Helium DPU SNIC

User Manual

Product Model	EC2004Y/EC2002P
Product Version	V1.0R1
Document Version	V1.1
Intended Audience	User/Maintainer/Developer

Contents

Figure Catalog

Table Directory

About This Document:

1. provides hardware parameters and specifications related to the Helium DPU SNIC, software and firmware details required for the smart network card.

2. provides how to build the Helium DPU SNIC and the corresponding Host step by step, so that the entire network card system can work normally.

3. provides detailed test methods and steps related to DPDK traffic test, virtual machine vhost-user traffic and live migration based on the Helium DPU SNIC and the Host.

Intended Audience:

This document is intended for Helium DPU SNIC users, installers and developers.

Technical Support:

TEL : 400-098-9811

EMAIL : support@asterfusion.com

URL : https://support.asterfusion.com/

Revision History：

Revision Time	Revision Content	Note
2021/12/31	Create	V1.0
2022/02/16	Update the EC2002P 100G optical module enable command	V1.0
2022/03/11	1.Add the description of the 6-pin power connector in section 4.1 2.Add EC2002P 1 minute 4 operations in Appendix B 3.Add steps for Installing the Helium System in Appendix C 4.Modify the Document Format	V1.0
2022/06/22	1.Add EC2002P restore to 100G from 4*25G in Appendix B	V1.1

1 Preface

1.1 References

Serial Number	Reference Documents
1	https://pcisig.com/specifications
2	https://www.dpdk.org/
3	http://www.openvswitch.org/

Table 1-1 References

1.2 Abbreviations

Abbreviations	EN	CN
OvS	Open vSwitch	
PCIe	Peripheral Component Interconnect Express	
PCIe ACS	PCIe Access Control Service	

Table 1-2 Abbreviations

1.3 Glossary

Term	Meaning
Host	Host of Helium DPU SNIC
SDP	Virtual port of Helium DPU SNIC
virtio-forwarder	Used for Host vhost-user

Table 1-3 Glossary

1.4 Command Description

In this document, all commands displayed in gray indicate that they can be run directly on the device. The sample is as follows:

```
# lspci -nn -d 177d:b200
```

1.5 Naming Specifications for Helium Product Packages

The naming specifications of the Helium software package are as follows:

Product - (Model) - Function - Version - (Special Hardware Architecture)

The explanations are as follows, where parentheses indicate non-essential items:

- Product : Helium
- Model : EC2002P/EC2004Y - This parameter is not required if the software package is generic
- Function : Describes the functionality provided or available by the package
- Version : Release version of the software package. The version format is Vx. yRz

  x - major version number, used for incompatibility changes

  y - minor version number, used to identify new or modified features

  z - revised version, used to identify problem fixes
- Special Hardware Architecture : This parameter is used to distinguish between different hardware architectures/platforms. Otherwise, this parameter is not required

2 Introduction

2.1 Product Overview

Helium is a 25GE/100GE Ethernet smart network card based on a high-performance DPU chip independently developed by Asterfusion. It supports standard PCIe*16 Gen3.0/Gen-4.0 interface and can be easily plugged into the PCIe slots of commercial data center servers or PC.

While Asterfusion provides Helium hardware, it also provides Linux kernel operating systems and development kits. The customer's various DPDK applications, VPP applications and ordinary Linux driver applications originally running on the x86 server can be quickly transplanted to the Helium smart network card with a simple compilation.This combination of VPP, DPDK and Linux technologies provides a powerful platform for easy and rapid expansions on new or emerging business applications and hence allows cloud data center administrators to build up their highly-efficient, highly-intelligent and highly-flexible networks operations while at the same time, minimize computing resource consumption in their data center servers and optimize their overall cost of operations.

2.2 Physical Drawing of Helium

The following table lists the meanings of the notes on the actual drawings of the Helium EC2004Y and EC2002P:

Helium	Note 1	Note 2	Note 3	Note 4	Note 5
EC2004Y	serial port	25G port	Management network port	25G port LED	Panel Port Number From left to right 1,2,3,4
EC2002P	serial port	100G port	Management network port	100G port LED	Panel Port Number From left to right 1,2

Table 2-1 Label Meaning Table of Physical Products

Figure 2-1 Helium - EC2004Y Physical Figure

Figure 2-2 Helium - EC2002P Physical Figure

2.3 Product Specifications

Category		Helium EC2004Y	Helium EC2002P
Interface	Network Interface	4*25GE SFP28	2*100GE QSFP28
	Host Interface	PCIe*16 Gen3.0/Gen4.0	
	Management Interface	1*Console Micro USB, 1*GE RJ45 OOB Port	
Power & Dimension	Power Consumption	60W	
	Dimension(W*H*D)	111.15mm*21.8mm *167.65mm	111.15mm*21.8mm *184.16mm
	Weight (kg)	0.8kg	
	Operating Temperature	0~35℃	
	Operating Humidity	10%~90%(non-condensing)	
Core CPU	Architecture	DPU	
	Part Number	CN96XX	
	Number of Processor Cores	24	
	Core Clock Frequency	1.8GHz	
	Number of CPU Part	1	
	Cache Capacity (MB)	L2 5MB, L3 14MB	
Memory & Storage	Memory Capacity	Single memory 8GB, 16GB or 32GB, configurable up to 2	
	Memory Type	DDR4 ECC SODIMM	
	Memory Capacity Expansion	64GB	
	Flash Storage (GB)	64GB EMMC 5.1	

Table 2-2 Product Specification Sheet

3 Installation Process

3.1 Scope of Application

This document is applicable to the following scenarios:

Scene	Reference
Install the driver and program for the first time	6.1, **Error! Reference source not found.**, **Error! Reference source not found.**, 7.1
Upgrade drivers and programs	6, 7
Uninstall drivers and programs	**Error! Reference source not found.**, 7.2
Install the Helium SNIC OS	Appendix C : Helium NIC OS Installation
Set the port rate of Helium	5.3.2, Appendix B :

Table 3-1 Scope of Application

3.2 Installation Steps

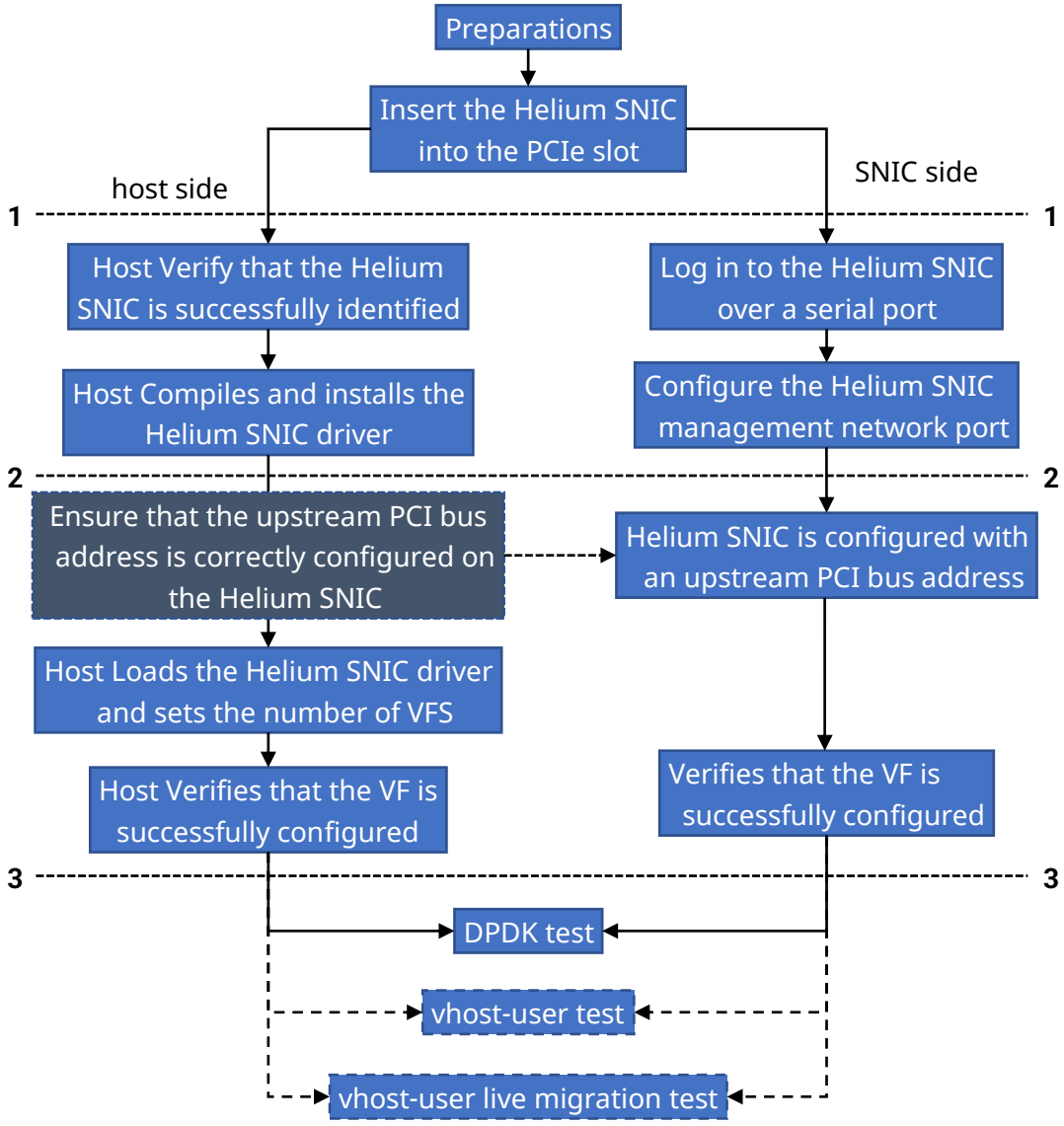


Figure 3-1 Installation Flow Chart

Step1: Insert Helium into the PCIe slot of the Host

Step2: Host side: Confirm that Helium is successfully recognized, Compiles and installs the driver

Helium side: Configure network related for subsequent login operations

Step3: Host side: Get the PCI bus address of Helium

Helium side: Set the PCI bus address obtained by the Host side

Host side: Load Helium driver and confirm

Helium side: Confirm that the loading was successful

Step4: Run the test program on the Host side and Helium side

3.3 Software Configuration List

Device	Software Package Name	Content description
Host	Helium-Driver-Vx.yRz.tar.gz	PF driver for Helium
	Helium-DPDK19.11-Vx.yRz.tar.gz	VF driver and dpdk demo
	Helium-VirtioForwarder-Vx.yRz-intel-hsw.bin Helium-VirtioForwarder-Vx.yRz-intel-ivb.bin	vhost-user driver(intel hsw/ivb cpu)
	Helium-VPP-Vx.yRz.tar.gz	vpp for Host
	Helium-HostAutoTest-Vx.y.Rz.sh	Automated test scripts for Host
Helium	Helium-Debian10-Vx.yRz.tar	Linux OS for Helium
	Helium-OvS-Vx.yRz.bin	OvS for Helium
	Helium-busybox-Vx.yRz.img	busybox image for Helium
	Helium-EC2004Y-uboot-Vx.yRz.img Helium-EC2002P-uboot-Vx.yRz.img	uboot image for Helium
	Helium-DPDK19.11-Vx.yRz.tar.gz	VF driver and dpdk demo
	Helium-VPP-Vx.yRz.tar.gz	vpp for Helium
	Helium-AutoTest-Vx.y.Rz.sh	Automated test scripts for Helium
	Helium-OpenSSL-Vx.yRz.tar.gz	OpenSSL for Helium

