Red Hat Enterprise Linux 8

配置 InfiniBand 和 RDMA 网络

配置和管理高速网络协议和 RDMA 硬件

Last Updated: 2024-06-27

Red Hat Enterprise Linux 8 配置 InfiniBand 和 RDMA 网络

配置和管理高速网络协议和 RDMA 硬件

法律通告

摘要

您可以使用各种协议在企业级配置和管理Remote Directory Memory Access (RDMA)网络和 InfiniBand 硬件。这包括 RDMA over Converged Ethernet (RoCE)、RoCE (Soft-RoCE)的软件实现、IP 网络协议，如 iWARP、iWARP (Soft-iWARP)的软件实现，以及通过 RDMA (NFSoRDMA)协议的网络文件系统，作为对 RDMA 支持的硬件的原生支持。对于低延迟和高吞吐量连接，您可以配置 IP over InfiniBand (IPoIB) 和 Open subnet Manager (OpenSM)。

目录

对红帽文档提供反馈

我们感谢您对我们文档的反馈。让我们了解如何改进它。

通过 Jira 提交反馈（需要帐户）

1. 登录到 Jira 网站。
2. 单击顶部导航栏中的 **Create**。
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您的建议以改进。包括文档相关部分的链接。
5. 点对话框底部的 **Create**。

第 1 章 了解 INFINIBAND 和 RDMA

InfiniBand 代表两个不同的因素：

- InfiniBand 网络的物理链路协议
- InfiniBand Verbs API，这是远程直接访问(RDMA)技术的实现

RDMA 提供两个计算机的主要内存访问，而无需涉及操作系统、缓存或存储。使用 RDMA，带有高吞吐量、低延迟和 CPU 使用率的数据传输。

在典型的 IP 数据传输中，当一个计算机上的应用程序向另一台机器上的应用程序发送数据时，接收终止时会出现以下操作：

1. 内核必须接收数据。
2. 内核必须确定该数据是否属于该应用程序。
3. 内核唤醒应用程序。
4. 内核会等待应用程序执行系统调用到内核。
5. 应用程序将内核的内部内存空间中的数据复制到应用程序提供的缓冲中。

此过程意味着，如果主机适配器使用直接内存访问(DMA)或者至少两次，则大多数网络流量会被复制到系统的主内存中。另外，计算机执行一些上下文切换以在内核和应用程序间切换。这些上下文切换可能会导致 CPU 负载高，但会降低其他任务的速度。

与传统的 IP 通信不同，RDMA 通信会绕过通信过程中的内核干预。这可减少 CPU 开销。RDMA 协议可让主机适配器在数据包进入网络后决定应用程序应该接收的网络以及将其保存到应用程序的内存空间中。主机适配器不将处理发送到内核并将其复制到用户应用程序的内存中，主机适配器直接在应用程序缓冲中放置数据包内容。此过程需要单独的 API、InfiniBand Verbs API 和应用程序需要实施 InfiniBand Verbs API 来使用 RDMA。

Red Hat Enterprise Linux 支持 InfiniBand 硬件和 InfiniBand Verbs API。另外，它支持以下技术在非 InfiniBand 硬件中使用 InfiniBand Verbs API:

- 互联网区 RDMA 协议(iWARP)：通过 IP 网络实现 RDMA 的网络协议
- RDMA over Converged Ethernet(RoCE)，也称为 InfiniBand over Ethernet(IBoE)：实现通过以太网网络实施 RDMA 的网络协议

其它资源

- 配置 RoCE

第 2 章 配置 ROCE

远程直接内存访问(RDMA)为直接内存访问(DMA)提供远程执行。RDMA over Converged Ethernet (RoCE)是一种网络协议，它通过以太网网络使用 RDMA。对于配置，RoCE 需要特定的硬件，并且一些硬件供应商是 Mellanox、Broadcom 和 QLogic。

2.1. ROCE 协议版本概述

RoCE 是一种网络协议，可实现通过以太网的远程直接访问(RDMA)。

以下是不同的 RoCE 版本：

RoCE v1

RoCE 版本 1 协议是一个以太网链路层协议，带有 ethertype **0x8915**，它允许同一以太网广播域中的任何两个主机间的通信。

RoCE v2

RoCE 版本 2 协议在 IPv4 或 IPv6 协议的 UDP 上存在。对于 RoCE v2，UDP 目标端口号为 **4791**。

RDMA_CM 设置客户端和服务器之间用来传输数据的可靠连接。RDMA_CM 为建立连接提供了一个与 RDMA 传输相关的接口。这个通信使用特定的 RDMA 设备和基于消息的数据传输。

