



Red Hat Enterprise Linux 9

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

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.

摘要

您可以使用各种协议在企业级配置和管理远程目录内存访问(RDMA)网络和 InfiniBand 硬件。这包括融合以太网上的 RDMA (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 RDMA (NFSoRDMA)协议上的网络文件系统, 作为 RDMA 支持的硬件的原生支持。对于低延迟和高吞吐量连接, 您可以配置通过 InfiniBand (IPoIB)的 IP。

目录

对红帽文档提供反馈	3
第 1 章 了解 INFINIBAND 和 RDMA	4
第 2 章 配置核心 RDMA 子系统	5
第 3 章 配置 IPOIB	7
3.1. IPOIB 通讯模式	7
3.2. 了解 IPOIB 硬件地址	7
3.3. 使用 SYSTEMD 链接文件重命名 IPOIB 设备	8
3.4. 使用 NMCLI 命令配置 IPOIB 连接	9
3.5. 使用 NETWORK RHEL 系统角色配置 IPOIB 连接	10
3.6. 使用 NM-CONNECTION-EDITOR 配置 IPOIB 连接	12
3.7. 配置 IPOIB 后，使用 IPERF3 测试 RDMA 网络	14
第 4 章 配置 ROCE	16
4.1. ROCE 协议版本概述	16
4.2. 临时更改默认 ROCE 版本	16
第 5 章 增加用户可以在系统中固定的内存量	18
第 6 章 在 NFS 服务器上启用 NFS OVER RDMA	19
第 7 章 配置 SOFT-IWARP	21
7.1. IWARP 和 SOFT-IWARP 概述	21
7.2. 配置 SOFT-IWARP	21
第 8 章 INFINIBAND 子网管理器	23

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. 在顶部导航栏中点 **Create**
3. 在 **Summary** 字段中输入描述性标题。
4. 在 **Description** 字段中输入您对改进的建议。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 RDMA（remote direct memory access）技术的一个实现

RDMA 提供了跨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。通过使用 RDMA，数据传输具有高吞吐量、低延迟和低 CPU 使用率。

在典型的 IP 数据传输中，当机器中的某个应用程序向另一台机器上的应用程序发送数据时，在接收层时会出现以下操作：

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. 应用程序将内核本身的内部内存空间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开关以在内核和应用程序上下文间进行切换。这些上下文切换可能会导致 CPU 负载高，并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 协议可以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:

- Internet 广域 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)：通过以太网实现 RDMA 的网络协议

其他资源

- [配置 RoCE](#)

第 2 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。

流程

- 安装 **rdma-core** 软件包：

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

验证

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

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

2. 列出可用的 InfiniBand 设备：

```
# ibv_devices

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

3. 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
```

4. 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: 0xf65214ffe7bcba2
  Link layer: Ethernet
```

5. `ibping` 程序通过配置参数来 ping InfiniBand 地址并以客户端/服务器运行。

- a. 使用 `-C` InfiniBand 证书颁发机构(CA)名称，在主机的端口号 `-P` 上启动服务器模式 `-S`：

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

- b. 启动客户端模式，在主机上使用带有 `-L` Local Identifier (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
```

第 3 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 IP 网络模拟层。这允许现有未修改的应用程序通过 InfiniBand 网络传输数据, 但如果应用程序可以原生使用 RDMA, 则性能较低。



注意

在 RHEL 8 和更新的版本中, 从 ConnectX-4 开始的 Mellanox 设备默认使用增强 IPoIB 模式 (仅数据报)。在这些设备中不支持连接模式。

3.1. IPOIB 通讯模式

IPoIB 设备可在 **Datagram** 或 **Connected** 模式中配置。不同之处在, IPoIB 层试图使用什么类型的队列对在通信的另一端的机器中打开:

- 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 个字节。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 **header** 字段, 存在一个 IP 数据包限制。因此, **Connected** 模式中的 IPoIB MTU 是 **65520** 字节。