Table 3-2 Helium Software Configuration List

Please register and download related image files from our support website (https://support.asterfusion.com/)

3.4 Compatibility List

The supported Host servers and their versions are as follows:

Linux OS	Kernel	Gcc Version	Host
CentOS7.9	4.15.18	Gcc version 9.3.1 20200408 (Red Hat 9.3.1-2) (GCC) Gcc version 4.8.5 20150623 (Red Hat 4.8.5-44) (GCC)	Inspur NF5270M3 DELL R740
CentOS8.5	4.18.0	Gcc version 8.5.0 20210514 (Red Hat 8.5.0-4) (GCC)	
Ubuntu18.04	4.15.18	Gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)	
Ubuntu20.04	5.4.0-81-generic	Gcc version 9.3.0 (Ubuntu 9.3.0-17ubuntu1~20.04)	

Table 3-3 Compatibility List

4 Preparation before Installation

4.1 Confirmation of Hardware and Network Environment

Serial Number	Description
1	Host Hardware installation has been completed, including memory modules, power supply units (psus), fans, etc.
2	Host device has met all the requirements for power on
3	Helium is connected to the terminal Host and can be accessed through the serial port
4	Helium and the Host are connected to the same network to ensure network connection

The following figure shows the 6-pin power connector for the Helium DPU SNIC: the three upper pins are GND pins, and the three lower pins are 12V VCC pins.

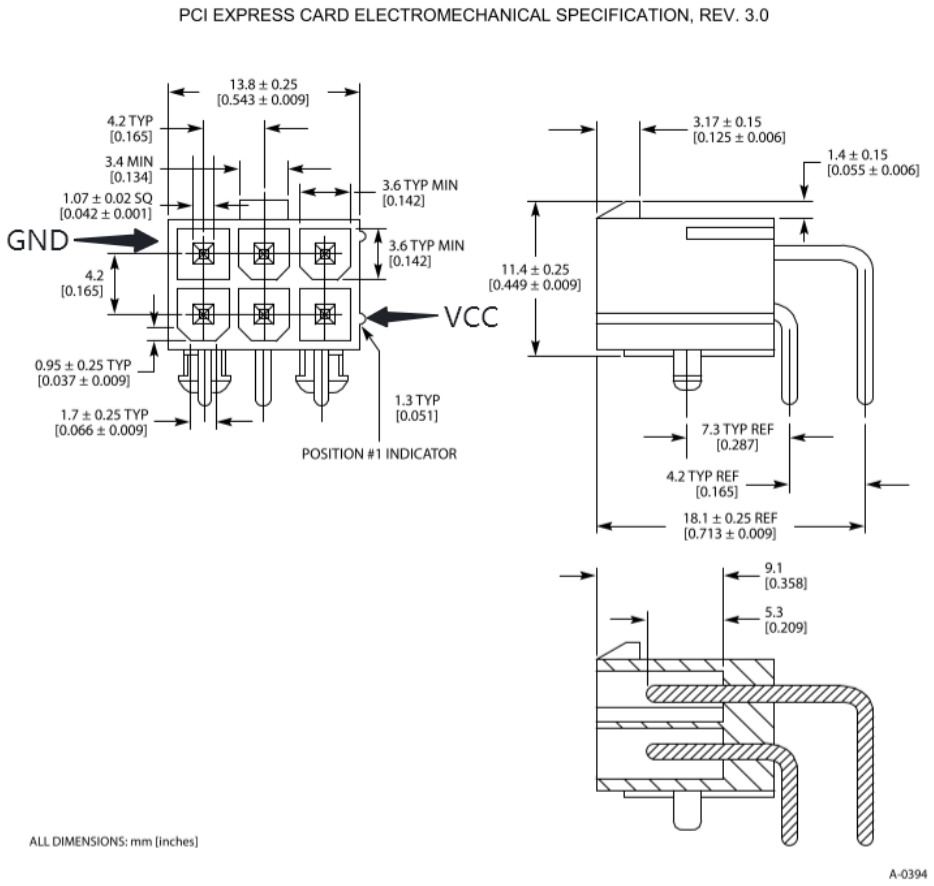


Figure 4-1 6PIN Power Connector of Helium

Note: The PCIe power supply must be the same as the external ATX power supply. That is, if the purchased ATX power supply is used, the PCIe power supply cannot be used. If it is plugged into a server, the 6-pin power socket of the NETWORK adapter needs to be powered off from the mainboard. By default, use the power supply on the server.

4.2 Preparation of Static Information

Content	Source	Explanation
MAC address to be allocated for deployment	Uniformly distributed from the factory or modified on site	Uniformly distributed from the factory or modified on site
IP address of the device to be deployed	Factory default assignment 192.168.2.100 or modified on site	For remote access and software updates, use the default IP address if the device is not online

5 Software Configuration of Helium

The Helium DPU SNIC has been installed with the Debian10 OS by default. The optional firmware is FusionNOS (with the standard convergence and diversion function) or OvS. To update or reinstall the Helium OS, refer to Appendix C : Helium NIC OS Installation.

5.1 Helium Login

There are three login modes for Helium: serial port, management network port, and Host mgmt_net.

5.1.1 Serial Port

User can configure the serial port through "HyperTerminal" application under Windows system as follows:

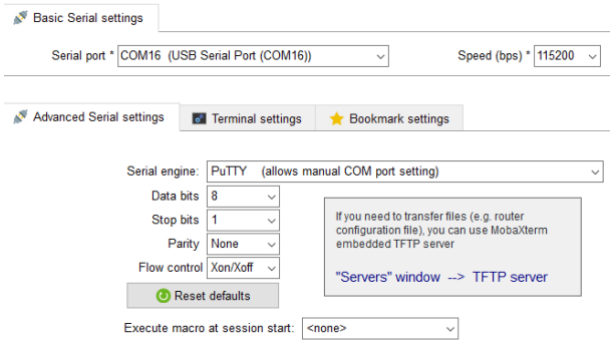


Figure 5-1 Connect to the Device through HyperTerminal

The configuration for Minicom in Linux is as follows:

Figure 5-2 Configuration for Minicom

System user name:root

password:passok

5.1.2 Management Network Port

Board type	Helium EC2004Y	Helium EC2002P
Management network port	eth4	eth2

The default ip address of the management network port is 192.168.2.100. You can use SSH to log in to the Helium.

You can also configure ip addresses by yourself: You can log in to the device through the serial port, set the ip address of the management network port, and configure SSH permissions,etc.

(For details about the network configuration, see 5.3 Network Configuration)

5.1.3 Host mgmt_net

The Host mgmt_net :

Host interacts with the Helium DPU SNIC through mvmgmt0 port.

You can access the Helium DPU SNIC through SSH on the Host and vice versa.

Host mgmt_net requires:

On the Helium: Correctly configure the upstream PCI address, load the mgmt_net driver, and configure the mvmgmt0 port address.

On the Host: Correctly load mgmt_net driver, and configure the mvmgmt0 port address.

（For details, see steps 2 and 3 in 7.1 Enable Virtual Port on the Host）

5.2 PCI Address of Helium

The corresponding PCI addresses of the Helium panel ports and 64 virtual ports are as follows:

Pci Device	Vendor Class Id	Helium-EC2004Y	Helium-EC2002P
0002:02:00.0	177d:a063	panel ports 4 - eth0	panel ports 2 - eth0
0002:03:00.0	177d:a063	panel ports 3 - eth1	panel ports 1 - eth1
0002:04:00.0	177d:a063	panel ports 2 - eth2	N/A

0002:05:00.0	177d:a063	panel ports 1 - eth3	N/A
0002:0f:00.2	177d:a0f7	SDP0, VF0 of Host	
… …	177d:a0f7	… …	
0002:0f:00.7	177d:a0f7	SDP5, VF5 of Host	
0002:0f:01.0	177d:a0f7	SDP6, VF6 of Host	
… …	177d:a0f7	… …	
0002:0f:01.7	177d:a0f7	SDP13, VF13 of Host	
0002:0f:02.0	177d:a0f7	SDP14, VF14 of Host	
… …	177d:a0f7	… …	
0002:0f:02.7	177d:a0f7	SDP21, VF21 of Host	
… …	177d:a0f7	… …	
0002:0f:08.0	177d:a0f7	SDP62, VF62 of Host	
0002:0f:08.1	177d:a0f7	SDP63, VF63 of Host	

Table 5-1 PCI Address of Helium

The numbers of virtual port SDP are generated only after the VFs are enabled on the Host

(For details about how to enable the VF on the Host, see 7.1 Enable Virtual Port on the Host)

5.3 Network Configuration

5.3.1 Static IP Configuration

The default ip address of the management port is 192.168.2.100. For details, see the following steps:

Modify the default ip to 192.168.0.212

Step1: Edit /etc/network/interfaces

vim /etc/network/interfaces:

```
auto eth4
iface eth4 inet static
address 192.168.0.212
netmask 255.255.255.0
gateway 192.168.0.1
broadcast 192.168.0.255
```

Step2: Restart network service

```
# sudo /etc/init.d/networking restart
```
or `# sudo service networking restart`

5.3.2 Port Rate Settings for Panel Ports

For EC2002P, 100G port optical module is disabled by default. Therefore, the port cannot work properly. You can run the command 'i2Cset -y 1 0x30 9 3' on the EC2002P to enable the 100G port.

For Helium, the rate of a port on the panel can be changed using the ethtool:

Panel Port Rate	Ethtool Argument
1G	0x20000000000
10G	0x100000
25G	0x200000000
40G	0x4000000
100G	0x8000000000

Table 5-2 Ethtool Argument for Port Rate Settings

The following uses eth1 on the EC2004Y panel as an example:

Modified to 25G, `# ethtool -s eth1 advertise 0x200000000`

Modified to 10G, `# ethtool -s eth1 advertise 0x100000`

Modified to 1G(optical/electrical module), `# ethtool -s eth1 advertise 0x20000000000`

Details for EC2002P 100G ports 1 divided into 4, see Appendix B : .