重要

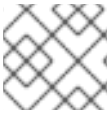
不支持在客户端中使用 RoCE v2 的不同版本，并在服务器中使用 RoCE v1。在这种情况下，将服务器和客户端都配置为通过 RoCE v1 进行通信。

其它资源

- 临时更改默认 RoCE 版本

2.2. 临时更改默认 ROCE 版本

不支持在客户端和服务器上的 RoCE v1 使用 RoCE v2 协议。如果您的服务器中的硬件只支持 RoCE v1，请为 RoCE v1 配置客户端，来与服务器进行通信。例如，您可以为只支持 RoCE v1 的 Mellanox ConnectX-5 InfiniBand 设备配置使用 **mlx5_0** 驱动程序的客户端。



注意

此处描述的更改将保持有效，直到重新引导主机为止。

先决条件

- 客户端使用带有 RoCE v2 协议的 InfiniBand 设备。
- 服务器使用只支持 RoCE v1 的 InfiniBand 设备。

流程

1. 创建 **/sys/kernel/config/rdma_cm/*mlx5_0*/** 目录：

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

2. 显示默认 RoCE 模式：

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

RoCE v2
```

3. 将默认 RoCE 模式改为版本 1:

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

2.3. 配置 SOFT-ROCE

Soft-RoCE 是 RDMA over Ethernet 的一个软件实现，它也称为 RXE。在没有 RoCE 主机频道适配器 (HCA)的主机上使用 Soft-RoCE。

重要

Soft-RoCE 功能仅作为技术预览提供。红帽产品服务级别协议（SLA）不支持技术预览功能，且其功能可能并不完善，因此红帽不建议在生产环境中使用它们。这些预览可让用户早期访问将来的产品功能，让用户在开发过程中测试并提供反馈意见。

如需有关 技术预览功能支持范围 的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

先决条件

- 已安装以太网适配器

流程

1. 安装 **iproute**、**libibverbs**、**libibverbs-utils** 和 **infiniband-diags** 软件包：

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

2. 显示 RDMA 链接：

```
# rdma link show
```

3. 加载 **rdma_rxe** 内核模块，再添加名为 **rxe0** 的新 **rxe** 设备，其使用 **enp0s1** 接口：

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

验证

1. 查看所有 RDMA 链接的状态：

```
# rdma link show

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

2. 列出可用的 RDMA 设备：

```
# ibv_devices

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

3. 您可以使用 **ibstat** 程序来显示详细的状态：

```
# 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 3 章 CONFIGURING SOFT-IWARP

远程直接内存访问(RDMA)通过以太网使用多个库和协议，如 iWARP、Soft-iWARP 用于提高性能和辅助编程接口。

3.1. IWARP 和 SOFT-IWARP 概述

远程直接内存访问(RDMA)使用互联网 Wide-area RDMA 协议(iWARP)并通过 TCP 进行融合和低延迟数据传输。使用标准以太网交换机和 TCP/IP 堆栈，iWARP 在 IP 子网之间路由流量。这提供了高效使用现有基础架构的灵活性。在 Red Hat Enterprise Linux 中，多个提供商在其硬件网络接口卡中实施 iWARP。例如：**cxgb4**,**irdma**,**qedr** 等等。

软硬件(siw)是基于软件的 iWARP 内核驱动程序和 Linux 程序库。它是一个基于软件的 RDMA 设备，在附加到网络接口卡时为 RDMA 硬件提供编程接口。它提供测试和验证 RDMA 环境的简便方法。

3.2. CONFIGURING SOFT-IWARP

软 IWARP(siw)通过 Linux TCP/IP 网络堆栈实施互联网 Wide-area RDMA 协议(iWARP)远程直接内存访问(RDMA)传输。它可让具有标准以太网适配器的系统与 iWARP 适配器或另一个系统互操作，运行 Soft-iWARP 驱动程序，或使用支持 iWARP 的硬件的主机进行互操作。

重要

Soft-iWARP 功能仅作为技术预览提供。红帽产品服务级别协议（SLA）不支持技术预览功能，且其功能可能并不完善，因此红帽不建议在生产环境中使用它们。这些预览可让用户早期访问将来的产品功能，让用户在开发过程中测试并提供反馈意见。

如需有关 技术预览功能支持范围 的信息，请参阅红帽客户门户网站中的技术预览功能支持范围。

要配置 Soft-iWARP，您可以在脚本中使用这个步骤在系统引导时自动运行。

先决条件

- 已安装以太网适配器

流程

1. 安装 **iproute**、**libibverbs**、**libibverbs-utils** 和 **infiniband-diags** 软件包：