Connected 模式性能更高, 但是消耗更多内核内存。

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** 模式下传递多播流量。另外, 当主机没有配置为使用 **Connected** 模式时, 系统会返回 **Datagram** 模式。

当运行一个应用程序, 这个应用程序在接口上发送最大到 MTU 的多播数据时, 将接口设置为处于 **Datagram** 模式, 或对应用程序进行配置, 使其对发送的数据包的大小有一个上限, 以适用于 datagram-sized 的数据包。

3.2. 了解 IPOIB 硬件地址

ipoib 设备有 **20** 个字节硬件地址, 它由以下部分组成:

- 前 4 个字节是标志和队列对号。
- 接下来的 8 个字节是子网前缀
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 8 个字节是 IPoIB 设备附加到的 InfiniBand 端口的全局唯一标识符(GUID)。



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。

3.3. 使用 SYSTEMD 链接文件重命名 IPOIB 设备

默认情况下，内核会命名 IP over InfiniBand (IPoIB) 设备，例如 **ib0**、**ib1** 等等。为避免冲突，请创建一个 **systemd** 链接文件，以创建永久和有意义的名称，如 **mlx4_ib0**。

先决条件

- 您已安装了一个 InfiniBand 设备。

流程

1. 显示设备 **ib0** 的硬件地址：

```
# 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. 为了将 MAC 地址为 **80:00:0a:28:fe:80:00:00:00:00:00:00:f4:52:14:03:00:7b:e1:b1** 的接口命名为 **mlx4_ib0**，创建带有以下内容的 **/etc/systemd/network/70-custom-ifnames.link** 文件：

```
[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
```

此链接文件与 MAC 地址相匹配，并将网络接口重命名为 **Name** 参数中设置的名称。

验证

1. 重启主机：

```
# reboot
```

2. 验证链接文件中指定的 MAC 地址的设备是否已分配给 **mlx4_ib0**：

```
# 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
```

其他资源

- 您系统上的 **systemd.link (5)** 手册页

3.4. 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

- 在服务器上安装 InfiniBand 设备
- 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. 设置 **P_Key**，例如：

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

3. 配置 IPv4 设置：

- 要使用 DHCP，请输入：

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

如果 **ipv4.method** 已设置为 **auto**（默认），请跳过这一步。

- 要设置静态 IPv4 地址、网络掩码、默认网关、DNS 服务器和搜索域，请输入：

```
# 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. 配置 IPv6 设置：

- 要使用无状态地址自动配置(SLAAC)，请输入：

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

如果 **ipv6.method** 已设置为 **auto**（默认），请跳过这一步。

- 要设置静态 IPv6 地址、网络掩码、默认网关、DNS 服务器和搜索域，请输入：

-

```
# 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
```

- 要在配置文件中自定义其他设置，请使用以下命令：

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

用空格或分号将值括起来。

- 激活配置文件：

```
# nmcli connection up mlx4_ib0
```

验证

- 使用 **ping** 工具将 ICMP 数据包发送到远程主机的 InfiniBand 适配器，例如：

```
# ping -c5 192.0.2.2
```

3.5. 使用 NETWORK RHEL 系统角色配置 IPOIB 连接

您可以使用 IP over InfiniBand (IPoIB)，通过 InfiniBand 接口发送 IP 数据包。要配置 IPoIB，请创建一个 NetworkManager 连接配置文件。通过使用 Ansible 和 **network** 系统角色，您可以自动执行此过程，并在 playbook 中定义的主机上远程配置连接配置文件。

您可以使用 **network** RHEL 系统角色配置 IPoIB，如果 InfiniBand 的父设备的连接配置文件不存在，角色也可以创建它。

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。
- 用于连接到受管节点的帐户具有 **sudo** 权限。
- 一个名为 **mlx4_ib0** 的 InfiniBand 设备被安装在受管节点上。
- 受管节点使用 NetworkManager 配置网络。