6 Host Driver Dependencies and Compilation

The following example installation commands are based on CentOS7.9.2009 kernel 4.15.18 (ACS support)

The current version of Host driver supports the following protocols:

Protocols	Support
SR-IOV	Y
vhost-user	Y
vhost-net	N

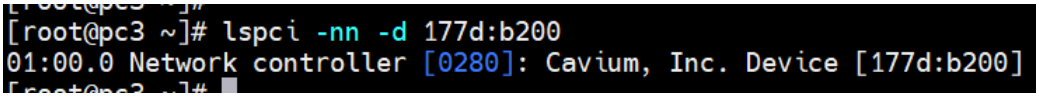
6.1 Helium Identification on the Host

Start the Host system after installing the Helium.

Run the following command under Linux system to check whether the Ethernet DPU is identified.

```
# lspci -nn -d 177d:b200
```

The correct results are as follows, including more than one device.



(If it cannot be identified, please confirm that the PCIE slot of the Helium is plugged correctly, then power off and reboot)

6.2 Host-side Driver Dependencies

To install the Helium driver on the Host side, the following dependencies must be met:

- CPU supports hardware virtualization, the board supports IOMMU, and is opened in BIOS
- The mainboard supports ACS

For motherboards that do not support ACS, you can install a kernel version that supports ACS Override Patch from the following links: https://queuecumber.gitlab.io/linux-acs-override/

> The Access Control Service (ACS) Extended Capability is introduced in the PCIe standard and is used to Control the normal routing (upward submission) for the received Transaction Layer Packets (TLP). Blocking or redirecting forwarding features. This feature prevents peer-to-peer data transmission between PCIe devices. Like a traditional PCI bus, the data packets sent by PCIe devices are sent up to the root of the PCIe device tree. In this way, the IOMMU can intercept all data packets sent by PCIe devices.

- CPU needs to set the kernel startup parameter

  Step1: Edit /etc/default/grub ,add the flowings to GRUB_CMDLINE_LINUX_DEFAULT:

  ```
  intel_iommu=on iommu=pt pci=assign-busses pcie_acs_override=downstream
  ```

> IOMMU is a common name for Intel VT-D and AMD-VI.
> For Intel CPU(VT-D), run intel_iOMMU =on. For AMD CPUS (AMD-VI), use amd_iOMMU =on. (Currently only tested and verified on Intel)

  Step2: Update kernel grub

  Ubuntu: `# update-grub`

  CentOS :

  legacy:


```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

uefi:

```
# grub2-mkconfig -o /boot/efi/EFI/centos/grub.cfg
```

uefi/legacy : dmesg | grep -i efi // exits for efi Or check whether the '/sys/firmware/efi ' folder exists //exits for efi

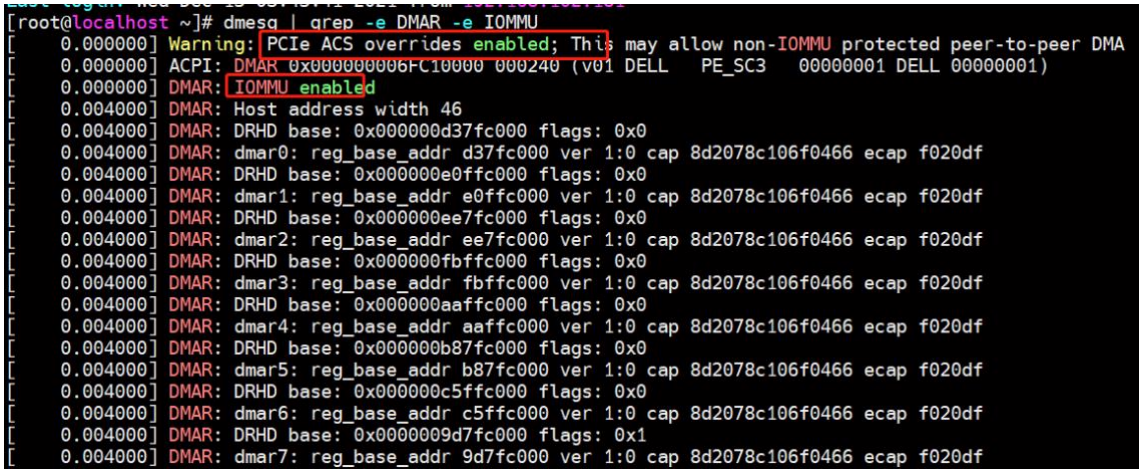
Step3: Reboot the device to take effect

```
# reboot
```

Step4: Restart to check whether the configuration takes effect

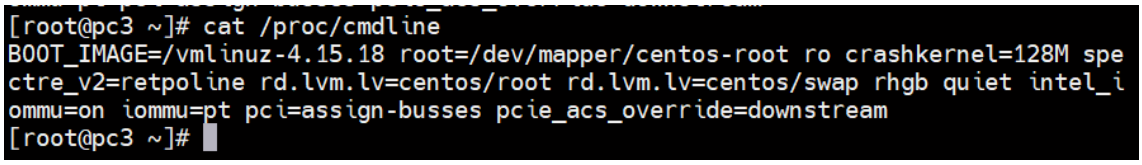
Make sure iOMMU is enabled and ACS is supported

```
# dmesg | grep -e DMAR -e IOMMU
```



Verify that the kernel changes are correct and take effect

```
# cat /proc/cmdline
```



6.3 Host-side Driver Compilation

Step1: Download the driver source package Helium-Driver-Vx.yRz.tar.gz

Step2: Decompress the driver source package and compile it

```
# tar -zxvf Helium-Driver-V1.0R1.tar.gz
# cd Helium-ep-driver
# make
```

Note: If you are prompted that there are no related dependencies, please install them separately

Step3: Check whether the driver compilation is successful

find . -name *.ko

```
[root@pc3 ~/code/sf5000-c/HOST_EC96/Helium-ep-driver]# find . -name *.ko
./drivers/legacy/modules/driver/bin/octeon_drv.ko
./drivers/legacy/modules/driver/bin/octnic.ko
./drivers/legacy/modules/driver/bin/octdbg.ko
./drivers/legacy/modules/driver/src/host/linux/kernel/drv/octnic/octdbg.ko
./drivers/legacy/modules/driver/src/host/linux/kernel/drv/octnic/octnic.ko
./drivers/legacy/modules/driver/src/host/linux/kernel/drv/octeon_drv.ko
./drivers/mgmt_net/mgmt_net.ko
```

6.4 Host-side Driver Update

Step1: Refer to **Error! Reference source not found. Error! Reference source not found.**

Step2: Uninstall the currently loaded driver (take care that uninstalling driver will cause the board to reboot. Wait for 2min after uninstalling)

Step3: Reload the driver (refer to Step 2 and step 3 of 7.1 Enable Virtual Port on the Host)

7 Configuration of Virtual Port on the Host

7.1 Enable Virtual Port on the Host

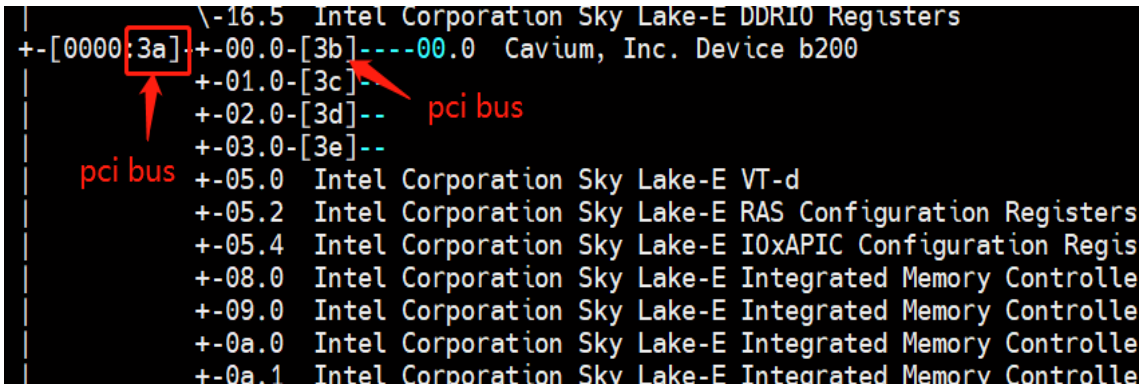
Step1: On the Host side, check the root pci bus address of the Helium DPU SNIC

lspci -tv // Search the device of the corresponding Helium DPU SNIC

for example:

the root pci bus address of Helium on this Host is 3a

+-[0000:3a]-+-00.0-[3b]--+-00.0



Step2: Configure the root pci bus address and mvmgmt0 port on the Helium

Log in to the Helium through SSH or serial port and run the following commands:

```
# ifconfig mgmt_net down 2>/dev/null
# echo 0 > /sys/bus/pci/devices/0000\:05\:00.0/sriov_numvfs
# rmmod mgmt_net 2>/dev/null
# rmmod pcie_ep
# rmmod dpi_dma
# echo 1 > /sys/bus/pci/devices/0000\:05\:00.0/sriov_numvfs
# modprobe dpi_dma
# modprobe pcie_ep     host_sid=0x33a00 pem_num=0 epf_num=0
```

configure port mvmgmt0 :

```
# modprobe mgmt_net
# ifconfig mvmgmt0 12.12.12.12
# ifconfig mvmgmt0 up
```

Confirm that the configuration is successful:

```
# ifconfig mvmgmt0
```

```
root@OCTEONTX:~# ifconfig mvmgmt0
mvmgmt0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9600
        inet 12.12.12.12  netmask 255.0.0.0  broadcast 12.255.255.255
        inet6 fe80::a079:34ff:febb:47f2  prefixlen 64  scopeid 0x20<link>
        ether a2:79:34:bb:47:f2  txqueuelen 1000  (Ethernet)
        RX packets 44  bytes 11852 (11.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 24  bytes 1904 (1.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@OCTEONTX:~#
```

Step3: Load the driver on the Host and configure the virtual port

Load the driver compiled in Chapter 5 on the Host and configure the number of virtual ports

```
# insmod \
    ./drivers/legacy/modules/driver/src/host/linux/kernel/drv/octeon_drv.ko\
    num_vfs=4     // num_vfs can be customized and supports up to 64
```

Configure mvmgmt0 port:

CentOS7.9:

create configure file : /etc/sysconfig/network-scripts/ifcfg-mvmgmt0

config static ip address : 12.12.12.1

vim /etc/sysconfig/network-scripts/ifcfg-mvmgmt0 :

TYPE=Ethernet

```
PROXY_METHOD=none
BROWSER_ONLY=no
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
NAME=mvmgmt0
DEVICE=mvmgmt0
ONBOOT=yes
IPADDR=12.12.12.1
NETMASK=255.0.0.0
GATEWAY=12.1.1.1
```

insmod mgmt_net.ko:

```
# insmod ./drivers/mgmt_net/mgmt_net.ko
```

up mvmgmt0 port:

```
# ifup /etc/sysconfig/network-scripts/ifcfg-mvmgmt0
```

Ubuntu:

```
# insmod ./drivers/mgmt_net/mgmt_net.ko
# ifconfig mvmgmt0 12.12.12.1
# ifconfig mvmgmt0 up
```

Confirm that the configuration is successful:

```
# ifconfig mvmgmt0
```

```
[root@pc3 /opt/tftp]# ifconfig mvmgmt0
mvmgmt0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9600
        inet 12.12.12.1  netmask 255.0.0.0  broadcast 12.255.255.255
        inet6 fe80::80ca:7eff:fefc:876d  prefixlen 64  scopeid 0x20<link>
        ether 82:ca:7e:fc:87:6d  txqueuelen 1000  (Ethernet)
        RX packets 24  bytes 1904 (1.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 44  bytes 6234 (6.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

After configuring mvmgmt0 correctly, you can log in to the Helium through SSH directly. As an example, the mvmgmt corresponding to the Ethernet DPU is 12.12.12.12.

ssh root@12.12.12.12 // You can log in to the Ethernet DPU

Note:

Set the ip of mvmgmt0 on Host, Static configuration is recommended. Dynamic mode currently under CentOS7.9: After the driver is loaded, CentOS will continuously try to assign DHCP addresses to mvmgmt0 and fail. CentOS7.9 will set the nic state to disconnected, which will cause the manually configured mvmgmt0 address to fail (this can be solved by configuring ip multiple times).