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

2. 显示 RDMA 链接：

```
# rdma link show
```

3. 加载 **siw** 内核模块：

```
# modprobe siw
```

4. 添加一个名为 **siw0** 的新 **siw** 设备，它使用 **enp0s1** 接口：

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

验证

1. 查看所有 RDMA 链接的状态：

```
# rdma link show

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

2. 列出可用的 RDMA 设备：

```
# ibv_devices

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

3. 您可以使用 **ibv_devinfo** 工具显示详细的状态：

```
# 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 4 章 配置核心 RDMA 子系统

rdma 服务配置管理网络协议和通信标准，如 InfiniBand、iWARP 和 RoCE。

4.1. 重命名 IPOIB 设备

默认情况下，内核通过 InfiniBand(IPoIB)设备命名互联网协议，如 **ib** 0、**ib1** 等等。为了避免冲突，红帽建议在 **udev** 设备管理器中创建一个规则来创建持久且有意义的名称，如 **mlx4_ib0**。

先决条件

- 您已安装了 InfiniBand 设备。

流程

1. 显示设备 **ib0** 的硬件地址：

```
# 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:80:00:00:00:00:00:00:00:02: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
```

   地址的最后八字节需要在下一步中创建 **udev** 规则。

2. 要配置使用 **00:02:c9:03:00:31:78:f2** 硬件地址重命名为 **mlx4_ib0** 的规则，请编辑 **/etc/udev/rules.d/70-persistent-ipoib.rules** 文件并添加 **ACTION** 规则：

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

3. 重启主机：

```
# reboot
```

其它资源

- **udev(7)** 手册页
- 了解 IPoIB 硬件地址

4.2. 增加用户被允许在系统中的内存量

远程直接内存访问(RDMA)操作需要固定物理内存。因此，内核不允许将内存写入 swap 空间。如果用户固定太多内存，系统会耗尽内存，内核会终止进程来释放更多内存。因此，内存固定是一个特权操作。

如果非 root 用户需要运行大型 RDMA 应用程序，则需要增加内存量，以便在主内存中一直保持页面固定。

流程

- 以 **root** 用户身份，创建具有以下内容的文件 **/etc/security/limits.conf** ：

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

验证

1. 编辑 **/etc/security/limits.conf** 文件后，作为 **rdma** 组的成员登录。
   请注意，当用户登录时，Red Hat Enterprise Linux 会应用更新的 **ulimit** 设置。

2. 使用 **ulimit -l** 命令显示限制：

```
$ ulimit -l
unlimited
```

   如果命令返回 **unlimited**，用户可以获得无限数量的内存。

其它资源

- **limits.conf(5)** 手册页

4.3. 配置 RDMA 服务

使用远程直接内存访问(RDMA)协议，您可以使用主内存通过网络在启用了 RDMA 的系统间传输数据。RDMA 协议提供低延迟和高吞吐量。要管理支持的网络协议和通信标准，您需要配置 **rdma** 服务。此配置包括 RoCE 和 iWARP 等高速度网络协议，以及 Soft-RoCE 和 Soft-iWARP 等通信标准。当 Red Hat Enterprise Linux 检测 InfiniBand、iWARP 或 RoCE 设备及其位于 **/etc/rdma/modules/*** 目录的配置文件时，**udev** 设备管理器会指示 **systemd** 启动 **rdma** 服务。**/etc/rdma/modules/rdma.conf** 文件中模块的配置在重启后会保持不变。您需要重启 **rdma-load-modules@rdma.service** 配置服务以应用更改。

流程

1. 编辑 **/etc/rdma/modules/rdma.conf** 文件，并取消您要启用的模块的注释：

