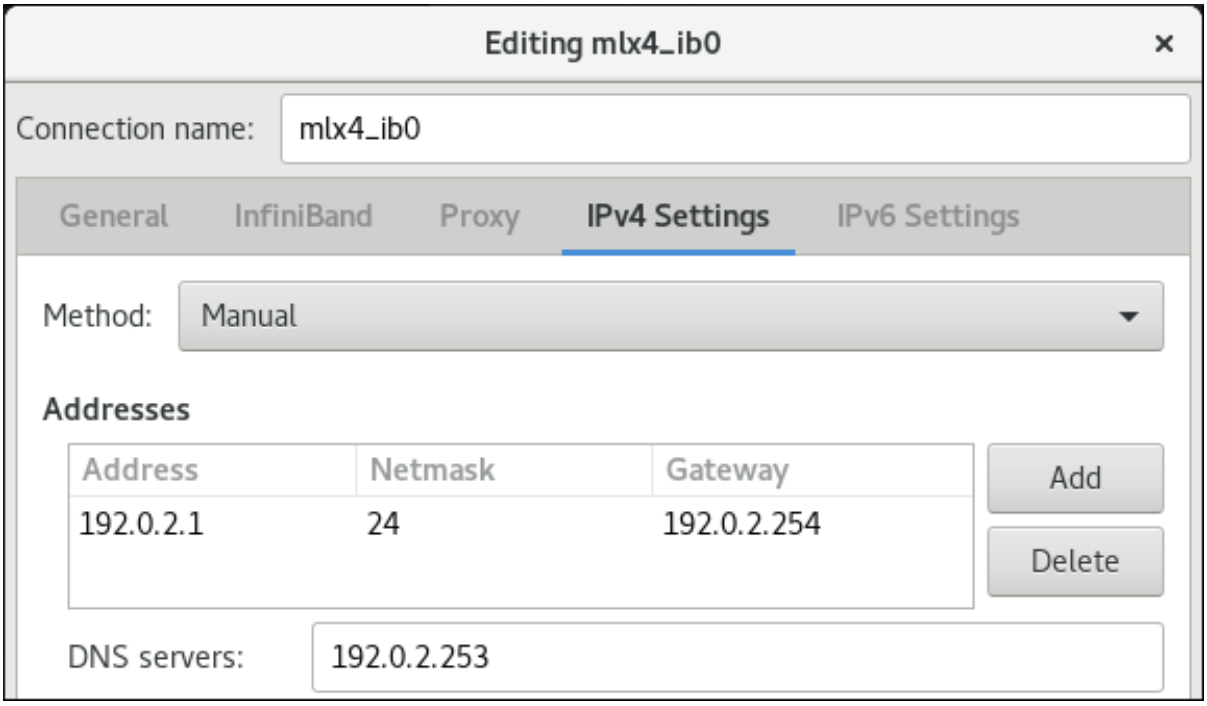
```
$ nm-connection-editor
```

2. **+** ボタンをクリックして、新しい接続を追加します。
3. **InfiniBand** 接続タイプを選択し、**Create** をクリックします。
4. **InfiniBand** タブで以下を行います。
   a. 必要に応じて、接続名を変更します。
   b. トランスポートモードを選択します。