Step4: Check that the loading is complete on the Host

```
# lspci -nn -d 177d:b203
```

or

```
# lspci | grep b203
```

The correct result is as follows, ‘num_vfs’ VF ports will be created.

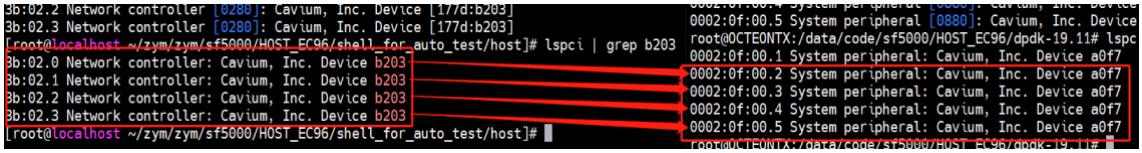
```
[root@localhost ~/zym/zym/sf5000/HOST_EC96/shell_for_auto_test/host]# lspci -nn -d 177d:b203
3b:02.0 Network controller [0280]: Cavium, Inc. Device [177d:b203]
3b:02.1 Network controller [0280]: Cavium, Inc. Device [177d:b203]
3b:02.2 Network controller [0280]: Cavium, Inc. Device [177d:b203]
3b:02.3 Network controller [0280]: Cavium, Inc. Device [177d:b203]
[root@localhost ~/zym/zym/sf5000/HOST_EC96/shell_for_auto_test/host]# lspci | grep b203
3b:02.0 Network controller: Cavium, Inc. Device b203
3b:02.1 Network controller: Cavium, Inc. Device b203
3b:02.2 Network controller: Cavium, Inc. Device b203
3b:02.3 Network controller: Cavium, Inc. Device b203
```

Step5: Check that the loading is complete on the Helium

```
# lspci -nn -d 177d:a0f7
```

```
root@OCTEONTX:/data/code/sf5000/HOST_EC96/dpdk-19.11# lspci -nn -d 177d:a0f7
0002:0f:00.1 System peripheral [0880]: Cavium, Inc. Device [177d:a0f7]
0002:0f:00.2 System peripheral [0880]: Cavium, Inc. Device [177d:a0f7]
0002:0f:00.3 System peripheral [0880]: Cavium, Inc. Device [177d:a0f7]
0002:0f:00.4 System peripheral [0880]: Cavium, Inc. Device [177d:a0f7]
0002:0f:00.5 System peripheral [0880]: Cavium, Inc. Device [177d:a0f7]
root@OCTEONTX:/data/code/sf5000/HOST_EC96/dpdk-19.11# lspci | grep a0f7
0002:0f:00.1 System peripheral: Cavium, Inc. Device a0f7
0002:0f:00.2 System peripheral: Cavium, Inc. Device a0f7
0002:0f:00.3 System peripheral: Cavium, Inc. Device a0f7
0002:0f:00.4 System peripheral: Cavium, Inc. Device a0f7
0002:0f:00.5 System peripheral: Cavium, Inc. Device a0f7
```

Step6: Mapping between virtual ports on the Host side and the Helium side



There is a one-to-one mapping between the VF port on the Host and the SDP port on the Helium(SDP start from 0002:0f:00.2)

7.2 Close / Change Virtual Port on the Host

To close the virtual port on the Host, you need to uninstall the driver first

To modify the virtual port on the Host, you need to uninstall the driver first and then reload the driver

Run the following commands on the Host to uninstall the driver:

```
# ifconfig mvmgmt0 down
# rmmod mgmt_net
# rmmod octeon_drv
```

> Note:
> Note After rmmod octeon_drv, the Helium DPU SNIC will automatically restarts. Do not load

octeon_drv. ko immediately. Wait until the Helium restarts (about 2 minutes) and then load the octeon_drv. ko

8 Run Test Program with DPDK

Download DPDK package：Helium-DPDK19.11-Vx.yRz.tar.gz , This package will compile and run on Hosts and all Helium cards.

Use DPDK test-pmd to test the traffic between virtual ports of Helium. The test networking is as follows:

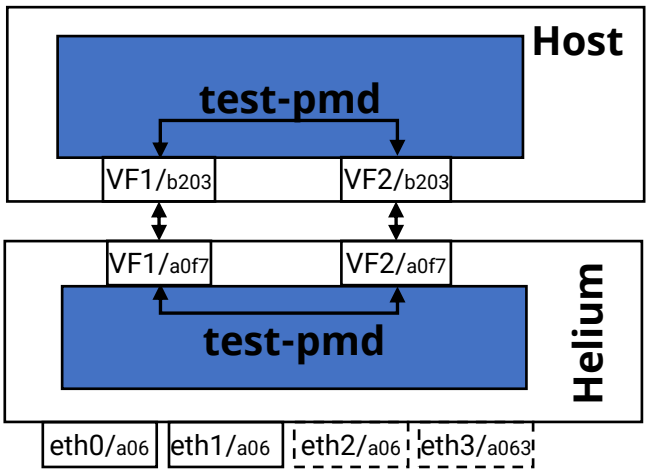


Table 8-1 Network Diagram of test-pmd Simplified Test

Both the Host and the Ethernet DPU run test-pmd, which uses the default transceiver mode, that is, the traffic entering from VF1 comes out of VF2 and the traffic entering from vF2 comes out of VF1. To simply test the traffic, test-pmd of the Ethernet DPU first transmits traffic tx_first for example.

The operations on the Host are as follows:

Step1: Unzip the source package of DPDK and compile it

```
# tar -zxvf Helium-DPDK19.11-V1.0R1.tar.gz
# cd dpdk-19.11
# export RTE_SDK=$PWD
# export RTE_TARGET=build
# make config T=x86_64-native-linuxapp-gcc
# make
```

Note: in case of related dependency libraries, you need to install the dependencies before continuing to compile, as follows:

numa:

yum install numactl-devel(CentOS)

or

apt-get install libnuma-dev(Ubuntu)

Step2: Run the routine

Enable the hugepage and reserve 2G memory for testpmd

mkdir -p /mnt/huge

mount -t hugetlbfs nodev /mnt/huge

sysctl vm.nr_hugepages=2048

Hugepage settings simple instructions:

Set 2048 here, the details are as follows:

1 The memory size of the hugepage on the example Host is 2m, so 1024 pages are required for 2G memory

grep -i huge /proc/meminfo

```
[root@localhost ~/zym/zym/sf5000-c/HOST_EC96/dpdk-19.11]# grep -i huge /proc/meminfo
AnonHugePages:     16384 kB
ShmemHugePages:        0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

2 Host cpus is distributed on 2 numas, so when opening hugepage 1024, each NUMA corresponds to 1G

cat /proc/cpuinfo| grep "physical id"| sort | uniq | wc -l //confirm numa

numactl –hard //numactl view the NUMA corresponding to the specific CPU. Numactl needs to be installed manually, or it can be judged by /proc/cpuinfo

```
[root@localhost ~/zym/zym/sf5000-c/HOST_EC96/dpdk-19.11]# cat /proc/cpuinfo| grep "physical id"| sort | uniq | wc -l
2
[root@localhost ~/zym/zym/sf5000-c/HOST_EC96/dpdk-19.11]# numactl --hard
available: 2 nodes (0-1)
node 0 cpus: 0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46
node 0 size: 63990 MB
node 0 free: 60949 MB
node 1 cpus: 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47
node 1 size: 64465 MB
node 1 free: 64029 MB
node distances:
node   0   1
  0:  10  21
  1:  21  10
```

3 When using testpmd, select the CPU on NUMA 0. In order to use 2G memory, the number of hugepages opened here is 2048(2G Each numa)

4 The memory size of the hugepage above is required by testpmd by default. If the memory of the hugepage is limited, testpmd can be configured manually without considering performance

Hugepage memory parameters used:

./build/app/testpmd -l 0,2,4 -w 0000:3b:02.0 -w 0000:3b:02.1 -- -i --rxd=4096 --total-num-mbufs 8192 --mbuf-size 2048 (Manually configure the memory size required by testpmd)

Load vfio driver

modprobe vfio-pci

#echo 1 > /sys/module/vfio/parameters/enable_unsafe_noiommu_mode

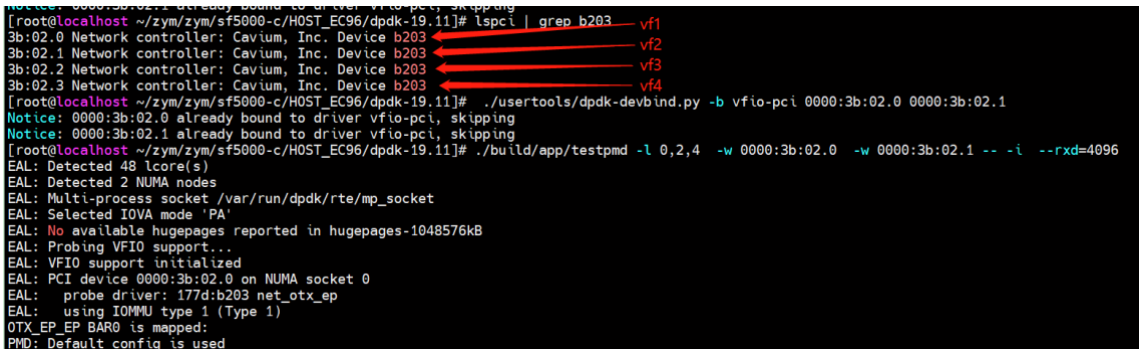
Bind VF0 and VF1 on the Host

lspci | grep b203 // confirm the PCI addresses of the current virtual ports VF1 and VF2