```
# 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
```

2. 重启该服务以使更改生效：

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

验证

- 重启后，检查服务状态：

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

4.4. 在 NFS 服务器中启用 RDMA 的 NFS

远程直接内存访问(RDMA)是一种协议，它允许客户端系统将数据直接从存储服务器的内存传输到其自身的内存。这提高了存储吞吐量，降低服务器和客户端之间的数据传输延迟，并减少两端的 CPU 负载。如果 NFS 服务器和客户端都通过 RDMA 连接，客户端可以使用 NFSoRDMA 来挂载导出的目录。

先决条件

- NFS 服务正在运行并配置了
- 在服务器中安装 InfiniBand 或 RDMA over Converged Ethernet (RoCE)设备。
- IP over InfiniBand (IPoIB)在服务器上被配置，InfiniBand 设备分配了一个 IP 地址。

流程

1. 安装 **rdma-core** 软件包：

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

2. 如果已经安装了软件包，请验证 **/etc/rdma/modules/rdma.conf** 文件中的 **xprtrdma** 和 **svcrdma** 模块是否已取消注释：

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

3. 可选。默认情况下，RDMA 上的 NFS 使用端口 20049。如果要使用其他端口，请在 **/etc/nfs.conf** 文件的 **[nfsd]** 部分中设置 **rdma-port** 设置：

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

4. 在 **firewalld** 中打开 NFSoRDMA 端口：

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

如果您设置了与 20049 不同的端口，请调整端口号。

5. 重启 **nfs-server** 服务：

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

验证

1. 在带有 InfiniBand 硬件的客户端中执行以下步骤：
   a. 安装以下软件包：

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

   b. 通过 RDMA 挂载导出的 NFS 共享：

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

   如果您设置了默认端口号(20049)，请将 port **= <*port_number*>** 传给命令：

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

   c. 验证共享是否已使用 **rdma** 选项挂载：

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

其它资源

- 配置 InfiniBand 和 RDMA 网络

第 5 章 配置 INFINIBAND 子网管理器

所有 InfiniBand 网络都必须运行子网管理器才能正常工作。即使两台机器没有使用交换机直接进行连接，也是如此。

有可能有一个以上的子网管理器。在这种情况下，一个 master 充当一个主子网管理器，另一个子网管理器充当从属子网管理器，当主子网管理器出现故障时将接管。

大多数 InfiniBand 交换机都包含一个嵌入式子网管理器。然而，如果您需要一个更新的子网管理器，或者您需要更多控制，请使用 Red Hat Enterprise Linux 提供的 **OpenSM** 子网管理器。

5.1. 安装 OPENSM 子网管理器

OpenSM 是一个子网管理器和管理员，它遵循 InfiniBand 规格来初始化 InfiniBand 硬件，其中至少有一个 **OpenSM** 服务实例始终在运行。

流程

1. 安装 **opensm** 软件包：

```
# yum install opensm
```

2. 如果默认安装与您的环境不匹配，请配置 OpenSM。
   当只有一个 InfiniBand 端口时，主机充当不需要任何自定义更改的 master 子网管理器。默认配置可在没有任何修改的情况下正常工作。

3. 启用并启动 **opensm** 服务：

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

其它资源

- **opensm(8)** 手册页

5.2. 使用简单方法配置 OPENSM

OpenSM 是一个基于 InfiniBand 规范的子网管理器和管理员，它配置 InfiniBand 光纤、一个网络拓扑来连接 InfiniBand 节点。

先决条件

- 服务器上安装了一个或多个 InfiniBand 端口

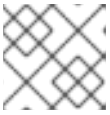
流程

1. 使用 **ibstat** 程序获取端口的 GUID：

```
# 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
```



注意

有些 InfiniBand 适配器在节点、系统和端口中使用相同的 GUID。

2. 编辑 **/etc/sysconfig/opensm** 文件并在 **GUIDS** 参数中设置 GUID：

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

3. 如果子网中有多个子网管理器，您可以设置 **PRIORITY** 参数。例如：

```
PRIORITY=15
```

其它资源

- **/etc/sysconfig/opensm**

5.3. 通过编辑 OPENSM.CONF 文件配置 OPENSM

OpenSM 性能取决于设备上可用的 InfiniBand 端口的数量。您可以通过编辑 **/etc/rdma/opensm.conf** 文件来自定义它。

先决条件

- 服务器上只安装一个 InfiniBand 端口。

流程

1. 编辑 **/etc/rdma/opensm.conf** 文件并自定义设置以匹配您的环境。

更新 **opensm** 软件包后，**yum** 程序会覆盖 **/etc/rdma/opensm.conf**，并创建一个副本，它是新的 OpenSM 配置文件 **/etc/rdma/opensm.conf.rpmnew**。因此，您可以比较之前的 和新文件，以识别更改并将其手动合并到 file **opensm.conf** 中。

2. 重启 **opensm** 服务：