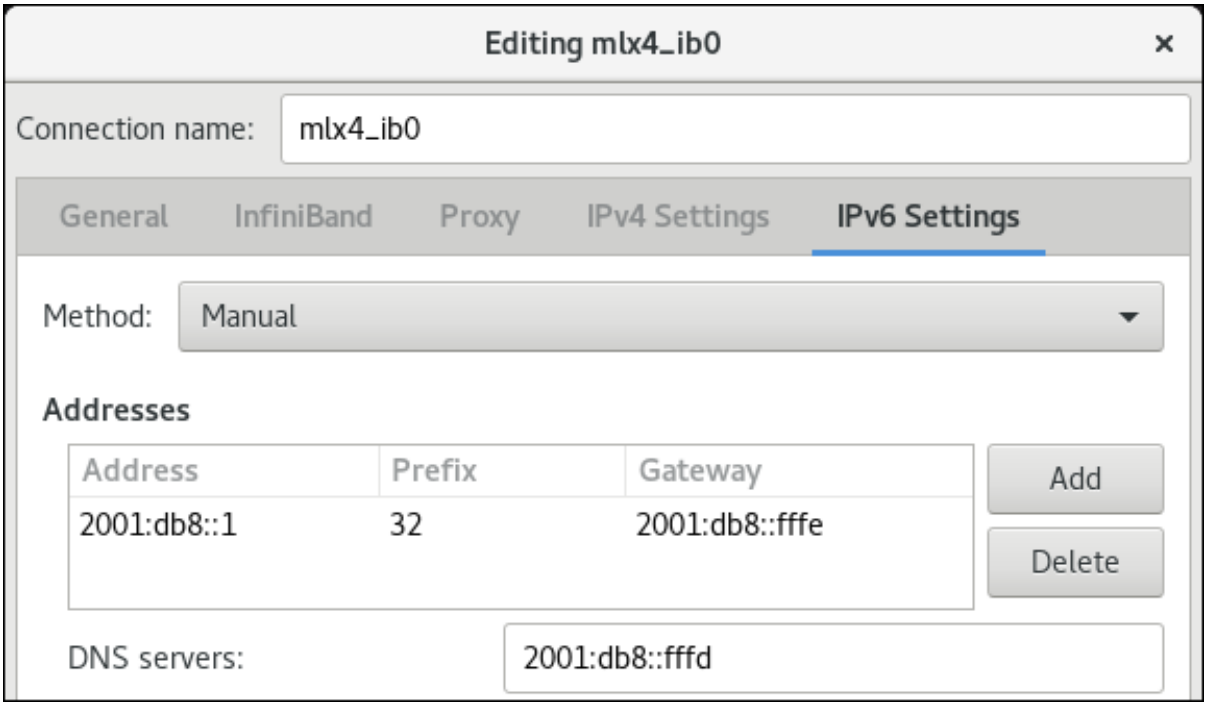
c. デバイスを選択します。

d. 必要に応じて MTU を設定します。

5. **IPv4 Settings** タブで、IPv4 設定を設定します。たとえば、静的な IPv4 アドレス、ネットワークマスク、デフォルトゲートウェイ、および DNS サーバーを設定します。



6. **IPv6 Settings** タブで、IPv6 設定を設定します。たとえば、静的な IPv6 アドレス、ネットワークマスク、デフォルトゲートウェイ、および DNS サーバーを設定します。



7. **Save** をクリックして、チーム接続を保存します。

8. **nm-connection-editor** を閉じます。

9. **P_Key** インターフェイスを設定することができます。この設定は **nm-connection-editor** では利用できないため、コマンドラインでこのパラメーターを設定する必要があります。

たとえば、**mlx4_ib0** 接続の **P_Key** インターフェイスとして **0x8002** を設定するには、以下のコマンドを実行します。

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

3.7. IPOIB の設定後に QPERF を使用した RDMA ネットワークのテスト

qperf ユーティリティーは、2 つのノード間の RDMA と IP のパフォーマンスを、帯域幅、レイテンシー、CPU 使用率の観点から測定します。

前提条件

- 両方のホストに **qperf** パッケージがインストールされている。
- IPoIB が両方のホストに設定されている。

手順

1. サーバーとして機能するオプションを指定せずに、いずれかのホストで **qperf** を起動します。

```
# qperf
```

2. クライアントで以下のコマンドを使用します。コマンドは、クライアントの **mlx4_0** ホストチャネルアダプターのポート **1** を使用して、サーバーの InfiniBand アダプターに割り当てられた IP アドレス **192.0.2.1** に接続します。

   a. ホストチャネルアダプターの設定を表示します。

```
# qperf -v -i mlx4_0:1 192.0.2.1 conf

conf:
    loc_node   =  rdma-dev-01.lab.bos.redhat.com
    loc_cpu    =  12 Cores: Mixed CPUs
    loc_os     =  Linux 4.18.0-187.el8.x86_64
    loc_qperf  =  0.4.11
    rem_node   =  rdma-dev-00.lab.bos.redhat.com
    rem_cpu    =  12 Cores: Mixed CPUs
    rem_os     =  Linux 4.18.0-187.el8.x86_64
    rem_qperf  =  0.4.11
```

