Red Hat Enterprise Linux 8

InfiniBand 및 RDMA 네트워크 구성

Red Hat Enterprise Linux 8에서 InfiniBand 및 RDMA 네트워크 구성 가이드

Last Updated: 2022-07-18

Red Hat Enterprise Linux 8 InfiniBand 및 RDMA 네트워크 구성

Red Hat Enterprise Linux 8에서 InfiniBand 및 RDMA 네트워크 구성 가이드

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

법적 공지

초록

이 문서에서는 InfiniBand 및 RDMA(원격 직접 메모리 액세스)의 개념과 InfiniBand 하드웨어를 구성하는 방법을 설명합니다. 또한 이 문서에서는 InfiniBand 관련 서비스를 구성하는 방법을 설명합니다.

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 CTO Chris Wright의 메시지 를 참조하십시오.

RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견을 보내 주십시오. Red Hat이 어떻게 이를 개선할 수 있는지 알려 주십시오.

- 특정 문구에 대한 간단한 주석은 다음과 같습니다.
  1. 문서가 *Multi-page HTML* 형식으로 표시되는지 확인합니다. 또한 문서 오른쪽 상단에 **Feedback** (피드백) 버튼이 있는지 확인합니다.
  2. 마우스 커서를 사용하여 주석 처리하려는 텍스트 부분을 강조 표시합니다.
  3. 강조 표시된 텍스트 아래에 표시되는 **피드백 추가** 팝업을 클릭합니다.
  4. 표시된 지침을 따릅니다.
- Bugzilla를 통해 피드백을 제출하려면 새 티켓을 생성합니다.
  1. Bugzilla 웹 사이트로 이동하십시오.
  2. Component로 **Documentation**을 선택하십시오.
  3. **Description** 필드에 문서 개선을 위한 제안 사항을 기입하십시오. 관련된 문서의 해당 부분 링크를 알려주십시오.
  4. **Submit Bug**를 클릭하십시오.

1장. INFINIBAND 및 RDMA 이해

InfiniBand는 다음 두 가지를 참조합니다.

- InfiniBand 네트워크의 물리적 링크 계층 프로토콜
- InfiniBand Verbs API - 원격 직접 메모리 액세스(RDMA) 기술 구현

RDMA는 운영 체제, 캐시 또는 스토리지를 포함하지 않고 두 컴퓨터의 기본 메모리 간의 액세스를 제공합니다. RDMA를 사용하여 높은 처리량, 짧은 대기 시간이 짧고 CPU 사용률이 낮은 데이터 전송.

일반적인 IP 데이터 전송에서는 한 시스템의 애플리케이션이 다른 머신의 애플리케이션에 데이터를 보내면 수신 끝에서 다음 작업이 수행됩니다.

1. 커널이 데이터를 수신해야 합니다.
2. 커널은 데이터가 애플리케이션에 속하는지 결정해야 합니다.
3. 커널이 애플리케이션을 활성화합니다.
4. 커널은 애플리케이션이 커널에 시스템 호출을 수행할 때까지 기다립니다.
5. 애플리케이션은 커널의 내부 메모리 공간에서 애플리케이션에서 제공하는 버퍼로 데이터를 복사합니다.

이 프로세스는 호스트 어댑터가 직접 메모리 액세스(DMA)를 사용하거나 두 번 이상 사용하는 경우 대부분의 네트워크 트래픽이 시스템의 기본 메모리에 복사됩니다. 또한 컴퓨터는 일부 컨텍스트 스위치를 실행하여 커널과 애플리케이션 간에 전환합니다. 이러한 컨텍스트 스위치는 다른 작업이 느려지는 동안 트래픽 속도가 높은 CPU 부하를 증가시킬 수 있습니다.

기존 IP 통신과 달리 RDMA 통신은 통신 프로세스의 커널 개입을 우회합니다. 이렇게 하면 CPU 오버헤드가 줄어듭니다. RDMA 프로토콜을 사용하면 호스트 어댑터가 패킷이 수신되는 네트워크와 해당 애플리케이션의 메모리 공간에 저장할 위치를 결정할 수 있습니다. 호스트 어댑터는 커널에 처리할 패킷을 보내고 사용자 애플리케이션의 메모리에 복사하는 대신 패킷 콘텐츠를 애플리케이션 버퍼에 직접 배치합니다. 이 프로세스에는 별도의 API, InfiniBand Verbs API가 필요하며 애플리케이션은 RDMA를 사용하려면 InfiniBand Verbs API를 구현해야 합니다.

Red Hat Enterprise Linux는 InfiniBand 하드웨어와 InfiniBand Verbs API를 모두 지원합니다. 또한 InfiniBand 비 하드웨어에서 InfiniBand Verbs API를 사용하도록 다음 기술을 지원합니다.

- iWARP(Internet Wide Area RDMA Protocol): IP 네트워크를 통해 RDMA를 구현하는 네트워크 프로토콜
- RoCE(RDMA over Converged Ethernet)를 IBoE(InfiniBand over Ethernet)라고도 합니다. 이더넷 네트워크를 통해 RDMA를 구현하는 네트워크 프로토콜

추가 리소스

- RoCE 구성

2장. ROCE 구성

이 섹션에서는 RDMA over Converged Ethernet (RoCE)에 대한 배경 정보와 기본 RoCE 버전을 변경하는 방법에 대해 설명합니다. 또한 소프트웨어 RoCE 어댑터를 구성하는 방법

RoCE 하드웨어를 제공하는 Mellanox, Broadcom 및 QLogic과 같은 다양한 벤더가 있습니다.

2.1. ROCE 프로토콜 버전 개요

RoCE는 이더넷을 통해 원격 직접 메모리 액세스(RDMA)를 활성화하는 네트워크 프로토콜입니다.

다음은 다른 RoCE 버전입니다.

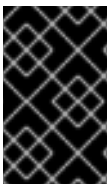
RoCE v1

RoCE 버전 1 프로토콜은 ethertype **0x8915** 를 사용하는 이더넷 링크 계층 프로토콜로, 동일한 이더넷 브로드캐스트 도메인에서 두 호스트 간의 통신을 가능하게 합니다.