```
# systemctl restart opensm
```

5.4. 配置多个 OPENSM 实例

OpenSM 是一个 InfiniBand 约束的子网管理器和管理员。要提供高性能，**OpenSM** 使用与 InfiniBand 网络节点互连的交换光纤网络拓扑。

先决条件

- 在服务器中安装一个或多个 InfiniBand 端口。

流程

1. 将 **/etc/rdma/opensm.conf** 文件复制到 **/etc/rdma/opensm.conf.orig** 文件中：

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

当您安装更新的 **opensm** 软件包时，**yum** 实用程序会覆盖 **/etc/rdma/opensm.conf**。与上一步中创建的副本进行比较，比较前面的 和新文件，以识别更改并在实例特定的 **opensm.conf** 文件中手动纳入它们。

2. 创建 **/etc/rdma/opensm.conf** 文件的副本：

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

对于每个实例，创建并将唯一的、连续的数追加到配置文件的副本中。

更新 **opensm** 软件包后，**yum** 实用程序会将新的 OpenSM 配置文件存储为 **/etc/rdma/opensm.conf.rpmnew**。将此文件与您自定义的 **/etc/rdma/opensm.conf.*** 文件进行比较，并手动纳入这些更改。

3. 编辑您在上一步中创建的副本，并自定义实例的设置以匹配您的环境。例如，设置 **guid**、**subnet_prefix** 和 **logdir** 参数。

4. 另外，还可在该子网 **partitions.conf** 中生成具有唯一名称的文件，并在该文件对应的副本中的 **partition_config_file** 参数中引用 **opensm.conf** 文件。

5. 对您要创建的每个实例重复前面的步骤。

6. 启动 **opensm** 服务：

```
# systemctl start opensm
```

opensm 服务会自动为 **/etc/rdma/** 目录中的每个 **opensm.conf.*** 文件启动一个唯一的实例。如果存在多个 **opensm.conf.*** 文件，该服务会忽略 **/etc/sysconfig/opensm** 文件中的设置，以及在基础 **/etc/rdma/opensm.conf** 文件中。

5.5. 创建分区配置

分区使管理员能够在 InfiniBand 上创建与以太网 VLAN 类似的子网。

重要

如果您使用特定速度（如 40 Gbps）定义分区，这个分区中的所有主机必须至少支持这个速度。如果主机没有满足速度要求，就无法加入该分区。因此，将分区的速度设置为有权加入分区的任何主机支持的最低速度。

先决条件

- 服务器上安装了一个或多个 InfiniBand 端口

流程

1. 编辑 **/etc/rdma/partitions.conf** 文件，以配置分区，如下所示：

注意

所有光纤必须包含 **0x7fff** 分区，所有交换机和所有主机都必须属于那个光纤。

将以下内容添加到 文件中，以减小速度 **10** Gbps 和分区 **0x0002** 创建 **0x7fff** 默认分区，其速度为 **40** Gbps：

```
# 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;
```

第 6 章 配置 IPOIB

默认情况下，InfiniBand 不使用 IP 进行通信。但是，IP over InfiniBand(IPoIB)在 InfiniBand 远程直接访问(RDMA)网络之上提供一个 IP 网络模拟层。这允许现有未经修改的应用程序通过 InfiniBand 网络传输数据，但性能低于应用程序原生使用 RDMA 时的数据。

注意

在 RHEL 8 及更高版本上，Mellanox 设备从 ConnectX-4 及以上启动，默认使用增强 IPoIB 模式（仅数据Gram）。这些设备不支持连接的模式。

6.1. IPOIB 通讯模式

IPoIB 设备可在 **Datagram** 或 **Connected** 模式中配置。区别在于 IPoIB 层试图在通信的另一端机器打开的队列对的类型：

- 在 **Datagram** 模式中，系统会打开一个不可靠、断开连接的队列对。
  这个模式不支持大于 InfiniBand 链路层的最大传输单元(MTU)的软件包。在传输数据时，IPoIB 层在 IP 数据包之上添加了一个 4 字节 IPoIB 标头。因此，IPoIB MTU 比 InfiniBand link-layer MTU 小 4 字节。因为 **2048** 是一个常见的 InfiniBand 链路层 MTU， **Datagram** 模式中的通用 IPoIB 设备 MTU 为 **2044**。

- 在 **Connected** 模式中，系统会打开一个可靠、连接的队列对。
  这个模式允许消息大于 InfiniBand link-layer MTU。主机适配器处理数据包分段和重新装配。因此，在 **Connected** 模式中，从 Infiniband 适配器发送的消息没有大小限制。但是，由于 **data** 字段和 TCP/IP 标头**字段** 导致 IP 数据包有限。因此， **Connected** 模式中的 IPoIB MTU 是 **65520** 字节。

  连接 模式的性能更高，但会消耗更多内核内存。

虽然系统被配置为使用**Connected**模式，但系统仍然使用 **Datagram** 模式发送多播流量，因为 InfiniBand 交换机和光纤无法在 **Connected** 模式下传送多播流量。另外，当主机没有配置为使用 **连接** 模式时，系统会返回 **Datagram** 模式。

在运行应用程序时，将多播数据发送到接口中的 MTU 时，使用 **Datagram** 模式配置接口，或将应用配置为以以数据报报数据包中的发送大小上限。

6.2. 了解 IPOIB 硬件地址

ipoIB 设备有 **20** 字节硬件地址，它由以下部分组成：

- 前 4 字节是标志和队列对号

- 下一个 8 字节是子网前缀
  默认子网前缀为 **0xfe:80:00:00:00:00:00:00**。设备连接到子网管理器后，设备会更改此前缀以匹配配置的子网管理器。

- 最后一个 8 字节是 InfiniBand 端口的全球唯一标识符(GUID)，附加到 IPoIB 设备

注意

因为前 12 个字节可以更改，因此请不要在 **udev** 设备管理器规则中使用它们。

6.3. 使用 NMCLI 命令配置 IPOIB 连接

nmcli 命令行工具控制 NetworkManager 并使用 CLI 报告网络状态。

先决条件

- InfiniBand 设备已安装在服务器上
- 加载对应的内核模块

流程

1. 创建 InfiniBand 连接，在 **Connected** 传输模式中使用 **mlx4_ib0** 接口，以及最大 MTU **65520** 字节：

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

2. 您还可以将 **0x8002** 设置为 **mlx4_ib0** 连接的 **P_Key** 接口：

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

3. 要配置 IPv4 设置，设置 **mlx4_ib0** 连接的静态 IPv4 地址、网络掩码、默认网关和 DNS 服务器：

