Red Hat Enterprise Linux 9

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: 2024-06-26

Red Hat Enterprise Linux 9 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

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) 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 を使用すると、データは、高スループット、低レイテンシー、低 CPU 使用率で転送されます。

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 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。

2.1. IWARP と SOFT-IWARP の概要

Remote Direct Memory Access (RDMA) は、イーサネットを介したインターネットワイドエリア RDMA プロトコル (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。

2.2. 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 Internet Wide-area RDMA Protocol (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。

```
# dnf 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3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。

3.1. 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。

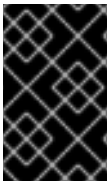
RoCE v1

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。このような場合は、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てください。

RoCE v1 はデータリンク層 (Layer 2) で動作し、同じネットワーク内の 2 台のマシンの通信のみをサポートします。デフォルトでは、RoCE v2 を使用できます。これは、ネットワーク層 (Layer 3) で機能します。RoCE v2 は、複数のイーサネットとの接続を提供するパケットルーティングをサポートします。

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

3.2. 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
```

第4章 コア RDMA サブシステムの設定

rdma サービス設定は、InfiniBand、iWARP、RoCE などのネットワークプロトコルと通信標準を管理します。

4.1. SYSTEMD リンクファイルを使用して IPOIB デバイスの名前を変更する

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。競合を回避するには、**systemd** リンクファイルを作成して、**mlx4_ib0** など、永続的でわかりやすい名前を作成します。

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 addr show ib0

7: ib0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 65520 qdisc fq_codel state UP
group default qlen 256
    link/infiniband 80:00:0a:28:fe:80:00:00:00:00:00:00:f4:52:14:03:00:7b:e1:b1 brd
00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff
    altname ibp7s0
    altname ibs2
    inet 172.31.0.181/24 brd 172.31.0.255 scope global dynamic noprefixroute ib0
       valid_lft 2899sec preferred_lft 2899sec
    inet6 fe80::f652:1403:7b:e1b1/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

2. **80:00:0a:28:fe:80:00:00:00:00:00:00:f4:52:14:03:00:7b:e1:b1** という MAC アドレスを持つインターフェイスに **mlx4_ib0** という名前を付けるには、**/etc/systemd/network/70-custom-ifnames.link** ファイルを次の内容で作成します。

```
[Match]
MACAddress=80:00:0a:28:fe:80:00:00:00:00:00:00:f4:52:14:03:00:7b:e1:b1

[Link]
Name=mlx4_ib0
```

このリンクファイルによって MAC アドレスが照合され、ネットワークインターフェイスの名前が **Name** パラメーターに設定された名前に変更されます。

検証

1. ホストを再起動します。

```
# reboot
```

2. リンクファイルで指定した MAC アドレスを持つデバイスが **mlx4_ib0** に割り当てられていることを確認します。

```
# ip addr show mlx4_ib0

7: mlx4_ib0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 65520 qdisc fq_codel state
UP group default qlen 256
    link/infiniband 80:00:0a:28:fe:80:00:00:00:00:00:00:f4:52:14:03:00:7b:e1:b1 brd
00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff
    altname ibp7s0
    altname ibs2
    inet 172.31.0.181/24 brd 172.31.0.255 scope global dynamic noprefixroute mlx4_ib0
       valid_lft 2899sec preferred_lft 2899sec
    inet6 fe80::f652:1403:7b:e1b1/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

関連情報

- **systemd.link(5)** man ページ

4.2. 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

- **limits.conf(5)** man ページ

4.3. 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5章 INFINIBAND サブネットマネージャーの設定

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。

ほとんどの InfiniBand スイッチには、サブネットマネージャーが組み込まれています。ただし、最新のサブネットマネージャーが必要な場合や、より詳細な制御が必要な場合は、Red Hat Enterprise Linux が提供する **OpenSM** サブネットマネージャーを使用してください。

詳細は、OpenSM サブネットマネージャーのインストール を参照してください。

第6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。

6.1. 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る場合、InfiniBand スイッチおよびファブリックは **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 **Datagram** モードを使用してマルチキャストトラフィックを引き続き送信します。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。

6.2. IPOIB ハードウェアアドレスについて

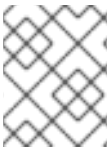
IPoIB デバイスには、以下の部分で構成される **20** バイトのハードウェアアドレスがあります。

- 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。

6.3. 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. また、**mlx4_ib0** 接続の **P_Key** インターフェイスとして **0x8002** を設定することも可能です。

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

3. IPv4 を設定するには、**mlx4_ib0** 接続の静的 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、および DNS サーバーを設定します。