```
# ./usertools/dpdk-devbind.py -b vfio-pci 0000:3b:02.0 0000:3b:02.1
```

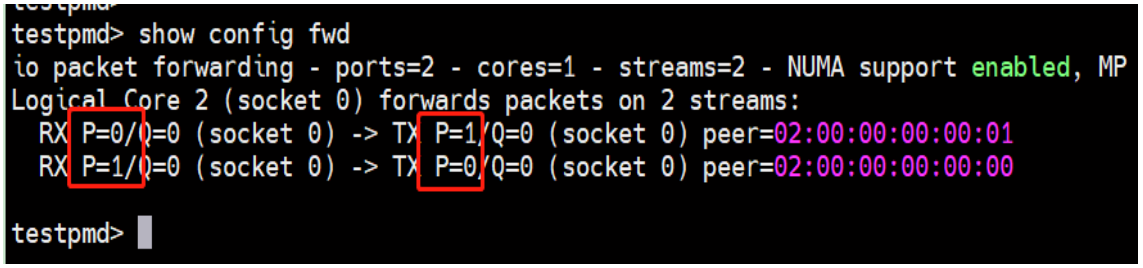
Run testpmd, bind VF1 and VF2, and use VF1, VF2 as forwarding ports

```
# ./build/app/testpmd -l 0,2,4 -w 0000:3b:02.0 -w 0000:3b:02.1 -- -I --rxd=4096
```



Enter testpmd interactive interface and run

```
testpmd> start //start
testpmd> show config fwd // viewing data flow
```



Note: The data flow of the demo program testpmd is that the Host receives the Helium packets from VF0 (0000:3b: 02.0) and then sends it from VF1 (0000:3b: 02.1) to Helium, and vice versa.

The operations on the Helium are as follows:

Step1: Unzip the source package of DPDK and compile it

```
# tar -zxvf Helium-DPDK19.11-V1.0R1.tar.gz
# cd dpdk-19.11
# export RTE_SDK=$PWD
# export RTE_TARGET=build
# make config T=arm64-octeontx2-linux-gcc
```

Run complier

```
# make -j8
```

Step2: Run the routine

Enable the hugepage and reserve 2G memory for testpmd.

```
# sysctl vm.nr_hugepages=12
```

Hugepage settings simple instructions:

Set 12 here, the details are as follows:

1 The memory size of the board's hugepage is 512M, so 2G memory needs 4 pages

```
# grep -i huge /proc/meminfo
```

2 The board CPU is on the same NUMA

```
# cat /proc/cpuinfo| grep "physical id"| sort | uniq | wc -l
```

3 The Helium uses 8 pages currently, so apply for another 4 pages in order to reach a total page size of 12

```
root@OCTEONTX:/data/code/sf5000/HOST_EC96/dpdk-19.11# grep -i huge /proc/meminfo
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
HugePages_Total:       8
HugePages_Free:        4
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:     524288 kB
root@OCTEONTX:/data/code/sf5000/HOST_EC96/dpdk-19.11# sysctl -w vm.nr_hugepages=12
vm.nr_hugepages = 12
root@OCTEONTX:/data/code/sf5000/HOST_EC96/dpdk-19.11# grep -i huge /proc/meminfo
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
HugePages_Total:      12
HugePages_Free:        8
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:     524288 kB
root@OCTEONTX:/data/code/sf5000/HOST_EC96/dpdk-19.11# free
              total        used        free      shared  buff/cache   available
Mem:       33260544     6688000    26278016       54784      294528    23796352
Swap:             0           0           0
root@OCTEONTX:/data/code/sf5000/HOST_EC96/dpdk-19.11# free -h
              total        used        free      shared  buff/cache   available
Mem:            31Gi       6.4Gi        25Gi        53Mi       287Mi        22Gi
Swap:             0B          0B          0B
```

Bind VF0, VF1 and panel port 4, then forward traffic between VF0 and VF1.

```
# lspci | grep a0f7
```
// confirm the PCI addresses of the current virtual ports VF1 and VF2

```
# lspci | grep a063
```
// confirm the PCI address of the current panel port

```
root@OCTEONTX:~# lspci | grep a0f7                                          af
0002:0f:00.1 System peripheral: Cavium, Inc. Device a0f7                   vf1
0002:0f:00.2 System peripheral: Cavium, Inc. Device a0f7                   vf2
0002:0f:00.3 System peripheral: Cavium, Inc. Device a0f7                   vf3
0002:0f:00.4 System peripheral: Cavium, Inc. Device a0f7                   vf4
0002:0f:00.5 System peripheral: Cavium, Inc. Device a0f7
root@OCTEONTX:~# lspci | grep a063
0002:02:00.0 Ethernet controller: Cavium, Inc. Device a063 (rev 09)        面板端口4
0002:03:00.0 Ethernet controller: Cavium, Inc. Device a063 (rev 09)        面板端口3
0002:04:00.0 Ethernet controller: Cavium, Inc. Device a063 (rev 09)        面板端口2
0002:05:00.0 Ethernet controller: Cavium, Inc. Device a063 (rev 09)        面板端口1
root@OCTEONTX:~#
```

```
# ./usertools/dpdk-devbind.py -b vfio-pci 0002:02:00.0 0002:0f:00.2 0002:0f:00.3
```

Run testpmd, bind VF1 and VF2 to panel port 4. Use VF1 and VF2 as forwarding ports.

```
# ./build/app/testpmd -l0,1-2 -w 0002:02:00.0 -w 0002:0f:00.2 -w 0002:0f:00.3 -- -i --portmask=0x6
```
// portmask=0x6 use port1 and port2 as the forwarding port, namely VF1 and vF2

```
root@OCTEONTX:/data/code/sf5000/HOST_EC96/dpdk-19.11# ./build/app/testpmd -l0,1-2 -w 0002:02:00.0 -w 0002:0f:00.2 -w 0002:0f:00.3 -- -i --portmask=0x6
EAL: Detected 24 lcore(s)
EAL: Detected 1 NUMA nodes
EAL: Multi-process socket /var/run/dpdk/rte/mp_socket
EAL: Selected IOVA mode 'VA'
EAL: No available hugepages reported in hugepages-2048kB
EAL: Probing VFIO support...
EAL: VFIO support initialized
EAL: PCI device 0002:02:00.0 on NUMA socket 0
EAL:   probe driver: 177d:a063 net_octeontx2
EAL:   using IOMMU type 1 (Type 1)
EAL: PCI device 0002:0f:00.2 on NUMA socket 0
EAL:   probe driver: 177d:a0f7 net_octeontx2
EAL: PCI device 0002:0f:00.3 on NUMA socket 0
EAL:   probe driver: 177d:a0f7 net_octeontx2
Interactive-mode selected
previous number of forwarding ports 3 - changed to number of configured ports 2
testpmd: create a new mbuf pool <mbuf_pool_socket_0>: n=653824, size=2176, socket=0
```

Enter the testpmd interactive interface, confirm the traffic forwarding on the port, and send the traffic.

```
testpmd> show config fwd
testpmd> start tx_frist
```

```
testpmd> show config fwd
io packet forwarding - ports=2 - cores=1 - streams=2 - NUMA support enabled, MP allocation mode: native
Logical Core 1 (socket 0) forwards packets on 2 streams:
  RX P=1/Q=0 (socket 0) -> TX P=2/Q=0 (socket 0) peer=02:00:00:00:00:02
  RX P=2/Q=0 (socket 0) -> TX P=1/Q=0 (socket 0) peer=02:00:00:00:00:01

testpmd> start tx_first
io packet forwarding - ports=2 - cores=1 - streams=2 - NUMA support enabled, MP allocation mode: native
Logical Core 1 (socket 0) forwards packets on 2 streams:
  RX P=1/Q=0 (socket 0) -> TX P=2/Q=0 (socket 0) peer=02:00:00:00:00:02
  RX P=2/Q=0 (socket 0) -> TX P=1/Q=0 (socket 0) peer=02:00:00:00:00:01

  io packet forwarding packets/burst=32
  nb forwarding cores=1 - nb forwarding ports=2
  port 0: RX queue number: 1 Tx queue number: 1
    Rx offloads=0x0 Tx offloads=0x10000
    RX queue: 0
```

Step3: Check the port counts on the Host or on the Helium

```
testpmd> show port stats all
```

```
testpmd> show port stats all

  ######################## NIC statistics for port 0  ########################
  RX-packets: 0          RX-missed: 0          RX-bytes:  0
  RX-errors: 0
  RX-nombuf:  0
  TX-packets: 0          TX-errors: 0          TX-bytes:  0

  Throughput (since last show)
  Rx-pps:            0          Rx-bps:            0
  Tx-pps:            0          Tx-bps:            0
  ############################################################################

  ######################## NIC statistics for port 1  ########################
  RX-packets: 504848625  RX-missed: 0          RX-bytes:  44426679000
  RX-errors: 0
  RX-nombuf:  0
  TX-packets: 504875560  TX-errors: 0          TX-bytes:  36351040320

  Throughput (since last show)
  Rx-pps:            0          Rx-bps:            0
  Tx-pps:            0          Tx-bps:            0
  ############################################################################

  ######################## NIC statistics for port 2  ########################
  RX-packets: 504875560  RX-missed: 0          RX-bytes:  44429049280
  RX-errors: 0
  RX-nombuf:  0
  TX-packets: 504848688  TX-errors: 0          TX-bytes:  36349105536

  Throughput (since last show)
  Rx-pps:            0          Rx-bps:            0
  Tx-pps:            0          Tx-bps:            0
  ############################################################################
testpmd>
```

9 Install and Test of vhost-user

9.1 Version Confirmation

The feature of vhost-user requires the kernel of virtual machine to support the virtio-net function

virtio-net relies on kernel modules kvm, virtio-blk and virtio-net. The details are as follows:

- Linux kernel >=2.6.25, and support kvm

  # lsmod | grep -i kvm

```
[root@pc2 /opt/kvm]# lsmod | grep -i kvm
kvm_intel             229376  6
kvm                   696320  1 kvm_intel
irqbypass              16384  9 kvm,vfio_pci
[root@pc2 /opt/kvm]#
```

- virtio-net driver suport (virtio-blk virtio-net)

  # grep -i virtio /boot/config-$(uname -r)

```
root@zym:~# grep -i virtio  /boot/config-4.15.18-acso
CONFIG_BLK_MQ_VIRTIO=y
CONFIG_VIRTIO_VSOCKETS=m
CONFIG_VIRTIO_VSOCKETS_COMMON=m
CONFIG_NET_9P_VIRTIO=m
CONFIG_VIRTIO_BLK=m
# CONFIG_VIRTIO_BLK_SCSI is not set
CONFIG_SCSI_VIRTIO=m
CONFIG_VIRTIO_NET=m
CONFIG_CAIF_VIRTIO=m
CONFIG_VIRTIO_CONSOLE=y
CONFIG_HW_RANDOM_VIRTIO=m
CONFIG_DRM_VIRTIO_GPU=m
CONFIG_VIRTIO=y
# Virtio drivers
CONFIG_VIRTIO_PCI=y
CONFIG_VIRTIO_PCI_LEGACY=y
CONFIG_VIRTIO_BALLOON=y
CONFIG_VIRTIO_INPUT=m
CONFIG_VIRTIO_MMIO=y
CONFIG_VIRTIO_MMIO_CMDLINE_DEVICES=y
# CONFIG_RPMSG_VIRTIO is not set
CONFIG_CRYPTO_DEV_VIRTIO=m
```

Both virtio_net kernel module and virtio_blk kernel module are configured and compiled.

- vhost-user needs the QEMU version of the Host to be greater than or equal to 2.7 (**currently only supported QEMU as server**)

  # qemu-img --version

  if libvirt is used, use command 'virsh version' to view the QEMU version used by libvirt