RoCE v2

RoCE 버전 2 프로토콜은 UDP over IPv4 또는 UDP over IPv6 프로토콜에 존재합니다. RoCE v2의 경우 UDP 대상 포트 번호는 **4791** 입니다.

RDMA_CM은 데이터를 전송하는 클라이언트와 서버 간에 안정적인 연결을 설정합니다. RDMA_CM은 연결 설정을 위한 RDMA 전송 중립 인터페이스를 제공합니다. 통신은 특정 RDMA 장치 및 메시지 기반 데이터 전송을 사용합니다.



중요

클라이언트에서 RoCE v2와 같은 다른 버전을 사용하고 서버에서 RoCE v1을 사용하는 것은 지원되지 않습니다. 이러한 경우 RoCE v1에서 통신하도록 서버와 클라이언트를 구성합니다.

추가 리소스

- 기본 RoCE 버전 임시 변경

2.2. 기본 ROCE 버전 일시적으로 변경

서버에서 클라이언트 및 RoCE v1에서 RoCE v2 프로토콜을 사용하는 것은 지원되지 않습니다. 서버의 하드웨어가 RoCE v1만 지원하면 RoCE v1을 사용하여 서버와 통신하도록 클라이언트를 구성합니다. 이 섹션에서는 Mellanox ConnectX-5 Infiniband 장치에 **mlx5_0** 드라이버를 사용하는 클라이언트에 RoCE v1을 적용하는 방법을 설명합니다.

이 섹션에 설명된 변경 사항은 호스트를 재부팅할 때까지 일시적일 뿐입니다.

사전 요구 사항

- 클라이언트는 RoCE v2 프로토콜에서 InfiniBand 장치를 사용합니다.
- 서버는 RoCE v1만 지원하는 InfiniBand 장치를 사용합니다.

절차

1. **/sys/kernel/config/rdma_cm/*mlx5_0*/** 디렉터리를 만듭니다.

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

2. 기본 RoCE 모드를 표시합니다.

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

RoCE v2
```

3. 기본 RoCE 모드를 버전 1로 변경합니다.

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

2.3. 소프트 로빈 구성

소프트 로빈은 이더넷을 통한 RDMA(Remote Direct Memory Access) 소프트웨어 구현으로 RXE라고도 합니다. RoCE HCA(호스트 채널 어댑터)가 없는 호스트에서 소프트 로빈을 사용합니다.

중요

field-RoCE 기능은 기술 프리뷰로만 제공됩니다. 기술 프리뷰 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않을 수 있으며, 기능적으로 완전하지 않을 수 있으며 Red Hat은 프로덕션에 이러한 기능을 사용하지 않는 것이 좋습니다. 이러한 프리뷰를 통해 향후 제품 기능에 조기 액세스할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 Red Hat 고객 포털의 기술 프리뷰 기능 지원 범위를 참조하십시오.

사전 요구 사항

- 이더넷 어댑터가 설치됨

절차

1. **iproute**,**libibverbs -utils**, **infiniband-diags** 패키지를 설치합니다.

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

2. RDMA 링크를 표시합니다.

```
# rdma link show
```

3. **rdma_rxe** 커널 모듈을 로드하고 **enp0s1** 인터페이스를 사용하는 **rxe 0** 이라는 새로운 rxe 장치를 추가합니다.

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

검증

1. 모든 RDMA 링크의 상태를 표시합니다.

```
# rdma link show

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

2. 사용 가능한 RDMA 장치를 나열합니다.

```
# ibv_devices

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

3. **ibstat** 유틸리티를 사용하여 자세한 상태를 표시할 수 있습니다.

```
# 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
```

3장. SOFT-IWARP 구성

이 섹션에서는 iWARP, soft-iWARP 및 soft-iWARP 구성에 대한 배경 정보를 설명합니다.

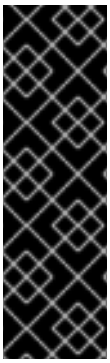
3.1. IWARP 및 SOFT-IWARP에 대한 개요

RDMA(Remote Direct Memory Access)는 TCP를 통한 통합 및 짧은 대기 시간 데이터 전송을 위해 이더넷을 통한 Internet Wide-area RDMA 프로토콜(iWARP)을 사용합니다. 표준 이더넷 스위치와 TCP/IP 스택을 사용하여 iWARP는 IP 서브넷 전체에 트래픽을 라우팅합니다. 이를 통해 기존 인프라를 효율적으로 사용할 수 있는 유연성을 제공합니다. Red Hat Enterprise Linux에서 여러 공급업체는 하드웨어 네트워크 인터페이스 카드에 iWARP를 구현합니다. 예를 들어 **cxgb4**,**irdma**,**qedr** 등입니다.

soft-iWARP(siw)는 Linux용 소프트웨어 기반 iWARP 커널 드라이버 및 사용자 라이브러리입니다. 네트워크 인터페이스 카드에 연결된 경우 RDMA 하드웨어에 프로그래밍 인터페이스를 제공하는 소프트웨어 기반 RDMA 장치입니다. RDMA 환경을 쉽게 테스트하고 검증할 수 있습니다.

3.2. SOFT-IWARP 구성

soft-iWARP(siw)는 Linux TCP/IP 네트워크 스택을 통해 RDP(Internet Wide-area RDMA Protocol) RDMA(Remote Direct Memory Access) 전송을 구현합니다. 표준 이더넷 어댑터가 있는 시스템이 iWARP 어댑터 또는 iWARP 드라이버를 실행하는 다른 시스템과 상호 운용하거나 iWARP를 지원하는 하드웨어를 사용하여 호스트를 실행할 수 있습니다.



중요

soft-iWARP 기능은 기술 프리뷰로만 제공됩니다. 기술 프리뷰 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원되지 않을 수 있으며, 기능적으로 완전하지 않을 수 있으며 Red Hat은 프로덕션에 이러한 기능을 사용하지 않는 것이 좋습니다. 이러한 프리뷰를 통해 향후 제품 기능에 조기 액세스할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

기술 프리뷰 기능의 지원 범위에 대한 자세한 내용은 Red Hat 고객 포털의 기술 프리뷰 기능 지원 범위를 참조하십시오.

soft-iWARP를 구성하려면 스크립트에서 이 절차를 사용하여 시스템이 부팅될 때 자동으로 실행될 수 있습니다.

사전 요구 사항

- 이더넷 어댑터가 설치됨

절차

1. **iproute**,**libibverbs -utils**, **infiniband-diags** 패키지를 설치합니다.

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

2. RDMA 링크를 표시합니다.