```
# 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. IPv6 を設定するには、**mlx4_ib0** 接続の静的 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、および DNS サーバーを設定します。

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

5. **mlx4_ib0** 接続をアクティブ化するには、以下を実行します。

```
# nmcli connection up mlx4_ib0
```

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

network RHEL システムロールを使用して、IP over InfiniBand (IPoIB) デバイスの NetworkManager 接続プロファイルをリモートで作成できます。たとえば、Ansible Playbook を実行して、次の設定で **mlx4_ib0** インターフェイスの InfiniBand 接続をリモートで追加します。

- IPoIB デバイス - **mlx4_ib0.8002**
- パーティションキー **p_key** - **0x8002**
- 静的 **IPv4** アドレス - **192.0.2.1** と **/24** サブネットマスク
- 静的 **IPv6** アドレス - **2001:db8:1::1** と **/64** サブネットマスク

Ansible コントロールノードで以下の手順を実行します。

前提条件

- 制御ノードと管理ノードを準備している
- 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は、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: Configure IPoIB
      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
```

この例のように **p_key** パラメーターを設定する場合は、IPoIB デバイスで **interface_name** パラメーターを設定しないでください。

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. **managed-node-01.example.com** ホストで、**mlx4_ib0.8002** デバイスの IP 設定を表示します。

```
# ip address show mlx4_ib0.8002
...
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。

```
# cat /sys/class/net/mlx4_ib0.8002/pkey
0x8002
```