```
[root@localhost ~/zym]# qemu-img --version
qemu-img version 2.12.0
Copyright (c) 2003-2017 Fabrice Bellard and the QEMU Project developers
[root@localhost ~/zym]#
[root@localhost ~/zym]#
[root@localhost ~/zym]#
[root@localhost ~/zym]# virsh version
Compiled against library: libvirt 4.5.0
Using library: libvirt 4.5.0
Using API: QEMU 4.5.0
Running hypervisor: QEMU 2.12.0
```

Note: if QEMU compiles by your self, pay attention to link qemu-kvm, otherwise it will report 'qemu-

```
kvm not found’
   exp:   ln -sf /usr/local/bin/qemu-system-x86_64    /usr/bin/qemu-kvm
          ln -sf /usr/local/bin/qemu-system-x86_64    /usr/libexec/qemu-kvm
          ln -sf /usr/local/bin/qemu-img    /usr/bin/qemu-img
```

9.2 Verify Networking

In order to verify vhost-user, at least two virtual machines are required, and each virtual machine needs at least one vhost-user port. The virio-forwarder service needs to be run on the Host. Bind the vhost-user port to the corresponding VF port on the Host through the tools provided by virtio-forwarder, so that the traffic sent by the virtual machine through the vhost-user port will pass through the Helium DPU SNIC. Run OvS on Helium and bind several VF virtual ports to the same network bridge. Finally, the two virtual machines can communicate with each other through vhost-user ports.

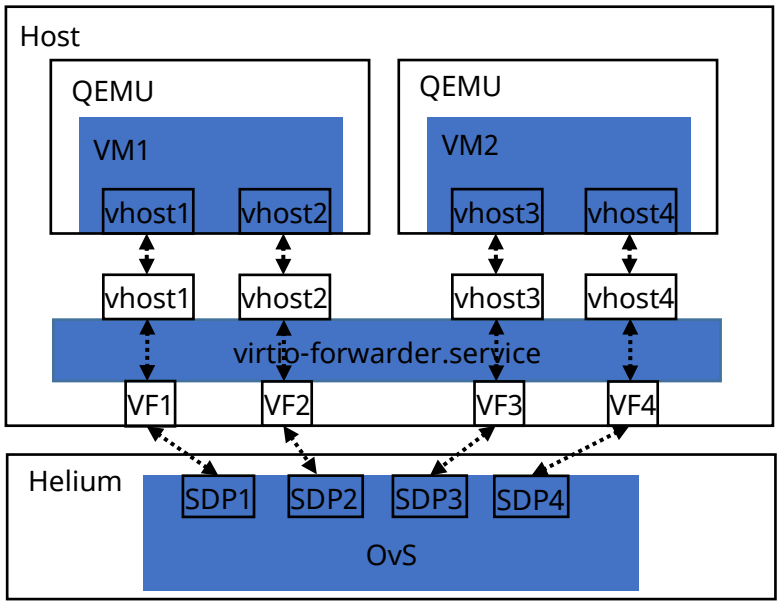


Figure 9-1 Network Diagram of vhost-user Test

9.3 Configuration of Virtual Machine

Vhost-user uses UNIX socket to control message interaction, while the shared memory uses hugepage memory. Therefore, the virtual machine needs to configure UNIX socket and use hugepage memory.

9.3.1 Modification Based on libvirt

if libvirt is used to turn on the virtual machine, after the virtual machine vm1 is successfully established

```
# virsh shutdown vm1 && virsh edit vm1
  <domain type='kvm'>
    <name>vm1</name>
    <uuid>368fab42-4b91-4ffb-93e9-d082a0357ff3</uuid>
    <memory unit='KiB'>1048576</memory>
    <currentMemory unit='KiB'>1048576</currentMemory>
    <memoryBacking>
      <hugepages>
        <page size='2048' unit='KiB' nodeset='0'/>
      </hugepages>
    </memoryBacking>
    <vcpu placement='static'>4</vcpu>
    <os>
      <type arch='x86_64' machine='pc-i440fx-2.12'>hvm</type>
      <boot dev='hd'/>
    </os>
    <features>
      <acpi/>
      <apic/>
    </features>
    <cpu mode='host-model' check='partial'>
      <model fallback='allow'/>
      <numa>
        <cell id='0' cpus='0-3' memory='1048576' unit='KiB' memAccess='shared'/>
      </numa>
    </cpu>
    <clock offset='utc'>
      <timer name='rtc' tickpolicy='catchup'/>
      <timer name='pit' tickpolicy='delay'/>
      <timer name='hpet' present='no'/>
    </clock>
    <on_poweroff>destroy</on_poweroff>
    <on_reboot>restart</on_reboot>
    <on_crash>destroy</on_crash>
    <pm>
      <suspend-to-mem enabled='no'/>
      <suspend-to-disk enabled='no'/>
    </pm>
    <devices>
      <emulator>/usr/libexec/qemu-kvm</emulator>
      <disk type='file' device='disk'>
        <driver name='qemu' type='qcow2'/>
        <source file='/share/vm1.qcow2'/>
        <target dev='vda' bus='virtio'/>
        <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>
      </disk>
      <disk type='file' device='cdrom'>
        <driver name='qemu' type='raw'/>
        <target dev='hda' bus='ide'/>
        <readonly/>
        <address type='drive' controller='0' bus='0' target='0' unit='0'/>
</disk>
      .....
       <interface type='network'>
        <mac address='52:54:00:61:fd:52'/>
        <source network='default'/>
        <model type='virtio'/>
        <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
      </interface>
      <interface type='vhostuser'>
```

modifications 1

modifications 2

```
      <source type='unix' path='/tmp/vhost1.sock' mode='server'/>
      <mtu size='9000'/>
      <model type='virtio'/>
    </interface>
    <interface type='vhostuser'>
      <source type='unix' path='/tmp/vhost2.sock' mode='server'/>
      <model type='virtio'/>
    </interface>

    <serial type='pty'>
      <target type='isa-serial' port='0'>
        <model name='isa-serial'/>
      </target>
</serial>
.....
```

Optional Configuration

modifications 3

Modification 1: add memoryBacking and enable hugepage memory. page_size uses the actual hugepage size of the device through the command ‘cat /proc/meminfo’. Take care that the free hugepage memory size of the device needs to meet the use size of the virtual machine.

Modification 2: configure the virtual machine CPU to use hugepage memory

Modification 3: configure vhost-user interface and socket_file /tmp/vhost1.sock. The path should be the same as the configuration in 9.4.3 below which is -s /tmp/vhost1.sock. If there is more than one port, configure multiple <interface type ='vhostuser'> and MTU size =’9000’ (optional configuration), which means that the MTU value of the vhost port corresponding to the configured virtual machine is 9000.

9.3.2 Modification Based on QEMU

if you use QEMU to turn on vm1, the sample script is as follows:

```
# qemu-system-x86_64
KVM_PATH="qemu-system-x86_64"
# Number of guest cpus
VCPUS_NR="4"
# Memory
MEM=2048
VIRTIO_OPTIONS="csum=off,gso=off,guest_tso4=off,guest_tso6=off,guest_ecn=off"
# Socket Path
SOCKET_PATH1="/tmp/vhost1.sock"
SOCKET_PATH2="/tmp/vhost2.sock"
# nfs mem Path
MEM_PATH="/opt/kvm/vm1.img"

$KVM_PATH   -enable-kvm   -m $MEM   -smp $VCPUS_NR   -cpu host   -name vm1 \
 -vnc :8 \
 -object memory-backend-file,id=ram-node0,mem-path=/dev/hugepages,share=yes,size=${MEM}M \
 -numa node,nodeid=0,cpus=0-3,memdev=ram-node0 \
 -hda $MEM_PATH \
 -chardev socket,id=chr0,path=$SOCKET_PATH1,server \
 -netdev type=vhost-user,id=net0,chardev=chr0,vhostforce \
 -device virtio-net-pci,netdev=net0,host_mtu=9000,mac=CC:CB:BB:BB:BB:BA,$VIRTIO_OPTIONS \
 -chardev socket,id=chr1,path=$SOCKET_PATH2,server \
 -netdev type=vhost-user,id=net1,chardev=chr1,vhostforce \
 -device virtio-net-pci,netdev=net1,host_mtu=9000,mac=CC:CB:BB:BB:BB:BB,$VIRTIO_OPTIONS \
 -netdev tap,id=netshare \
 -device virtio-net-pci,netdev=netshare,mac=52:54:00:2a:de:c5 \
 -monitor telnet::3333,server,nowait
```

modifications 1

modifications 2

Optional Configuration

Modification 1: configure to use hugepage memory.

Modification 2: configure vhost-user port and socket_file path = /tmp/vhost1.sock. The path should be the same as the configuration in 9.4.3 above which is -s /tmp/vhost1.sock. And host_mtu = 9000 is an optional configuration, indicating that the MTU value of the port is 9000.

9.4 virtio-forwarder Configuration

virtio-forwarder is mainly divided into three parts: virtio-forwarder sercive, virtio-forwarder static configuration and virtio-forwarder dynamic configuration.

virtio-forwarder service: virtio-forwarder.service basic service, which manages the configuration of vm vhost port and board VF port, traffic forwarding, etc.

virtio-forwarder static configuration: /etc/default/virtioforwarder, static configuration file, read by default after virtio-forwarder.service startup

virtio-forwarder dynamic configuration: manage ports through dynamic configuration tools

- virtioforwarder_port_control：add /delete VF and vhost-user ports dynamically
- virtioforwarder_stats： display port status

9.4.1 virtio-forwarder Dependency Installation

Step1: Enable hugepage memory on the Host

- Confirm hugepage memory
  `# grep -i huge /proc/meminfo` or `# sysctl -a | grep -i huge`
- Enable hugepage
  `# sysctl -w vm.nr_hugepages=3072`

```
[root@pc3 ~/EC2004Y]# grep -i huge /proc/meminfo
AnonHugePages:     14336 kB
ShmemHugePages:        0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
[root@pc3 ~/EC2004Y]# sysctl -w vm.nr_hugepages=3072
vm.nr_hugepages = 3072
[root@pc3 ~/EC2004Y]# grep -i huge /proc/meminfo
AnonHugePages:     14336 kB
ShmemHugePages:        0 kB
HugePages_Total:    3072
HugePages_Free:     3072
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

Step2: Load the Host driver and start VF

Refer to 7 Configuration of Virtual Port on the Host for further details.