```
# rdma link show
```

3. **siw** 커널 모듈을 로드합니다.

```
# modprobe siw
```

4. **enp0s1** 인터페이스를 사용하는 **siw 0** 이라는 새 siw 장치를 추가합니다.

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

검증

1. 모든 RDMA 링크의 상태를 표시합니다.

```
# rdma link show

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

2. 사용 가능한 RDMA 장치를 나열합니다.

```
# ibv_devices

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

3. **ibv_devinfo** 유틸리티를 사용하여 자세한 상태를 표시할 수 있습니다.

```
# 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
```

4장. 코어 RDMA 하위 시스템 구성

이 섹션에서는 **rdma** 서비스를 구성하고 사용자가 시스템에서 고정할 수 있는 메모리 양을 늘리는 방법을 설명합니다.

4.1. IPOIB 장치 이름 변경

기본적으로 커널은 InfiniBand(IPoIB) 장치를 통한 인터넷 프로토콜(예: **ib0**,**ib1** 등)을 지정합니다. 충돌을 피하기 위해 Red Hat은 **udev** 장치 관리자에 규칙을 생성하여 **mlx4_ib0** 과 같은 지속적이고 의미 있는 이름을 만들 것을 권장합니다.

사전 요구 사항

- InfiniBand 장치가 설치됨

절차

1. 장치 **ib0** 의 하드웨어 주소를 표시합니다.

```
# 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:00:00:00:00:00:00:00:ff:ff:ff:ff
```

   주소의 마지막 8바이트는 다음 단계에서 **udev** 규칙을 생성하는 데 필요합니다.

2. 장치의 이름을 **00:02:c9:03:00:31:78:f2** 하드웨어 주소를 **mlx4_ib0** 으로 변경하는 규칙을 구성하려면 **/etc/udev/rules.d/70-persistent-ipoib.rules** 파일을 편집합니다.

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

3. **호스트를 재부팅합니다.**

```
# reboot
```

추가 리소스

- **udev(7) 도움말 페이지**
- **IPoIB 하드웨어 주소 이해**

4.2. 사용자가 시스템에서 고정할 수 있는 메모리 양 증가

RDMA(Remote Direct Memory Access) 작업을 수행하려면 실제 메모리 고정이 필요합니다. 결과적으로 커널은 스왑 공간에 메모리를 쓸 수 없습니다. 사용자가 메모리를 너무 많이 고정하면 시스템은 메모리가 부족해질 수 있으며 커널은 프로세스를 종료하여 더 많은 메모리를 확보합니다. 따라서 메모리 고정은 권한 있는 작업입니다.

루트가 아닌 사용자가 대용량 **RDMA** 애플리케이션을 실행하는 경우 이 사용자가 시스템에 고정할 수 있는 메모리 양을 늘려야 합니다. 이 섹션에서는 **rdma** 그룹에 대해 무제한 메모리 양을 구성하는 방법을 설명합니다.

절차

- **root** 사용자로 다음 콘텐츠를 사용하여 **/etc/security/limits.conf** 파일을 만듭니다.

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

검증

1. **/etc/security/limits.conf** 파일을 편집한 후 **rdma** 그룹의 멤버로 로그인합니다.

   Red Hat Enterprise Linux는 사용자가 로그인할 때 업데이트된 **ulimit** 설정을 적용합니다.

2. **ulimit -l** 명령을 사용하여 제한을 표시합니다.

```
$ ulimit -l
unlimited
```

   명령이 무제한 을 반환하는 경우 사용자는 무제한 메모리를 고정할 수 있습니다.

추가 리소스

- **limits.conf(5)** 매뉴얼 페이지

4.3. RDMA 서비스 구성

rdma 서비스는 커널의 스택을 관리합니다. **Red Hat Enterprise Linux**가 **InfiniBand, iWARP** 또는 **RoCE** 장치 및 동일한 장치의 구성 파일을 **/etc/rdma/modules/*** 에 감지한 경우 **udev** 장치 관리자는

systemd 에 **rdma** 서비스를 시작하도록 지시합니다. 기본적으로 **/etc/rdma/modules/rdma.conf** 는 이러한 서비스를 구성하고 로드합니다.

절차

1. **/etc/rdma/modules/rdma.conf** 파일을 편집하고 활성화할 변수를 **yes** 로 설정합니다.

```
# Load IPoIB
IPOIB_LOAD=yes
# Load SRP (SCSI Remote Protocol initiator support) module
SRP_LOAD=yes
# Load SRPT (SCSI Remote Protocol target support) module
SRPT_LOAD=yes
# Load iSER (iSCSI over RDMA initiator support) module
ISER_LOAD=yes
# Load iSERT (iSCSI over RDMA target support) module
ISERT_LOAD=yes
# Load RDS (Reliable Datagram Service) network protocol
RDS_LOAD=no
# Load NFSoRDMA client transport module
XPRTRDMA_LOAD=yes
# Load NFSoRDMA server transport module
SVCRDMA_LOAD=no
# Load Tech Preview device driver modules
TECH_PREVIEW_LOAD=no
```

2. **rdma** 서비스를 다시 시작하십시오.