   b. Reliable Connection (RC) ストリーミングの双方向帯域幅を表示します。

```
# qperf -v -i mlx4_0:1 192.0.2.1 rc_bi_bw

rc_bi_bw:
    bw              =  10.7 GB/sec
    msg_rate        =   163 K/sec
    loc_id          =  mlx4_0
    rem_id          =  mlx4_0:1
    loc_cpus_used   =    65 % cpus
    rem_cpus_used   =    62 % cpus
```

   c. RC ストリーミングの一方向帯域幅を表示します。

```
# qperf -v -i mlx4_0:1 192.0.2.1 rc_bw

rc_bw:
    bw              =  6.19 GB/sec
    msg_rate        =  94.4 K/sec
    loc_id          =  mlx4_0
    rem_id          =  mlx4_0:1
    send_cost       =  63.5 ms/GB
    recv_cost       =    63 ms/GB
    send_cpus_used  =  39.5 % cpus
    recv_cpus_used  =    39 % cpus
```

関連情報

- システムの **qperf (1)** man ページ

第4章 ROCE の設定

Remote Direct Memory Access (RDMA) は、直接メモリーアクセス (DMA) のリモート実行を提供します。RDMA over Converged Ethernet (RoCE) は、イーサネットネットワーク上で RDMA を利用するネットワークプロトコルです。RoCE の設定には特定のハードウェアが必要です。ハードウェアベンダーには Mellanox、Broadcom、QLogic などがあります。

4.1. ROCE プロトコルバージョンの概要

RoCE は、イーサネット上で Remote Direct Memory Access (RDMA) を有効にするネットワークプロトコルです。

以下は、RoCE のさまざまなバージョンです。

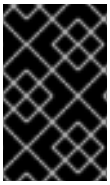
RoCE v1

RoCE バージョン 1 プロトコルは、イーサタイプ **0x8915** を持つイーサネットリンク層プロトコルです。同じイーサネットブロードキャストドメイン内にある 2 つのホスト間の通信を可能にします。

RoCE v2

RoCE バージョン 2 プロトコルは、UDP over IPv4 または UDP over IPv6 プロトコルの上位に存在します。RoCE v2 の場合、UDP の宛先ポート番号は **4791** です。

RDMA_CM は、データを転送するためにクライアントとサーバーとの間に信頼できる接続を確立します。RDMA_CM は、接続を確立するために RDMA トランスポートに依存しないインターフェイスを提供します。通信は、特定の RDMA デバイスとメッセージベースのデータ転送を使用します。



重要

クライアントで RoCE v2 を使用し、サーバーで RoCE v1 を使用するなど、異なるバージョンの使用はサポートされていません。この場合は、サーバーとクライアントの両方が RoCE v1 で通信するように設定します。

関連情報

- デフォルトの RoCE バージョンを一時的に変更する

4.2. デフォルトの ROCE バージョンを一時的に変更する

クライアントで RoCE v2 プロトコルを使用し、サーバーで RoCE v1 を使用することはサポートされていません。サーバーのハードウェアが RoCE v1 のみをサポートしている場合は、サーバーと通信できるようにクライアントを RoCE v1 用に設定します。たとえば、RoCE v1 のみをサポートする Mellanox ConnectX-5 InfiniBand デバイス用には、**mlx5_0** ドライバーを使用するクライアントを設定できます。

注記

ここで説明する変更は、ホストを再起動するまで有効です。

前提条件

- クライアントが、RoCE v2 プロトコルに対応した InfiniBand デバイスを使用している。
- サーバーが、RoCEv1 のみをサポートする InfiniBand デバイスを使用している。

手順

1. **/sys/kernel/config/rdma_cm/mlx5_0/** ディレクトリーを作成します。

```
# mkdir /sys/kernel/config/rdma_cm/mlx5_0/
```