Verify that the installation was successful:

```
[root@pc3 ~/EC2004Y]# lspci -nn -d 177d:b203
02:00.0 Network controller [0280]: Cavium, Inc. Device [177d:b203]
02:00.1 Network controller [0280]: Cavium, Inc. Device [177d:b203]
02:00.2 Network controller [0280]: Cavium, Inc. Device [177d:b203]
02:00.3 Network controller [0280]: Cavium, Inc. Device [177d:b203]
[root@pc3 ~/EC2004Y]#
```

Step3: Load the vfio-pci driver module

```
# modprobe vfio-pci
```

Step4: Install dynamic library: libprotobuf-c.so.1, libzmq.so.5

- install protobuf/protobuf-c:

```
# yum search protobuf
# yum install protobuf.x86_64 -y
# yum install protobuf-c.x86_64 -y
```

- install czmq:

```
# yum install epel-release
# yum search czmq
# yum install czmq.x86_64 -y
```

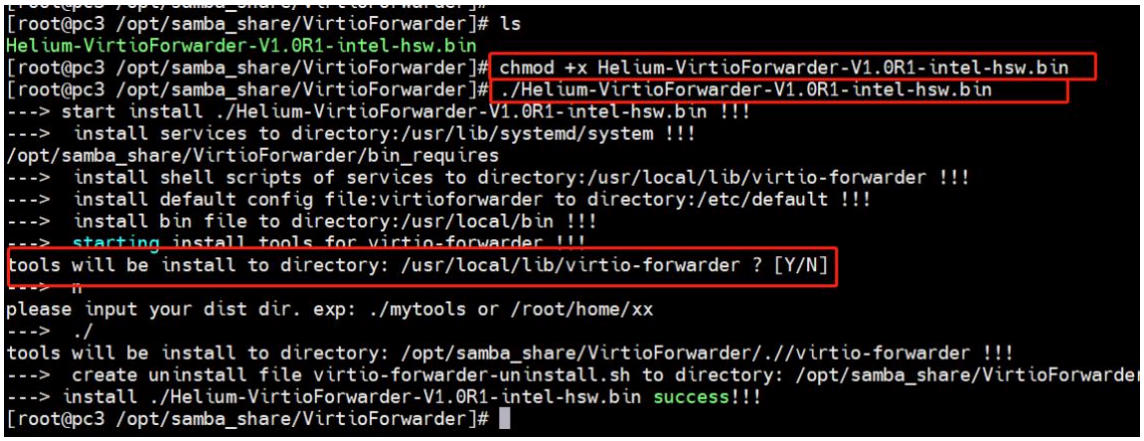
9.4.2 virtio-forwarder Package Installation

Step1: Unzip the installation package

chmod +x Helium-VirtioForwarder-Vx.yRz-intel-hsw.bin

Step2: Run the installation script

./ Helium-VirtioForwarder-Vx.yRz-intel-hsw.bin



- During the installation process, the virtio-forwarder dynamic configuration tool will appear selection of installation directory. The directory supports relative/absolute path. If there is no operation for 30s or choose Y, the path /usr/local/lib/virtio-forwarder will be used as the final dynamic configuration tool path.
- As shown in the figure above, the installation process will display all the paths of virtio-forwarder related files.
- After installation, virtio-forwarder-uninstall.sh will be generated in the directory selected in Note.1 for uninstallation.

9.4.3 Run virtio-forwarder

Step1: Start virtio-forwarder service

systemctl start virtio-forwarder.service //start service

systemctl status virtio-forwarder.service //show service status

systemctl enable virtio-forwarder.service // set the service to start at startup

When the service starts, the configuration in the static configuration file will be read first, and then the service will be started.

The static configuration file /etc/default/virtioforwarder, at present, **is only recommended** to modify the configuration as follows:

VIRTIOFWD_CPU_MASK : cpu core configuration

VIRTIOFWD_LOG_LEVEL : log level

VIRTIOFWD_TSO : tso is on or not (priority is lower than dynamic configuration)

VIRTIOFWD_HUGETLBFS_MOUNT_POINT : hugepage memory path

VIRTIOFWD_VFIO_MTU_DEFAULT : VF default mtu value (priority is lower than dynamic configuration)

VIRTIOFWD_DPDK_SOCKMEM_NUMA : hugepage memory size required for a single NUMA.

Configuration example: 64 VFs are enabled, and the configured CPU cores are evenly distributed on 2 numas.

1 At present, the hugepage required by a single VF < 100m

2 It is allocated on two numas

So VIRTIOFWD_DPDK_SOCKMEM_NUMA = 64*100M/2 = 3200M < 4096M, 4096 is recommended here.

LD_LIBRARY_PATH : the path used to configure the dynamic library required by virtio-forwarder

```
[root@pc3 ~/EC2004Y]# systemctl start virtio-forwarder.service
[root@pc3 ~/EC2004Y]# systemctl status virtio-forwarder.service
● virtio-forwarder.service - Virtio-forwarder
   Loaded: loaded (/usr/lib/systemd/system/virtio-forwarder.service; disabled; vendor preset: disabled)
   Active: active (running) since Fri 2021-09-03 11:49:32 CST; 5s ago
  Process: 10589 ExecStopPost=/usr/local/lib/virtio-forwarder/vio4wd-post-stop.sh (code=exited, status=0/
  Process: 11385 ExecStartPre=/usr/local/lib/virtio-forwarder/vio4wd-pre-start.sh (code=exited, status=0/
 Main PID: 11393 (vio4wd-start.sh)
    Tasks: 15
   CGroup: /system.slice/virtio-forwarder.service
           ├─11393 /bin/bash /usr/local/lib/virtio-forwarder/vio4wd-start.sh
           └─11395 /usr/local/bin/virtio-forwarder -C1,2 --nodaemon=log_syslog -l6 -p/var/run -H/dev/huge

Sep 03 11:49:32 pc3 systemd[1]: Starting Virtio-forwarder...
Sep 03 11:49:32 pc3 systemd[1]: Started Virtio-forwarder.
Sep 03 11:49:32 pc3 virtio-forwarder[11395]: Starting virtio-forwarder 1.1.6-218-g55a3594
Sep 03 11:49:32 pc3 virtio-forwarder[11395]: DPDK version: DPDK 19.11.0
Sep 03 11:49:32 pc3 vio4wd-start.sh[11393]: EAL: No available hugepages reported in hugepages-1048576kB
```

Note:

if the dynamic library cannot be found due to startup error, please confirm whether the dynamic library is successful installed and confirm the installation path.

You can configure the static configuration file /etc/default/virtioforwarder:

LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/:/your_lib_dir

You can also configure the required dynamic library link path or configure the dynamic library environment variables.

Step2: Use the configuration tool to dynamically configure vhost and VF ports (the tool directory is specified during installation, please refer to 9.4.2)

- add vhost and VF port pairs

```
# ./virtioforwarder_port_control add_sock --vhost-path="/tmp/vhost1.sock" --pci-addr="02:00.0"  --tso=on  --mtu=64500
# ./virtioforwarder_port_control add_sock --vhost-path="/tmp/vhost2.sock" --pci-addr="02:00.1"  --tso=on  --mtu=64500
```

`# ./virtioforwarder_port_control -h` you can view specific help. The meanings of relevant parameters are as follows:

--add_sock : add new vhost-user and VF

--vhost-path : correspond to vhost-user sock path of vm

--pci-addr : correspond to the PCI address (0000:02:00.0 or 02:00.0) of vm

--tso : enalbe the tso offload function of the virtual machine and VF. The priority is higher than the static configuration. If the static configuration does not enable tso and tso=on, the vhost + VF will still enable tso

--mtu : set the MTU value of VF. The priority is over static configuration

Set the mtu of VF port here. If it is not set, try to obtain the mtu of the corresponding virtual machine (the corresponding mtu needs to be configured in the virtual machine configuration). If the

acquisition fails, use the default mtu value VIRTIOFWD_VFIO_MTU_DEFAULT in the static configuration

After adding the configuration, the configuration will be saved to /etc/default/virtio-forwarder-dynamic, and loaded automatically after the service is restarted.

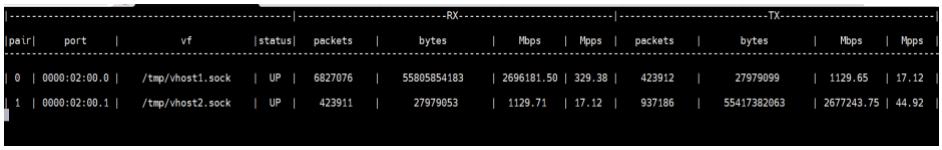
- Delete vhost and VF port pairs:

./virtioforwarder_port_control remove_sock --vhost-path="/tmp/vhost1.sock" --pci-addr="02:00.0"

After deleting the configuration, it will be deleted from /etc/default/virtio-forwarder-dynamic.

Step3: Get the count and status of port

./virtioforwarder_stats



./virtioforwarder_stats -h view specific help

Parameters:

-p : vhost+VF pairing id, that is, the pair item in the figure above.

-d : dynamic refresh, default -d 1. Cancel dynamic refresh -d 0

-o : display mode of count. At present, there are three modes: simple, detail and protobuf. Simple is used by default.

-c : clear count. If set -p, the count of the corresponding vhost + VF will be cleared. If it is not set, all port counts will be cleared

9.5 Configuration of OvS on Helium DPU SNIC

9.5.1 Install and Verify OvS

Step1: Install OvS

chmod +x Helium-OvS-Vx.yRz.bin

./ Helium-OvS-Vx.yRz.bin

Step2: Start OvS

cd ovs_install

chmod +x ovs_start.sh

./ovs_start.sh

or start ovs-start service (**recommended**)

systemctl start ovs-start.service

Step3: Verify the corresponding versions of OvS and DPDK

#ovs-vsctl get Open_vSwitch . dpdk_initialized

true

#ovs-vswitchd --version

ovs-vswitchd (Open vSwitch) 2.11.0

DPDK 19.11.0

#ovs-vsctl get Open_vSwitch . dpdk_version

"DPDK 19.11.0"

Step4: Stop OvS

chmod +x ovs_stop.sh

./ovs_stop.sh

or

#/usr/local/share/openvswitch/scripts/ovs-ctl stop

if OvS start via serivce :

#systemctl stop ovs-start.service

9.5.2 Configure OvS

Step1: OvS configuration dependencies

Enable hugepage memory

sysctl vm.nr_hugepages=32

Bind VF port and panel port

./usertools/dpdk-devbind.py -b vfio-pci 0002:02:00.0 0002:0f:00.2 0002:0f:00.3 0002:0f:00.4 0002:0f:00.5

Step2: Start OvS

systemctl start ovs-start.service

Default configuration env file path for ovs-start.service:

/usr/local/etc/openvswitch/systemd/ovs-systemd-env

The configuration is as follows:

OVS_SCR_PATH=/usr/local/share/openvswitch/scripts //ovs-ctl default path

OVS_DB_SOCK=/usr/local/var/run/openvswitch/db.sock //OvS db connect way

OVS_DB_FILE=/usr/local/etc/openvswitch/conf.db //OvS default db file path

OVS_DPDK_MEM=4096 // OvS uses the large page memory size (M) by default. After the modification, restart the service to take effect.