```
# systemctl restart rdma
```

4.4. RDMA를 통한 NFS 활성화 (NFSORDMA)

RDMA(Remote Direct Memory Access) 서비스는 **Red Hat Enterprise Linux 8**의 **RDMA** 가능 하드웨어에서 자동으로 작동합니다.

절차

1. **rdma-core** 패키지를 설치합니다.

```
# yum install rdma-core
```

2. **/etc/rdma/modules/rdma.conf** 파일에서 **xprtrdma** 및 **svcrdma** 가 있는 행을 주석 처리했는지 확인합니다.

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

3. NFS 서버에서 디렉토리 /mnt/nfsordma 를 만들고 /etc/exports 로 내보냅니다.

```
# mkdir /mnt/nfsordma
# echo "/mnt/nfsordma *(fsid=0,rw,async,insecure,no_root_squash)" >> /etc/exports
```

4. NFS 클라이언트에서 서버 IP 주소(예: 172.31.0.186)를 사용하여 nfs-share를 마운트합니다.

```
# mount -o rdma,port=20049 172.31.0.186:/mnt/nfs-share /mnt/nfs
```

5. nfs-server 서비스를 다시 시작합니다.

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

추가 리소스

- RFC 5667 표준

5장. INFINIBAND 서브넷 관리자 구성

모든 InfiniBand 네트워크에는 네트워크가 작동하려면 서브넷 관리자가 실행 중이어야 합니다. 두 시스템이 연결된 스위치 없이 직접 연결된 경우에도 마찬가지입니다.

서브넷 관리자가 두 개 이상 있을 수 있습니다. 이 경우 마스터 역할을 하고 다른 서브넷 관리자는 마스터 서브넷 관리자가 실패할 경우 이를 수행하는 슬레이브 역할을 합니다.

대부분의 InfiniBand 스위치에는 서브넷 관리자가 포함되어 있습니다. 그러나 최신 서브넷 관리자가 필요하거나 추가 제어가 필요한 경우 Red Hat Enterprise Linux에서 제공하는 OpenSM 서브넷 관리자를 사용하십시오.

5.1. OPENSM 서브넷 관리자 설치

이 섹션에서는 OpenSM 서브넷 관리자를 설치하는 방법에 대해 설명합니다.

절차

1. opensm 패키지를 설치합니다.

```
# yum install opensm
```

2. 기본 설치가 사용자 환경과 일치하지 않는 경우 OpenSM을 설정합니다.

   하나의 InfiniBand 포트만 사용하면 호스트는 사용자 지정 변경 사항이 필요하지 않은 마스터 서브넷 관리자 역할을 합니다. 기본 구성은 수정 없이 작동합니다.

3. opensm 서비스를 활성화하고 시작합니다.

```
# systemctl enable --now opensm
```

추가 리소스

- opensm(8) 매뉴얼 페이지

5.2. 간단한 방법을 사용하여 OPENSM 구성

이 섹션에서는 사용자 지정 설정 없이 **OpenSM**을 구성하는 방법에 대해 설명합니다.

사전 요구 사항

- 하나 이상의 **InfiniBand** 포트가 서버에 설치됨

절차

1. **ibstat** 유틸리티를 사용하여 포트의 **GUID**를 가져옵니다.

```
# ibstat -d mlx4_0

CA 'mlx4_0'
  CA type: MT4099
  Number of ports: 2
  Firmware version: 2.42.5000
  Hardware version: 1
  Node GUID: 0xf4521403007be130
  System image GUID: 0xf4521403007be133
  Port 1:
    State: Active
    Physical state: LinkUp
    Rate: 56
    Base lid: 3
    LMC: 0
    SM lid: 1
    Capability mask: 0x02594868
    Port GUID: 0xf4521403007be131
    Link layer: InfiniBand
  Port 2:
    State: Down
    Physical state: Disabled
    Rate: 10
    Base lid: 0
    LMC: 0
    SM lid: 0
    Capability mask: 0x04010000
    Port GUID: 0xf65214fffe7be132
    Link layer: Ethernet
```

참고

일부 **InfiniBand** 어댑터는 노드, 시스템 및 포트에 동일한 **GUID**를 사용합니다.

2. /etc/sysconfig/opensm 파일을 편집하고 GUIDS 매개변수에서 GUID를 설정합니다.

```
GUIDS="GUID_1 GUID_2"
```

3. 서브넷에 여러 서브넷 관리자를 사용할 수 있는 경우 PRIORITY 매개변수를 설정할 수 있습니다. 예를 들면 다음과 같습니다.

```
PRIORITY=15
```

추가 리소스

- /etc/sysconfig/opensm

5.3. OPENSM.CONF 파일을 편집하여 OPENSM 구성

이 섹션에서는 /etc/rdma/opensm.conf 파일을 편집하여 OpenSM을 구성하는 방법을 설명합니다. InfiniBand 포트를 하나만 사용할 수 있는 경우 이 방법을 사용하여 OpenSM 구성을 사용자 지정합니다.

사전 요구 사항

- 서버에는 하나의 InfiniBand 포트만 설치됨

절차

1. /etc/rdma/opensm.conf 파일을 편집하고 사용자 환경에 맞게 설정을 사용자 지정합니다.

   opensm 패키지를 업데이트한 후 yum 유틸리티는 /etc/rdma/opensm.conf 를 재정의하고 새로운 OpenSM 설정 파일 /etc/rdma/opensm.conf.rpmnew 를 만듭니다. 따라서 이전 파일과 새 파일을 비교하여 변경 사항을 확인하고 file opensm.conf 에 수동으로 통합할 수 있습니다.

2. opensm 서비스를 다시 시작합니다.

```
# systemctl restart opensm
```

5.4. 여러 OPENSM 인스턴스 구성

이 섹션에서는 **OpenSM**의 여러 인스턴스를 설정하는 방법에 대해 설명합니다.

사전 요구 사항

- 하나 이상의 **InfiniBand** 포트가 서버에 설치됨

절차

1. **/etc/rdma/opensm.conf** 파일을 **/etc/rdma/opensm.conf.orig** 파일에 복사합니다.