```
# 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. 要配置 IPv6 设置，设置 **mlx4_ib0** 连接的静态 IPv6 地址、网络掩码、默认网关和 DNS 服务器：

```
# 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. 激活 **mlx4_ib0** 连接：

```
# nmcli connection up mlx4_ib0
```

6.4. 使用 NETWORK RHEL 系统角色配置 IPOIB 连接

您可以使用 **network** RHEL 系统角色为 IP 通过 InfiniBand (IPoIB)设备远程创建 NetworkManager 连接配置集。例如，通过运行 Ansible playbook，使用以下设置为 **mlx4_ib0** 接口远程添加一个 InfiniBand 连接：

- 一个 IPoIB 设备 - **mlx4_ib0.8002**
- 一个分区密钥 **p_key** - **0x8002**
- 一个静态 **IPv4** 地址 - **192.0.2.1**，子网掩码为 **/24**
- 静态 **IPv6** 地址 - **2001:db8:1::1** ，子网掩码为 **/64**

在 Ansible 控制节点上执行此步骤。

先决条件

- 您已准备好控制节点和受管节点
- 以可在受管主机上运行 playbook 的用户登录到控制节点。
- 用于连接受管节点的帐户具有 **sudo** 权限。
- 名为 **mlx4_ib0** 的 InfiniBand 设备安装在受管节点中。
- 受管节点使用 NetworkManager 配置网络。

流程

1. 创建一个包含以下内容的 playbook 文件，如 ~/**playbook.yml** ：

```
---
- 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
```

   如果您像本例所示设置 **p_key** 参数，请不要在 IPoIB 设备中设置 **interface_name** 参数。

2. 验证 playbook 语法：

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

   请注意，这个命令只验证语法，并不会防止错误但有效的配置。

3. 运行 playbook：

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

验证

1. 在 **managed-node-01.example.com** 主机上显示 **mlx4_ib0.8002** 设备的 IP 设置：

```
# 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. 显示 **mlx4_ib0.8002** 设备的分区密钥(P_Key)：

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

3. 显示 **mlx4_ib0.8002** 设备的模式：

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

其它资源

- **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件
- **/usr/share/doc/rhel-system-roles/network/** directory

6.5. 使用 NM-CONNECTION-EDITOR 配置 IPOIB 连接

nmcli-connection-editor 应用程序使用管理控制台配置和管理 NetworkManager 存储的网络连接。

先决条件

- InfiniBand 设备已安装在服务器上。
- 加载相应的内核模块
- 已安装 **nm-connection-editor** 软件包。

流程

1. 输入命令：

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

2. 点 **+** 按钮添加新连接。
3. 选择 **InfiniBand** 连接类型并点 **Create**。
4. 在 **InfiniBand** 标签页中：