2. デフォルトの RoCE モードを表示します。

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

RoCE v2
```

3. デフォルトの RoCE モードをバージョン 1 に変更します。

```
# echo "IB/RoCE v1" > /sys/kernel/config/rdma_cm/mlx5_0/ports/1/default_roce_mode
```

4.3. SOFT-ROCE の設定

Soft-RoCE は、イーサネット経由のリモートダイレクトメモリーアクセス (RDMA) のソフトウェア実装で、RXE とも呼ばれます。RoCE ホストチャンネルアダプター (HCA) のないホストで Soft-RoCE を使用します。

重要

Soft-RoCE 機能はテクノロジープレビューとしてのみ提供されます。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) ではサポートされておらず、機能的に完全ではない可能性があるため、Red Hat では実稼働環境での使用を推奨していません。テクノロジープレビュー機能では、最新の製品機能をいち早く提供します。これにより、お客様は開発段階で機能をテストし、フィードバックを提供できます。

テクノロジープレビュー機能のサポート範囲については、Red Hat カスタマーポータルの テクノロジープレビュー機能のサポート範囲 を参照してください。

前提条件

- イーサネットアダプターが搭載されている。

手順

1. **iproute** パッケージ、**libibverbs** パッケージ、**libibverbs-utils** パッケージ、および **infiniband-diags** パッケージをインストールします。

```
# yum install iproute libibverbs libibverbs-utils infiniband-diags
```

2. RDMA リンクを表示します。

```
# rdma link show
```

3. **rdma_rxe** カーネルモジュールをロードし、**enp0s1** インターフェイスを使用する **rxe0** という名前の新しい **rxe** デバイスを追加します。

```
# rdma link add rxe0 type rxe netdev enp1s0
```

-

検証

1. すべての RDMA リンクの状態を表示します。

```
# rdma link show

link rxe0/1 state ACTIVE physical_state LINK_UP netdev enp1s0
```

2. 利用可能な RDMA デバイスをリスト表示します。

```
# ibv_devices

    device                 node GUID
    ------              ----------------
    rxe0                505400fffed5e0fb
```

3. **ibstat** ユーティリティーを使用して詳細なステータスを表示することができます。

```
# ibstat rxe0

CA 'rxe0'
 CA type:
 Number of ports: 1
 Firmware version:
 Hardware version:
 Node GUID: 0x505400fffed5e0fb
 System image GUID: 0x0000000000000000
 Port 1:
  State: Active
  Physical state: LinkUp
  Rate: 100
  Base lid: 0
  LMC: 0
  SM lid: 0
  Capability mask: 0x00890000
  Port GUID: 0x505400fffed5e0fb
  Link layer: Ethernet
```

第5章 システムでユーザーがピニング (固定) できるメモリーの量を増やす

Remote Direct Memory Access (RDMA) の操作には、物理メモリーのピニングが必要です。これにより、カーネルがスワップ領域にメモリーを書き込むことができなくなります。ユーザーがメモリーを過剰に固定すると、システムのメモリーが不足し、カーネルがプロセスを終了してより多くのメモリーを解放することがあります。したがって、メモリーのピニングは特権が必要な操作です。

root 以外のユーザーが大規模な RDMA アプリケーションを実行する必要がある場合は、プライマリーメモリー内のページを常にピニングしておくために、メモリーの量を増やす必要があります。

手順

- **root** ユーザーで、**/etc/security/limits.conf** ファイルを以下の内容で作成します。

```
@rdma soft memlock unlimited
@rdma hard memlock unlimited
```

検証

1. **/etc/security/limits.conf** ファイルの編集後、**rdma** グループのメンバーとしてログインします。
   Red Hat Enterprise Linux は、ユーザーのログイン時に、更新された **ulimit** の設定を適用することに注意してください。

2. **ulimit -l** コマンドを使用して制限を表示します。