```
# cp /etc/rdma/opensm.conf /etc/rdma/opensm.conf.orig
```

업데이트된 **opensm** 패키지를 설치하면 **yum** 유틸리티에서 **/etc/rdma/opensm.conf** 를 덮어씁니다. 이 단계에서 생성된 복사본을 사용하여 이전 파일과 새 파일을 비교하여 변경 사항을 식별하고 인스턴스별 **opensm.conf** 파일에 수동으로 통합합니다.

2. **/etc/rdma/opensm.conf** 파일의 사본을 만듭니다.

```
# cp /etc/rdma/opensm.conf /etc/rdma/opensm.conf.1
```

생성한 각 인스턴스에 대해 고유한 연속 숫자를 구성 파일의 복사본에 추가합니다.

opensm 패키지를 업데이트한 후 **yum** 유틸리티는 새 **OpenSM** 구성 파일을 **/etc/rdma/opensm.conf.rpmnew** 로 저장합니다. 이 파일을 사용자 지정된 **/etc/rdma/opensm.conf.conf.*** 파일과 비교하고 변경 사항을 수동으로 통합합니다.

3. 이전 단계에서 만든 복사본을 편집하고 환경에 맞게 인스턴스의 설정을 사용자 지정합니다. 예를 들어 **guid,subnet_prefix, logdir** 매개변수를 설정합니다.

4. 선택적으로 이 서브넷에 대한 고유 이름을 사용하여 **partitions.conf** 파일을 생성하고, **opensm.conf** 파일의 해당 사본에 **partition_config_file** 매개변수에서 해당 파일을 참조합니다.

5. 생성할 각 인스턴스에 대해 이전 단계를 반복합니다.

6. **opensm** 서비스를 시작합니다.

```
# systemctl start opensm
```

opensm 서비스는 **/etc/rdma/** 디렉토리의 각 **opensm.conf.*** 파일에 대해 고유한 인스턴스를 자동으로 시작합니다. 여러 **opensm.conf.*** 파일이 있는 경우 서비스는 **/etc/sysconfig/opensm** 파일의 설정과 기본 **/etc/rdma/opensm.conf** 파일의 설정을 무시합니다.

5.5. 파티션 설정 생성

파티션을 사용하면 관리자가 이더넷 **VLAN**과 유사한 **InfiniBand**에서 서브넷을 만들 수 있습니다.

중요

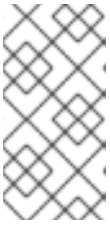
40Gbps와 같은 특정 속도로 파티션을 정의하는 경우 이 파티션 내의 모든 호스트가 최소 속도를 지원해야 합니다. 호스트가 속도 요구 사항을 충족하지 않으면 파티션에 참여할 수 없습니다. 따라서 파티션의 속도를 파티션에 가입할 수 있는 권한이 있는 모든 호스트에서 지원하는 최저 속도로 설정합니다.

사전 요구 사항

- 하나 이상의 **InfiniBand** 포트가 서버에 설치됨

절차

1. **/etc/rdma/partitions.conf** 파일을 편집하여 다음과 같이 파티션을 구성합니다.



참고

모든 패브릭은 **0x7fff** 파티션을 포함해야 하며 모든 스위치와 모든 호스트가 해당 패브릭에 속해야 합니다.

다음 콘텐츠를 파일에 추가하여 **0x7fff** 기본 파티션을 **10 Gbps**의 감소 속도로 생성하고 파티션 **0x0002** 를 **40 Gbps**로 추가합니다.

```
# For reference:
# IPv4 IANA reserved multicast addresses:
```

```
#   http://www.iana.org/assignments/multicast-addresses/multicast-addresses.txt
# IPv6 IANA reserved multicast addresses:
#   http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-
addresses.xml
#
# mtu =
#   1 = 256
#   2 = 512
#   3 = 1024
#   4 = 2048
#   5 = 4096
#
# rate =
#   2  = 2.5 GBit/s
#   3  = 10   GBit/s
#   4  = 30   GBit/s
#   5  = 5   GBit/s
#   6  = 20   GBit/s
#   7  = 40   GBit/s
#   8  = 60   GBit/s
#   9  = 80   GBit/s
#   10 = 120   GBit/s

Default=0x7fff, rate=3, mtu=4, scope=2, defmember=full:
   ALL, ALL_SWITCHES=full;
Default=0x7fff, ipoib, rate=3, mtu=4, scope=2:
   mgid=ff12:401b::ffff:ffff   # IPv4 Broadcast address
   mgid=ff12:401b::1          # IPv4 All Hosts group
   mgid=ff12:401b::2          # IPv4 All Routers group
   mgid=ff12:401b::16          # IPv4 IGMP group
   mgid=ff12:401b::fb          # IPv4 mDNS group
   mgid=ff12:401b::fc          # IPv4 Multicast Link Local Name Resolution group
   mgid=ff12:401b::101          # IPv4 NTP group
   mgid=ff12:401b::202          # IPv4 Sun RPC
   mgid=ff12:601b::1          # IPv6 All Hosts group
   mgid=ff12:601b::2          # IPv6 All Routers group
   mgid=ff12:601b::16          # IPv6 MLDv2-capable Routers group
   mgid=ff12:601b::fb          # IPv6 mDNS group
   mgid=ff12:601b::101          # IPv6 NTP group
   mgid=ff12:601b::202          # IPv6 Sun RPC group
   mgid=ff12:601b::1:3          # IPv6 Multicast Link Local Name Resolution group
   ALL=full, ALL_SWITCHES=full;

ib0_2=0x0002, rate=7, mtu=4, scope=2, defmember=full:
      ALL, ALL_SWITCHES=full;
ib0_2=0x0002, ipoib, rate=7, mtu=4, scope=2:
   mgid=ff12:401b::ffff:ffff   # IPv4 Broadcast address
   mgid=ff12:401b::1          # IPv4 All Hosts group
   mgid=ff12:401b::2          # IPv4 All Routers group
   mgid=ff12:401b::16          # IPv4 IGMP group
   mgid=ff12:401b::fb          # IPv4 mDNS group
   mgid=ff12:401b::fc          # IPv4 Multicast Link Local Name Resolution group
   mgid=ff12:401b::101          # IPv4 NTP group
   mgid=ff12:401b::202          # IPv4 Sun RPC
   mgid=ff12:601b::1          # IPv6 All Hosts group
   mgid=ff12:601b::2          # IPv6 All Routers group
```