流程

- 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: 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

```

示例 playbook 中指定的设置包括以下内容：

type: <profile_type>

设置要创建的配置文件的类型。示例 playbook 创建两个连接配置文件：一个用于 InfiniBand 连接，一个用于 IPoIB 设备。

parent: <parent_device>

设置 IPoIB 连接配置文件的父设备。

p_key: <value>

设置 InfiniBand 分区密钥。如果您设置了此变量，请不要在 IPoIB 设备上设置 **interface_name**。

transport_mode: <mode>

设置 IPoIB 连接操作模式。您可以将此变量设置为 **datagram**（默认）或 **connected**。

有关 playbook 中使用的所有变量的详情，请查看控制节点上的 **/usr/share/ansible/roles/rhel-system-roles.network/README.md** 文件。

2. 验证 playbook 语法：

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

请注意，这个命令只验证语法，不会防止错误，但保护有效配置。

3. 运行 playbook：

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

验证

1. 显示 **mlx4_ib0.8002** 设备的 IP 设置：

```

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

```
# 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. 显示 `mlx4_ib0.8002` 设备的模式 :

```
# 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
```

其他资源

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

3.6. 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

- 在服务器上安装一个 InfiniBand 设备。
- 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. 点击 **+** 按钮来添加一个新的连接。
3. 选择 **InfiniBand** 连接类型并点 **Create**。
4. 在 **InfiniBand** 选项卡中 :
 - 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** 选项卡中，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：

The screenshot shows the 'Editing mlx4_ib0' dialog box with the 'IPv4 Settings' tab selected. The 'Connection name' is 'mlx4_ib0'. The 'Method' is 'Manual'. The 'Addresses' table is as follows:

Address	Netmask	Gateway
192.0.2.1	24	192.0.2.254

The 'DNS servers' field contains the value '192.0.2.253'.

6. 在 **IPv6 设置** 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：

The screenshot shows the 'Editing mlx4_ib0' dialog box with the 'IPv6 Settings' tab selected. The 'Connection name' is 'mlx4_ib0'. The 'Method' is 'Manual'. The 'Addresses' table is as follows:

Address	Prefix	Gateway
2001:db8::1	32	2001:db8::fffe

The 'DNS servers' field contains the value '2001:db8::fffd'.

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，要将 **0x8002** 设置为 **mlx4_ib0** 连接的 **P_Key** 接口：

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

3.7. 配置 IPOIB 后，使用 IPERF3 测试 RDMA 网络

在以下示例中，大型缓冲区大小用于执行 60 秒测试，以测量最大吞吐量，并使用 **iperf3** 工具充分利用两台主机之间的带宽和延迟。

先决条件

- 您已在两个主机上配置了 IPoIB。

流程

1. 要在系统中运行 **iperf3** 作为服务器，定义一个时间间隔，以提供定期带宽更新 **-i** 以侦听，作为一个服务器 **-s**，等待客户端连接的响应：

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

2. 要作为客户端在另一个系统上运行 **iperf3**，请定义一个间隔，提供定期带宽更新 **-i** 连接到侦听的服务器 **-c** 连接到 IP 地址 **192.168.2.2** 和 **-t** 秒（以秒为单位）：

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

3. 使用以下命令：

- a. 在作为服务器的系统中显示测试结果：

```
# 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. 在作为客户端的系统中显示测试结果：

```
# 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
```

其他资源

- 系统上的 **iperf3** man page

第 4 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。融合以太网上的 RDMA(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，一些硬件提供商是 Mellanox、Broadcom 和 QLogic。

4.1. ROCE 协议版本概述

RoCE 是一个网络协议，它允许通过以太网进行远程直接访问(RDMA)。

以下是不同的 RoCE 版本：

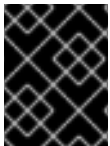
RoCE v1

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 设置客户端和服务端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。