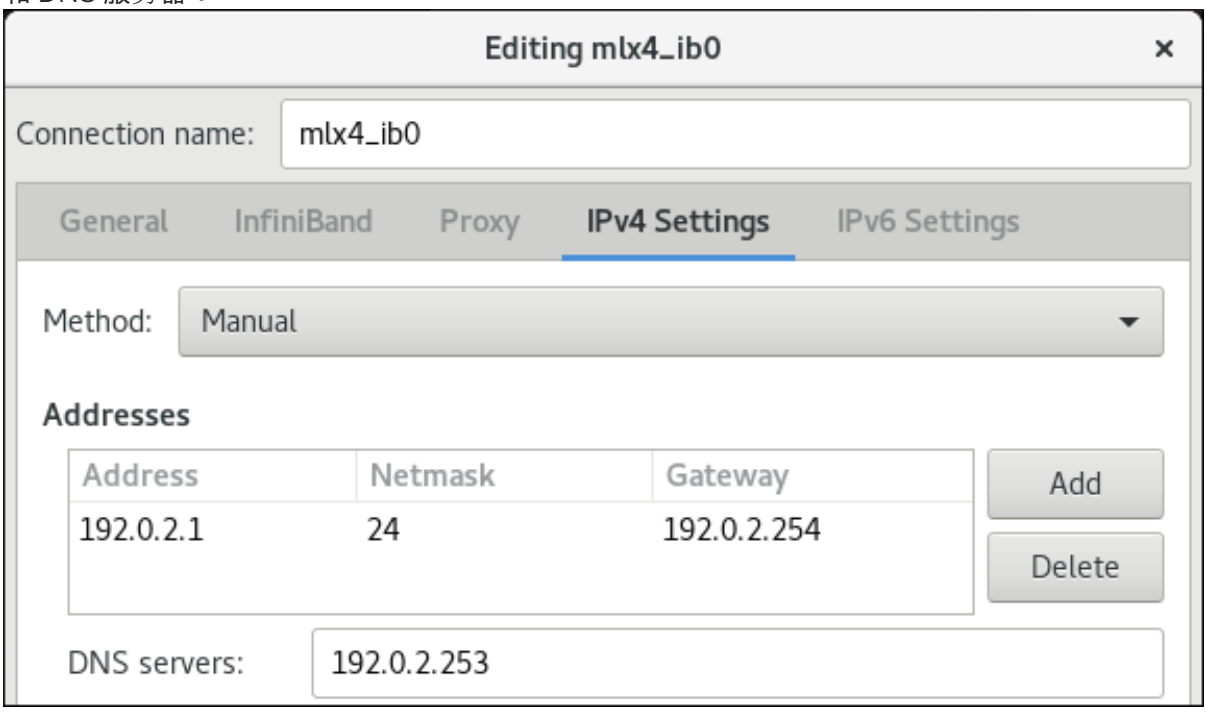
a. 如果您想更改连接名称。

b. 选择传输模式。

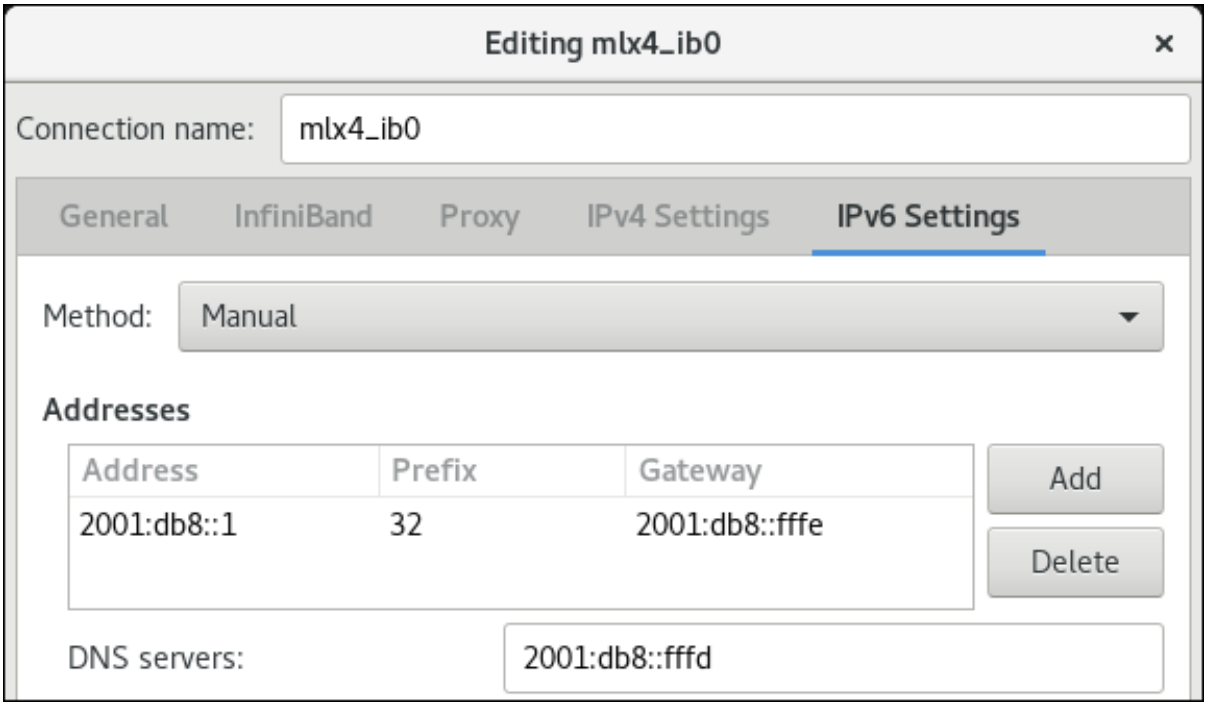
c. 选该设备。

d. 如果需要，设置 MTU。

5. 在 **IPv4 Settings** 选项卡上，配置 IPv4 设置。例如，设置静态 IPv4 地址、网络掩码、默认网关和 DNS 服务器：



6. 在 **IPv6 Settings** 选项卡上，配置 IPv6 设置。例如，设置静态 IPv6 地址、网络掩码、默认网关和 DNS 服务器：



7. 点 **Save** 保存 team 连接。

8. 关闭 **nm-connection-editor**。

9. 您可以设置 **P_Key** 接口。因为 **nm-connection-editor** 中没有此设置，所以您必须在命令行中设置此参数。
   例如，将 **mlx4_ib0** 连接的 **0x8002** 设置为 **P_Key** 接口：

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

第 7 章 测试 INFINIBAND 网络

7.1. 测试早期 INFINIBAND RDMA 操作

InfiniBand 为远程直接内存访问(RDMA)提供低延迟和高性能。

注意

除了 InfiniBand 外，如果您使用基于 IP 的设备，如Internet Wide-area Remote Protocol(iWARP) 或 RDMA over Converged Ethernet (RoCE)或 InfiniBand over Ethernet (IBoE)设备，请参阅：

- 使用 ping 程序测试 IPoIB
- 配置 IPoIB 后使用 qperf 测试 RDMA 网络

先决条件

- 您已配置了 **rdma** 服务。
- 您已安装了 **libibverbs-utils** 和 **infiniband-diags** 软件包。

流程

1. 列出可用的 InfiniBand 设备：

```
# ibv_devices

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

2. 显示 **mlx4_1** 设备的信息：

```
# 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. 显示 **mlx4_1** 设备的状态：

```