```
mgid=ff12:601b::16          # IPv6 MLDv2-capable Routers group
mgid=ff12:601b::fb          # IPv6 mDNS group
mgid=ff12:601b::101         # IPv6 NTP group
mgid=ff12:601b::202         # IPv6 Sun RPC group
mgid=ff12:601b::1:3         # IPv6 Multicast Link Local Name Resolution group
ALL=full, ALL_SWITCHES=full;
```

6장. IPOIB 구성

기본적으로 InfiniBand는 통신에 인터넷 프로토콜(IP)을 사용하지 않습니다. 그러나 IP over InfiniBand(IPoIB)는 InfiniBand RDMA(Remote Direct Memory Access) 네트워크 위에 IP 네트워크 에뮬레이션 계층을 제공합니다. 이렇게 하면 수정되지 않은 기존 애플리케이션에서 InfiniBand 네트워크를 통해 데이터를 전송할 수 있지만 성능은 애플리케이션이 기본적으로 RDMA를 사용하는 것보다 낮습니다.

참고

Mellanox 장치는, ConnectX-4 이상에서 시작하여 RHEL 8에서는 기본적으로 향상된 IPoIB 모드를 사용합니다(데이터그램만 해당). 연결 모드는 이러한 장치에서 지원되지 않습니다.

6.1. IPOIB 통신 모드

IPoIB 장치는 Datagram 또는 Connected 모드에서 구성할 수 있습니다. 차이점은 통신의 다른 끝에서 IPoIB 계층이 머신으로 열려고 하는 대기열 쌍의 유형입니다.

- Datagram 모드에서는 시스템이 신뢰할 수 없고 연결이 끊긴 큐 쌍을 엽니다.

  이 모드는 InfiniBand 링크 계층의 MTU(최대 전송 단위)보다 큰 패키지를 지원하지 않습니다. 데이터를 전송하는 동안, IPoIB 계층은 IP 패킷 상단에 4 바이트 IPoIB 헤더를 추가합니다. 결과적으로 IPoIB MTU는 InfiniBand 링크 계층 MTU보다 4바이트 미만입니다. 2048 는 일반적인 InfiniBand 링크 계층 MTU이므로 Datagram 모드의 공통 IPoIB 장치 MTU는 2044 입니다.

- 연결 모드에서는 시스템이 신뢰할 수 있고 연결된 큐 쌍을 엽니다.

  이 모드에서는 InfiniBand 링크 MTU보다 큰 메시지를 사용할 수 있습니다. 호스트 어댑터는 패킷 분할 및 재사양을 처리합니다. 결과적으로 연결 모드에서 Infiniband 어댑터에서 보낸 메시지에 크기 제한이 없습니다. 그러나 데이터 필드 및 TCP/IP 헤더 필드로 인해 IP 패킷이 제한됩니다. 이러한 이유로 연결된 모드의 IPoIB MTU는 65520 바이트입니다.

  연결된 모드는 성능이 높지만 커널 메모리를 더 많이 사용합니다.

시스템이 Connected 모드를 사용하도록 구성되었지만 InfiniBand 스위치 및 패브릭은 연결된 모드에서 멀티 캐스트 트래픽을 전달할 수 없기 때문에 여전히 데이터그램 모드를 사용하여 멀티캐스트 트래픽을 보냅니다. 또한 호스트가 Connected 모드를 사용하도록 구성되지 않은 경우 시스템이 데이터그램 모드로 전환됩니다.

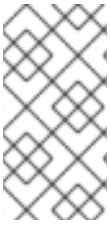
인터페이스의 **MTU**로 멀티 캐스트 데이터를 전송하는 애플리케이션을 실행하는 동안 **Datagram** 모드에서 인터페이스를 구성하거나 데이터그램 패킷에 적합한 패킷의 전송 크기를 제한하도록 애플리케이션을 구성합니다.

6.2. IPOIB 하드웨어 주소 이해

IPoIB 장치에는 다음 부분으로 구성된 **20** 바이트 하드웨어 주소가 있습니다.

- 처음 **4**바이트는 플래그 및 큐 쌍 번호입니다.
- 다음 **8**바이트는 서브넷 접두사입니다.

  기본 서브넷 접두사는 **0xfe:80:00:00:00:00**입니다. 장치가 서브넷 관리자에 연결한 후 장치는 이 접두사를 구성된 서브넷 관리자와 일치하도록 변경합니다.
- 마지막 **8**바이트는 **IPoIB** 장치에 연결되는 **InfiniBand** 포트의 **GUID(Globally Unique Identifier)**입니다.



참고

처음 **12**바이트가 변경될 수 있으므로 **udev** 장치 관리자 규칙에서는 사용하지 마십시오.

6.3. NMCLI 명령을 사용하여 IPOIB 연결 구성

nmcli 명령줄 유틸리티는 **NetworkManager**를 제어하고 **CLI**를 사용하여 네트워크 상태를 보고합니다.

사전 요구 사항

- **InfiniBand** 장치가 서버에 설치됨
- 해당 커널 모듈이 로드됩니다.

절차

1. 연결된 전송 모드에서 **mlx4_ib0** 인터페이스와 **65520** 바이트의 최대 **MTU**를 사용하려면 **InfiniBand** 연결을 만듭니다.

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

2. **mlx4_ib0** 연결의 **P_Key** 인터페이스로 **0x8002** 를 설정할 수도 있습니다.

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

3. **IPv4** 설정을 구성하려면 **mlx4_ib0** 연결의 정적 **IPv4** 주소, 네트워크 마스크, 기본 게이트웨이 및 **DNS** 서버를 설정합니다.