3. **mlx4_ib0.8002** デバイスのモードを表示します。

```
# cat /sys/class/net/mlx4_ib0.8002/mode
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

6.5. 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

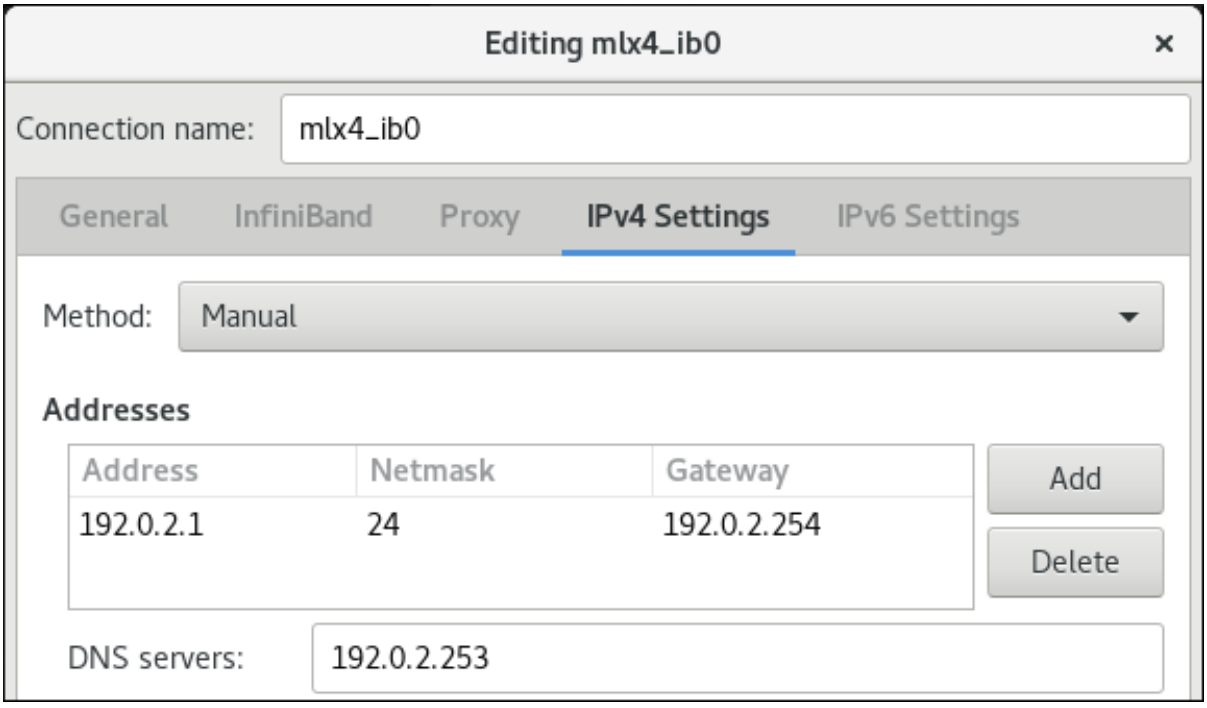
- InfiniBand デバイスがサーバーに取り付けられている。
- 対応するカーネルモジュールがロードされている。
- **nm-connection-editor** パッケージがインストールされている。

手順

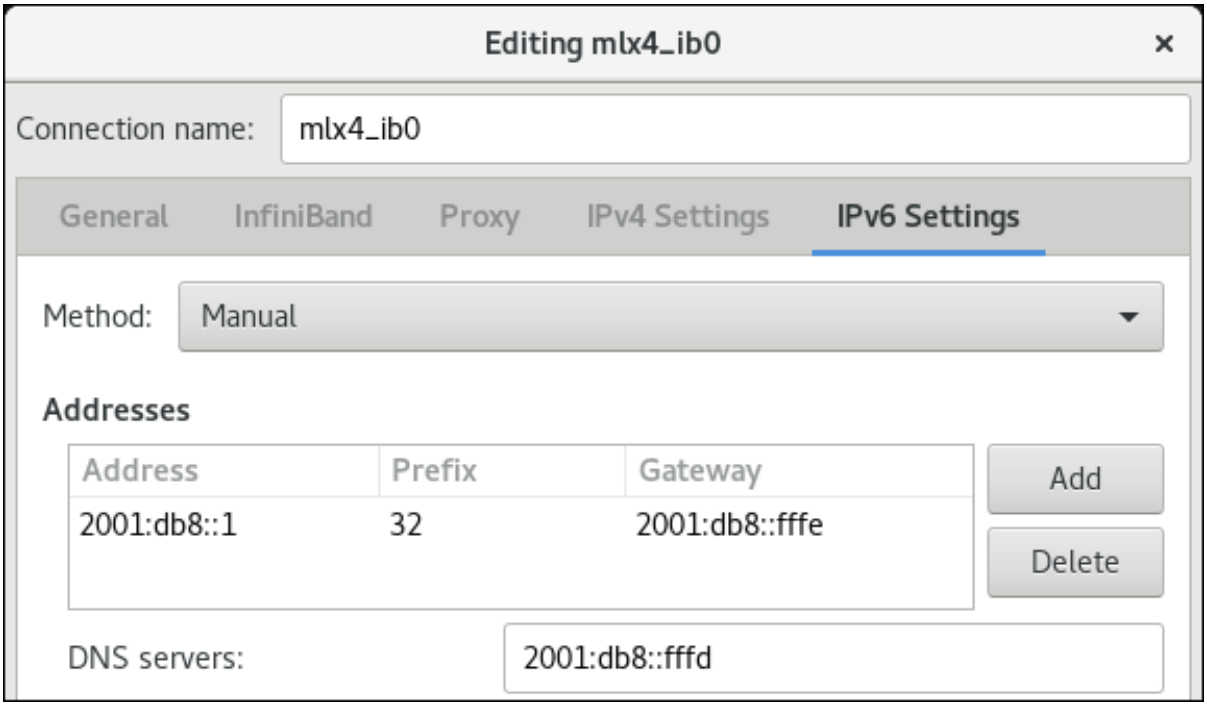
1. コマンドを入力します。

```
$ 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
```

第7章 INFINIBAND ネットワークのテスト

7.1. 初期の INFINIBAND RDMA 操作のテスト

InfiniBand は、Remote Direct Memory Access (RDMA) に低レイテンシーと高パフォーマンスを提供します。

注記

InfiniBand とは別に、Internet Wide-area Remote Protocol (iWARP)、RDMA over Converged Ethernet (RoCE)、または InfiniBand over Ethernet (IBoE) デバイスなどの IP ベースのデバイスを使用する場合は、次を参照してください。

- ping ユーティリティーを使用した IPoIB のテスト
- IPoIB の設定後に iperf3 を使用して RDMA ネットワークをテストする

前提条件

- **rdma** サービスが設定されている。
- **libibverbs-utils** パッケージと **infiniband-diags** パッケージがインストールされている。

手順

1. 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
```