```
$ ulimit -l
unlimited
```

   コマンドが **unlimited** を返す場合、ユーザーはメモリーのピニングを無制限に行うことができます。

関連情報

- システム上の **limits.conf (5)** man ページ

第6章 NFS サーバーで NFS OVER RDMA を有効にする

Remote Direct Memory Access (RDMA) は、クライアントシステムがストレージサーバーのメモリーから自身のメモリーにデータを直接転送できるようにするプロトコルです。これにより、ストレージのスループットが向上し、サーバーとクライアント間のデータ転送の遅延が減少し、両側の CPU 負荷が軽減されます。NFS サーバーとクライアントの両方が RDMA 経由で接続されている場合、クライアントは NFS over RDMA (NFSoRDMA) を使用してエクスポートされたディレクトリーをマウントできます。

前提条件

- NFS サービスが実行および設定されている。
- InfiniBand または RDMA over Converged Ethernet (RoCE) デバイスがサーバーにインストールされている。
- サーバーに IP over InfiniBand (IPoIB) が設定され、InfiniBand デバイスに IP アドレスが割り当てられている。

手順

1. **rdma-core** パッケージをインストールします。

```
# dnf install rdma-core
```

2. パッケージがすでにインストールされている場合は、**/etc/rdma/modules/rdma.conf** ファイル内の **xprtrdma** および **svcrdma** モジュールのコメントが解除されていることを確認します。

```
# NFS over RDMA client support
xprtrdma
# NFS over RDMA server support
svcrdma
```

3. オプション: デフォルトでは、NFS over RDMA はポート 20049 を使用します。別のポートを使用する場合は、**/etc/nfs.conf** ファイルの **[nfsd]** セクションで **rdma-port** 設定を指定します。

```
rdma-port=<port>
```

4. **firewalld** で NFSoRDMA ポートを開きます。

```
# firewall-cmd --permanent --add-port={20049/tcp,20049/udp}
# firewall-cmd --reload
```

   20049 以外のポートを設定する場合は、ポート番号を変更します。

5. **nfs-server** サービスを再起動します。

```
# systemctl restart nfs-server
```

検証

1. InfiniBand ハードウェアを搭載したクライアントで、次の手順を実行します。
   a. 以下のパッケージをインストールします。
      -

```
# dnf install nfs-utils rdma-core
```

b. エクスポートされた NFS 共有を RDMA 経由でマウントします。

```
# mount -o rdma server.example.com:/nfs/projects/ /mnt/
```

デフォルト (20049) 以外のポート番号を設定する場合は、コマンドに **port=<port_number>** を渡します。

```
# mount -o rdma,port=<port_number> server.example.com:/nfs/projects/ /mnt/
```

c. **rdma** オプションを使用して共有がマウントされたことを確認します。

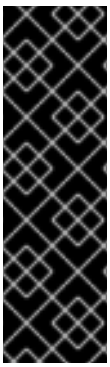
```
# mount | grep "/mnt"
server.example.com:/nfs/projects/ on /mnt type nfs (...,proto=rdma,...)
```

関連情報

- InfiniBand ネットワークおよび RDMA ネットワークの設定

第7章 SOFT-IWARP の設定

Remote Direct Memory Access (RDMA) は、パフォーマンス向上と補助プログラミングインターフェイスのために、iWARP、Soft-iWARP など、いくつかのライブラリーとプロトコルをイーサネット上で使用します。



重要

Soft-iWARP はテクノロジープレビュー機能です。テクノロジープレビュー機能は、Red Hat 製品のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat では、実稼働環境での使用を推奨していません。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。Red Hat のテクノロジープレビュー機能のサポート範囲についての詳細は、https://access.redhat.com/ja/support/offerings/techpreview/ を参照してください。

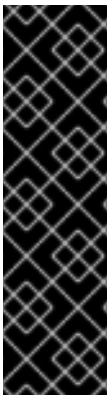
7.1. IWARP と SOFT-IWARP の概要

Remote Direct Memory Access (RDMA) は、イーサネットを介した iWARP を使用して、TCP 経由で集中型の低レイテンシーのデータ送信を行います。iWARP は、標準のイーサネットスイッチと TCP/IP スタックを使用して、IP サブネット間でトラフィックをルーティングします。これにより、既存のインフラストラクチャーを効率的に使用するための柔軟性が提供されます。Red Hat Enterprise Linux では、複数のプロバイダーがハードウェアネットワークインターフェイスカードに iWARP を実装しています。たとえば、**cxgb4**、**irdma**、**qedr** などです。

Soft-iWARP (siw) は、Linux 用のソフトウェアベースの iWARP カーネルドライバーおよびユーザーライブラリーです。これはソフトウェアベースの RDMA デバイスであり、ネットワークインターフェイスカードに接続すると、RDMA ハードウェアにプログラミングインターフェイスを提供します。Soft-iWARP は、RDMA 環境をテストおよび検証する簡単な方法を提供します。

7.2. SOFT-IWARP の設定

Soft-iWARP (siw) は、Linux TCP/IP ネットワークスタックを介して iWARP Remote Direct Memory Access (RDMA) トランスポートを実装します。これにより、標準のイーサネットアダプターを備えたシステムが、iWARP アダプター、または Soft-iWARP ドライバーを実行している別のシステム、または iWARP をサポートするハードウェアを備えたホストと相互運用できるようになります。



重要

Soft-iWARP 機能は、テクノロジープレビューとしてのみ提供されます。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) ではサポートされておらず、機能的に完全ではない可能性があるため、Red Hat では実稼働環境での使用を推奨していません。テクノロジープレビュー機能では、最新の製品機能をいち早く提供します。これにより、お客様は開発段階で機能をテストし、フィードバックを提供できます。

テクノロジープレビュー機能のサポート範囲については、Red Hat カスタマーポータルの テクノロジープレビュー機能のサポート範囲 を参照してください。

Soft-iWARP を設定する際には、スクリプトで次の手順を使用して、システムの起動時に自動的にスクリプトを実行できます。

前提条件

- イーサネットアダプターが搭載されている。

手順

1. **iproute** パッケージ、**libibverbs** パッケージ、**libibverbs-utils** パッケージ、および **infiniband-diags** パッケージをインストールします。