```
# 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** 설정을 구성하려면 **mlx4_ib0** 연결의 정적 **IPv6** 주소, 네트워크 마스크, 기본 게이트웨이 및 **DNS** 서버를 설정합니다.

```
# 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** 연결을 활성화하려면 다음을 수행합니다.

```
# nmcli connection up mlx4_ib0
```

6.4. NM-CONNECTION-EDITOR를 사용하여 IPOIB 연결 구성

nmcli-connection-editor 애플리케이션은 **GUI**를 사용하여 **NetworkManager**에 저장된 네트워크 연결을 구성하고 관리합니다.

사전 요구 사항

- **InfiniBand** 장치가 서버에 설치됨

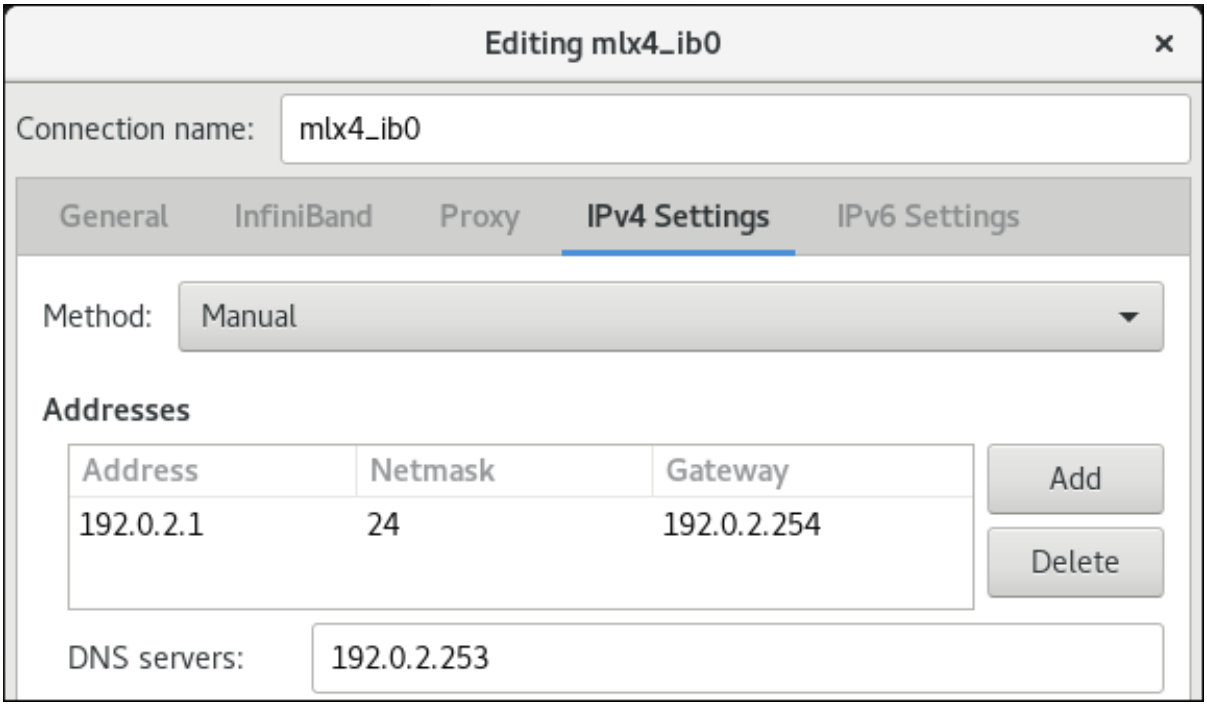
- 해당 커널 모듈이 로드됨
- **nm-connection-editor** 패키지가 설치됨

절차

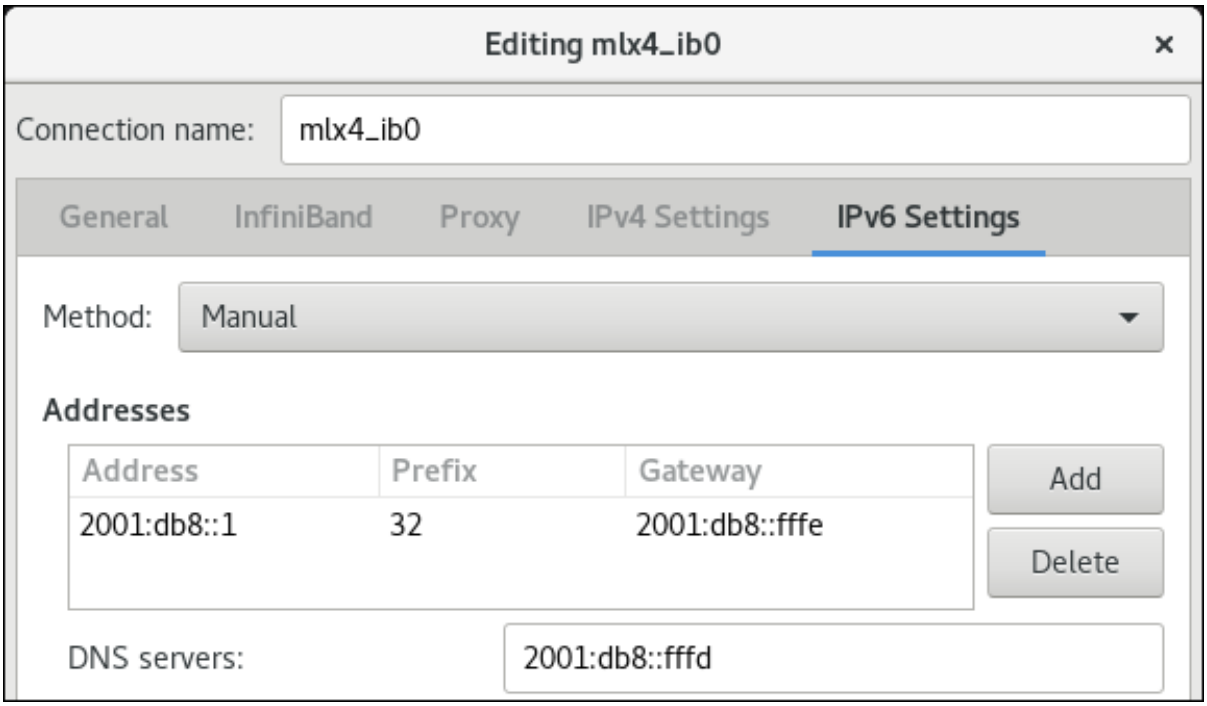
1. 명령을 입력합니다.

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

2. **+** 버튼을 클릭하여 새 연결을 추가합니다.
3. **InfiniBand** 연결 유형을 선택하고 생성을 클릭합니다.
4. **InfiniBand** 탭에서 다음을 수행합니다.
   a. 원하는 경우 연결 이름을 변경합니다.
   b. 전송 모드를 선택합니다.
   c. 장치를 선택합니다.
   d. 필요한 경우 **MTU**를 설정합니다.
5. **IPv4 Settings(IPv4** 설정 **)** 탭에서 **IPv4** 설정을 구성합니다. 예를 들어 정적 **IPv4** 주소, 네트워크 마스크, 기본 게이트웨이 및 **DNS** 서버를 설정합니다.



6. **IPv6 Settings(IPv6 설정) 탭에서 IPv6 설정을 구성합니다. 예를 들어 정적 IPv6 주소, 네트워크 마스크, 기본 게이트웨이 및 DNS 서버를 설정합니다.**



7. **Save(저장)를 클릭하여 팀 연결을 저장합니다.**

8. **nm-connection-editor 를 닫습니다.**

9.

P_Key 인터페이스를 설정할 수 있습니다. nm-connection-editor 에서 이 설정을 사용할 수 없으므로 명령줄에서 이 매개변수를 설정해야 합니다.

예를 들어, 0x8002 를 mlx4_ib0 연결의 P_Key 인터페이스로 설정하려면 다음을 수행합니다.

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

7장. INFINIBAND 네트워크 테스트

이 섹션에서는 InfiniBand 네트워크를 테스트하는 방법을 설명합니다.

7.1. 초기 INFINIBAND RDMA 작업 테스트

이 섹션에서는 InfiniBand RDMA(원격 직접 메모리 액세스) 작업을 테스트하는 방법을 설명합니다.

참고

이 섹션은 InfiniBand 장치에만 적용됩니다. Internet Wide-area Remote Protocol(iWARP) 또는 RDMA over Converged Ethernet (RoCE) 또는 InfiniBand over Ethernet(IBoE) 장치와 같은 IP 기반 장치를 사용하는 경우 다음을 참조하십시오.

- ping 유틸리티를 사용하여 IPoIB 테스트
- IPoIB를 구성한 후 qperf를 사용하여 RDMA 네트워크 테스트

사전 요구 사항

- rdma 서비스가 구성되어 있습니다.
- libibverbs-utils 및 infiniband-diags 패키지가 설치됨

절차

1. 사용 가능한 InfiniBand 장치를 나열합니다.