# 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** 程序 ping InfiniBand 地址，并作为客户端/服务器运行。

   a. 要在主机上启动服务器模式，请在带有 **-C** InfiniBand 证书颁发机构(CA)名称的端口号 **-P** 上使用 **-S**：

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

   b. 要在另一个主机上启动客户端模式，请使用带有**-L** 本地标识符(LID)的 **-C** InfiniBand 证书颁发机构(CA)名称，在端口号 **-P** 上发送一些数据包**-c**：

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

其它资源

- **ibping(8)** 手册页

7.2. 使用 PING 程序测试 IPOIB

在为 InfiniBand(IPoIB)配置了 IP 后，使用 **ping** 程序发送 ICMP 数据包来测试 IPoIB 连接。

先决条件

- 两个 RDMA 主机在带有 RDMA 端口的同一个 InfiniBand 光纤中连接
- 两个主机中的 IPoIB 接口使用同一子网中的 IP 地址配置

流程

- 使用 **ping** 程序将五个 ICMP 数据包发送到远程主机的 InfiniBand 适配器：

```
# ping -c5 192.0.2.1
```

7.3. 配置 IPOIB 后使用 QPERF 测试 RDMA 网络

qperf 程序根据带宽、延迟和 CPU 使用率来测量两个节点间的 RDMA 和 IP 性能。

先决条件

- 您已在两台主机上都安装了 **qperf** 软件包。
- ipoib 是在两个主机上配置的。

流程

1. 在没有选项作为服务器的主机上启动 **qperf** ：

```
# qperf
```

2. 在客户端中运行以下命令。命令使用客户端中 **mlx4_0** 主机频道适配器的端口 **1** 连接到服务器中分配给 InfiniBand 适配器的 IP 地址 **192.0.2.1**。

   a. 显示主机通道适配器的配置：

```
# 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. 显示可靠的连接(RC)流双向带宽：

```
# 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. 显示 RC 流单向带宽：

```
# 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
```

其它资源

- **qperf(1)** 手册页