```
# yum install iproute libibverbs libibverbs-utils infiniband-diags
```

2. RDMA リンクを表示します。

```
# rdma link show
```

3. **siw** カーネルモジュールをロードします。

```
# modprobe siw
```

4. **enp0s1** インターフェイスを使用する、**siw0** という名前の新しい **siw** デバイスを追加します。

```
# rdma link add siw0 type siw netdev enp0s1
```

検証

1. すべての RDMA リンクの状態を表示します。

```
# rdma link show

link siw0/1 state ACTIVE physical_state LINK_UP netdev enp0s1
```

2. 利用可能な RDMA デバイスをリスト表示します。

```
# ibv_devices

 device                 node GUID
 ------              ----------------
 siw0                0250b6fffea19d61
```

3. **ibv_devinfo** ユーティリティーを使用して、詳細なステータスを表示することができます。

```
# ibv_devinfo siw0

    hca_id:               siw0
    transport:            iWARP (1)
    fw_ver:               0.0.0
    node_guid:            0250:b6ff:fea1:9d61
    sys_image_guid:       0250:b6ff:fea1:9d61
    vendor_id:            0x626d74
    vendor_part_id:       1
    hw_ver:               0x0
    phys_port_cnt:        1
        port:             1
```

```
state:          PORT_ACTIVE (4)
max_mtu:        1024 (3)
active_mtu:     1024 (3)
sm_lid:         0
port_lid:       0
port_lmc:       0x00
link_layer:     Ethernet
```

第8章 INFINIBAND サブネットマネージャー

すべての InfiniBand ネットワークでは、ネットワークが機能するために、サブネットマネージャーが実行されている必要があります。これは、2 台のマシンがスイッチなしで直接接続されている場合にも当てはまります。

複数のサブネットマネージャーを使用することもできます。その場合、1 つのサブネットマネージャーはマスターとして、もう 1 つはスレーブとして機能します。スレーブは、マスターサブネットマネージャーに障害が発生した場合に引き継ぎます。

Red Hat Enterprise Linux は、InfiniBand サブネットマネージャーの実装である **OpenSM** を提供します。ただし、**OpenSM** は機能が限られており、アップストリームの開発が活発に行われていません。通常、InfiniBand スイッチに組み込まれているサブネットマネージャーのほうが、より多くの機能を備えており、最新の InfiniBand ハードウェアをサポートしています。詳細は、OpenSM InfiniBand サブネットマネージャーのインストールと設定を 参照してください。