```
# ibv_devices

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

2. mlx4_1 장치의 정보를 표시하려면 다음을 수행합니다.

```
# 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 장치의 상태를 표시하려면 다음을 수행합니다.**

```
# 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 유틸리티는 InfiniBand 주소를 ping하고 클라이언트/서버로 실행됩니다.

a. 호스트에서 서버 모드를 시작하려면 -C InfiniBand CA(인증 기관) 이름을 사용하여 포트 번호 -P 에서 -S 매개 변수를 사용합니다.

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

b. 다른 호스트에서 클라이언트 모드를 시작하려면 -C InfiniBand CA(인증 기관) 이름을 -L ID(local Identifier)로 사용하여 포트 번호 -P 에서 일부 패킷 -c 를 보냅니다.

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

추가 리소스

- ibping(8) 매뉴얼 페이지

7.2. PING 유틸리티를 사용하여 IPOIB 테스트

InfiniBand (IPoIB)를 통해 IP를 구성한 후 ping 유틸리티를 사용하여 ICMP 패킷을 보내 IPoIB 연결을 테스트합니다.

사전 요구 사항

- 두 개의 RDMA 호스트는 RDMA 포트가 있는 동일한 InfiniBand 패브릭에 연결되어 있습니다.
- 두 호스트의 IPoIB 인터페이스는 동일한 서브넷 내의 IP 주소로 구성됩니다.

절차

-

ping 유틸리티를 사용하여 5개의 ICMP 패킷을 원격 호스트의 InfiniBand 어댑터에 전송합니다.

```
# ping -c5 192.0.2.1
```

7.3. IPOIB 구성 후 QPERF를 사용하여 RDMA 네트워크 테스트

qperf 유틸리티는 대역폭, 대기 시간, CPU 사용률 측면에서 두 노드 간에 RDMA 및 IP 성능을 측정합니다.

사전 요구 사항

- qperf 패키지는 두 호스트 모두에 설치되어 있습니다.
- IPoIB가 두 호스트 모두에 구성되어 있습니다.

절차

1. 서버 역할을 하는 옵션 없이 호스트 중 하나에서 qperf 을 시작합니다.

```
# qperf
```

2. 클라이언트에서 다음 명령을 사용합니다. 이 명령은 클라이언트의 mlx4_0 호스트 채널 어댑터의 포트 1 을 사용하여 서버의 InfiniBand 어댑터에 할당된 IP 주소 192.0.2.1 에 연결합니다.

   a. 구성을 표시하려면 다음을 수행합니다.

```
# 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.

신뢰할 수 있는 연결 (RC)을 표시하는 방법 2 방향 대역폭:

```
# 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 스트리밍 1방향 대역폭을 표시하려면 다음을 수행합니다.

```
# 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
```

추가 리소스

- qperf(1) 매뉴얼 페이지