RoCE v1 在 Data Link layer(Layer 2)工作，它只支持同一网络中的两台计算机的通信。默认情况下，RoCE v2 可用。它适用于网络层(Layer 3)。RoCE v2 支持数据包路由，以便提供与多个以太网的连接。

其他资源

- [临时更改默认 RoCE 版本](#)

4.2. 临时更改默认 ROCE 版本

在客户端中使用 RoCE v2 协议，并不支持服务器上的 RoCE v1。如果服务器中的硬件只支持 RoCE v1，请将 RoCE v1 的客户端配置为与服务器进行通信。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

这里描述的更改将保持有效，直到重启主机为止。

先决条件

- 客户端使用具有 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
```

第 5 章 增加用户可以在系统中固定的内存量

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交换空间。如果用户固定太多内存，系统会耗尽内存，并且内核会终止进程来释放更多内存。因此，内存固定是一种特权操作。

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固定。

流程

- 以 **root** 用户身份，使用以下内容创建文件 `/etc/security/limits.conf`：

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

验证

- 在编辑 `/etc/security/limits.conf` 文件后，以 **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。
- 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。

其他资源

- 您系统上的 **limits.conf (5)** 手册页

第 6 章 在 NFS 服务器上启用 NFS OVER RDMA

远程直接内存访问(RDMA)是一种协议，它使客户端系统将数据直接从存储服务器的内存传到其自己的内存。这提高了存储吞吐量，降低了服务器和客户端之间数据传输的延迟，并减少了两端的 CPU 负载。如果 NFS 服务器和客户端都通过 RDMA 连接在一起，则客户端可以使用 NFSoRDMA 来挂载导出的目录。

先决条件

- NFS 服务正在运行且已配置
- 服务器上安装了 InfiniBand 或 RDMA over Converged Ethernet (RoCE)设备。
- IP over InfiniBand (IPoIB)已在服务器上配置，InfiniBand 设备已分配了一个 IP 地址。

流程

1. 安装 **rdma-core** 软件包：

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

2. 如果软件包已安装，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，NFS over RDMA 使用端口 20049。如果要使用不同的端口，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 硬件的客户端上执行以下步骤：

- 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/
```

如果您设置了与默认端口号(20049)不同的端口号，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` 选项挂载了：

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

其他资源

- [配置 InfiniBand 和 RDMA 网络](#)

第 7 章 配置 SOFT-IWARP

远程直接内存访问(RDMA)使用以太网上的多个库和协议，如 iWARP、Soft-iWARP 提高性能并辅助编程接口。



重要

Soft-iWARP 只是一个技术预览功能。技术预览功能不受红帽产品服务等级协议 (SLA) 支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。有关红帽技术预览功能支持范围的详情，请参考 <https://access.redhat.com/support/offerings/techpreview/>。

7.1. IWARP 和 SOFT-IWARP 概述

远程直接内存访问(RDMA)使用 iWARP over Ethernet 进行聚合，以及通过 TCP 的低延迟数据传输。通过使用标准的以太网交换机和 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设施的灵活性。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。

Soft-iWARP (siw) 是一个基于软件的、用于 Linux 的 iWARP 内核驱动器和用户程序库。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。

7.2. 配置 SOFT-IWARP

Soft-iWARP (siw) 实现通过 Linux TCP/IP 网络堆栈的 iWARP 远程直接访问(RDMA)传输。它可以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：

```
# dnf 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. 添加一个新的名为 **siw0** 的 **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 8 章 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。在那种情况下，当主子网管理器出现故障时，另外一个作为从网管理器的系统会接管。

Red Hat Enterprise Linux 提供了 **OpenSM**， InfiniBand 子网管理器的一种实现。但是，**OpenSM** 的功能是有限的，且没有活跃的上游开发。通常，InfiniBand 交换机中嵌入的子网管理器提供更多功能，并支持最新的 InfiniBand 硬件。详情请查看 [安装和配置 OpenSM InfiniBand 子网管理器](#)。