2. **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
```

3. **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
```

4. **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、ホストで **-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
```

関連情報

- **ibping(8)** man ページ

7.2. PING ユーティリティーを使用した IPOIB のテスト

IP over InfiniBand (IPoIB) を設定したら、**ping** ユーティリティーを使用して ICMP パケットを送信し、IPoIB 接続をテストします。

前提条件

- 2 台の RDMA ホストが、同じ InfiniBand ファブリックに RDMA ポートで接続されている。
- 両方のホストの IPoIB インターフェイスが、同じサブネット内の IP アドレスで設定されている。

手順

- **ping** ユーティリティーを使用して、5 つの 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。

```
# ping -c5 192.0.2.1
```

7.3. IPOIB の設定後に IPERF3 を使用して RDMA ネットワークをテストする

次の例では、大きなバッファーサイズを使用して、60 秒のテストを実行し、最大スループットを測定します。**iperf3** ユーティリティーを使用して 2 つのホスト間の帯域幅とレイテンシーを十分に使用します。

前提条件

- 両方のホストで IPoIB を設定している。

手順

1. **iperf3** をシステム上のサーバーとして実行するには、時間の間隔を定義して、定期的な帯域幅の更新 **-i** を指定し、クライアント接続の応答を待機するサーバー **-s** としてリッスンします。

```
# iperf3 -i 5 -s
```

2. **iperf3** を別のシステムでクライアントとして実行するには、時間の間隔を定義して、定期的な帯域幅の更新 **-i** を指定し、**-t** 秒使用して IP アドレス **192.168.2.2** のリスニングサーバー **-c** に接続します。

```
# iperf3 -i 5 -t 60 -c 192.168.2.2
```

3. 以下のコマンドを使用します。

   a. サーバーとして動作するシステムでテスト結果を表示します。

```
# iperf3 -i 10 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.2.3, port 22216
[5] local 192.168.2.2 port 5201 connected to 192.168.2.3 port 22218
[ID] Interval           Transfer     Bandwidth
[5]   0.00-10.00  sec  17.5 GBytes  15.0 Gbits/sec
[5]  10.00-20.00  sec  17.6 GBytes  15.2 Gbits/sec
[5]  20.00-30.00  sec  18.4 GBytes  15.8 Gbits/sec
[5]  30.00-40.00  sec  18.0 GBytes  15.5 Gbits/sec
[5]  40.00-50.00  sec  17.5 GBytes  15.1 Gbits/sec
[5]  50.00-60.00  sec  18.1 GBytes  15.5 Gbits/sec
[5]  60.00-60.04  sec  82.2 MBytes  17.3 Gbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ID] Interval           Transfer     Bandwidth
[5]   0.00-60.04  sec  0.00 Bytes    0.00 bits/sec  sender
[5]   0.00-60.04  sec   107 GBytes  15.3 Gbits/sec  receiver
```

b. クライアントとして動作するシステムでテスト結果を表示します。

```
# iperf3 -i 1 -t 60 -c 192.168.2.2

Connecting to host 192.168.2.2, port 5201
[4] local 192.168.2.3 port 22218 connected to 192.168.2.2 port 5201
[ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[4]   0.00-10.00  sec  17.6 GBytes  15.1 Gbits/sec    0   6.01 MBytes
[4]  10.00-20.00  sec  17.6 GBytes  15.1 Gbits/sec    0   6.01 MBytes
[4]  20.00-30.00  sec  18.4 GBytes  15.8 Gbits/sec    0   6.01 MBytes
[4]  30.00-40.00  sec  18.0 GBytes  15.5 Gbits/sec    0   6.01 MBytes
[4]  40.00-50.00  sec  17.5 GBytes  15.1 Gbits/sec    0   6.01 MBytes
[4]  50.00-60.00  sec  18.1 GBytes  15.5 Gbits/sec    0   6.01 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ID] Interval           Transfer     Bandwidth       Retr
[4]   0.00-60.00  sec   107 GBytes  15.4 Gbits/sec    0   sender
[4]   0.00-60.00  sec   107 GBytes  15.4 Gbits/sec        receiver
```

関連情報

- **iperf3** man ページ