Note: Hugepage memory configuration here

1 When OvS is started, there is a memory configuration in the OvS db, that is, other_config:dpdk-socket-mem, then this configuration does not take effect, use the configuration in the db as the default large page memory, and enter condition 3

2 When OvS is started, OvS db does not have a huge page memory configuration, then use this configuration as the default huge page memory, and enter condition 3

3 If the current large page remaining memory is greater than the default configuration, use the current default configuration as the configuration to start, otherwise use the remaining available large page memory as the configuration to start

OvS can also be started via the script ovs_start.sh:

ovs_start.sh is located in the ovs_install directory. The env file above the script mode no longer works. It can be manually configured or modified and added to the ovs_start.sh script.

```
# cd ovs_install
# chmod +x ovs_start.sh
# ./ovs_start.sh
```

Step3: Bind the VF port corresponding to the Helium DPU SNIC to OvS
Examples are as follows:

Configure bridge br0

```
# ovs-vsctl add-br br0 -- set bridge br0 datapath_type=netdev
```

Add VF1 to br0 and name it sdp1, and turn on tso

```
# ovs-vsctl add-port br0 sdp1 -- set Interface sdp1 type=dpdk options:dpdk-devargs=0002:0f:00.2 mtu_request=65400
# ovs-vsctl set Interface sdp1 options:tx_offloads=0x1803e
```

Add VF2 to br0 and name it sdp2, and enable tso

```
# ovs-vsctl add-port br0 sdp2 -- set Interface sdp2 type=dpdk options:dpdk-devargs=0002:0f:00.3 mtu_request=65400
# ovs-vsctl set Interface sdp2 options:tx_offloads=0x1803e
```

Add VF3 to br0 and name it sdp3, and enable tso

```
# ovs-vsctl add-port br0 sdp3 -- set Interface sdp3 type=dpdk options:dpdk-devargs=0002:0f:00.4 mtu_request=65400
# ovs-vsctl set Interface sdp3 options:tx_offloads=0x1803e
```

Add VF4 to br0 and name it sdp4, and enable tso

```
# ovs-vsctl add-port br0 sdp4 -- set Interface sdp4 type=dpdk options:dpdk-devargs=0002:0f:00.5 mtu_request=65400
# ovs-vsctl set Interface sdp4 options:tx_offloads=0x1803e
```

Add the panel port eth0 to br0 and name it eth0, and enable tso

```
# ovs-vsctl add-port br0 eth0 -- set Interface eth0 type=dpdk options:dpdk-devargs=0002:02:00.0 mtu_request=9200
# ovs-vsctl set Interface eth0 options:tx_offloads=0x1803e
```

Set the maximum rate of the port (optional)

`# ovs-vsctl set interface sdp1 options:rx_max_rate="10000"` Set the maximum export rate of sdp1, in Mbps

9.6 Power on the Virtual Machine and Test

Start the virtual machines vm1 and vm2, configure the corresponding vhost-user port up, and use iperf3 for traffic testing.

`# virsh start vm1`

`# ifconfig -a` // You can see that there are 2 ports ens7 and ens8, and their macs are the same as the macs in vhost-user in virsh dumpxml vm1. You can also view the successful creation of these 2 ports through lspci.

An example of the operation of vm1 is as follows:

```
root@vm1:~# ifconfig -a
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.122.231  netmask 255.255.255.0  broadcast 192.168.122.255
        inet6 fe80::5054:ff:fe61:fd52  prefixlen 64  scopeid 0x20<link>
        ether 52:54:00:61:fd:52  txqueuelen 1000  (Ethernet)
        RX packets 280  bytes 29183 (29.1 KB)
        RX errors 0  dropped 116  overruns 0  frame 0
        TX packets 126  bytes 16663 (16.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens7: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether 52:54:00:e7:a2:6d  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens8: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether 52:54:00:5e:ca:64  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 92  bytes 7100 (7.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 92  bytes 7100 (7.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@vm1:~# ifconfig ens7 192.168.5.6
root@vm1:~#
```

`# virsh dumpxml vm1`

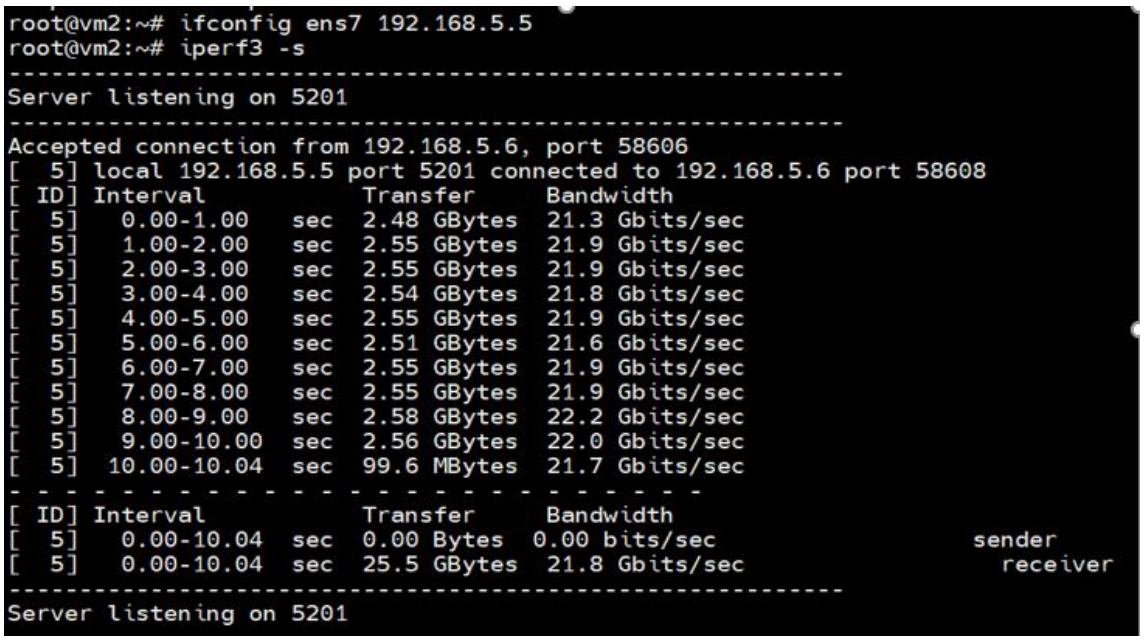
```
    </controller>
    <interface type='network'>
      <mac address='52:54:00:61:fd:52'/>
      <source network='default'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
    </interface>
    <interface type='vhostuser'>
      <mac address='52:54:00:e7:a2:6d'/>
      <source type='unix' path='/tmp/vhost1.sock' mode='server'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>
    </interface>
    <interface type='vhostuser'>
      <mac address='52:54:00:5e:ca:64'/>
      <source type='unix' path='/tmp/vhost2.sock' mode='server'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x08' function='0x0'/>
    </interface>
```

```
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 92  bytes 7100 (7.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@vm1:~# ifconfig ens7 192.168.5.6
root@vm1:~# lspci
00:00.0 Host bridge: Intel Corporation 440FX - 82441FX PMC [Natoma] (rev 02)
00:01.0 ISA bridge: Intel Corporation 82371SB PIIX3 ISA [Natoma/Triton II]
00:01.1 IDE interface: Intel Corporation 82371SB PIIX3 IDE [Natoma/Triton II]
00:01.3 Bridge: Intel Corporation 82371AB/EB/MB PIIX4 ACPI (rev 03)
00:02.0 VGA compatible controller: Cirrus Logic GD 5446
00:03.0 Ethernet controller: Red Hat, Inc. Virtio network device
00:04.0 USB controller: Intel Corporation 82801I (ICH9 Family) USB UHCI Controller #1 (rev 03)
00:04.1 USB controller: Intel Corporation 82801I (ICH9 Family) USB UHCI Controller #2 (rev 03)
00:04.2 USB controller: Intel Corporation 82801I (ICH9 Family) USB UHCI Controller #3 (rev 03)
00:04.7 USB controller: Intel Corporation 82801I (ICH9 Family) USB2 EHCI Controller #1 (rev 03)
00:05.0 SCSI storage controller: Red Hat, Inc. Virtio block device
00:06.0 Unclassified device [00ff]: Red Hat, Inc. Virtio memory balloon
00:07.0 Ethernet controller: Red Hat, Inc. Virtio network device
00:08.0 Ethernet controller: Red Hat, Inc. Virtio network device
root@vm1:~#
```

Configure ens7 up on vm1, configure ens7 up on vm2, use vm1 as the iperf3 client, and vm2 as the iperf3 server:

```
root@vm1:~# ifconfig ens7 192.168.5.6
root@vm1:~# iperf3 -c 192.168.5.5
Connecting to host 192.168.5.5, port 5201
[  4] local 192.168.5.6 port 58608 connected to 192.168.5.5 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  2.59 GBytes  22.3 Gbits/sec    0   3.10 MBytes
[  4]   1.00-2.00   sec  2.55 GBytes  21.9 Gbits/sec    0   3.10 MBytes
[  4]   2.00-3.00   sec  2.55 GBytes  21.9 Gbits/sec    0   3.10 MBytes
[  4]   3.00-4.00   sec  2.53 GBytes  21.8 Gbits/sec    0   3.10 MBytes
[  4]   4.00-5.00   sec  2.55 GBytes  21.9 Gbits/sec    0   3.10 MBytes
[  4]   5.00-6.00   sec  2.51 GBytes  21.6 Gbits/sec    0   3.10 MBytes
[  4]   6.00-7.00   sec  2.55 GBytes  21.9 Gbits/sec    0   3.10 MBytes
[  4]   7.00-8.00   sec  2.55 GBytes  21.9 Gbits/sec    0   3.10 MBytes
[  4]   8.00-9.00   sec  2.58 GBytes  22.1 Gbits/sec    0   3.10 MBytes
[  4]   9.00-10.00  sec  2.55 GBytes  21.9 Gbits/sec    0   3.10 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  25.5 GBytes  21.9 Gbits/sec    0             sender
[  4]   0.00-10.00  sec  25.5 GBytes  21.9 Gbits/sec                  receiver

iperf Done.
root@vm1:~#
```

An example of the operation of vm2 is as follows:



10 Live Migration Test of vhost-user

10.1 Migration Constraints

Source and destination Hosts:

- Physical configuration should be as same as possible
- The same hypervisor version and supports live migration. Currently using qemu-kvm/libvirt
- The virtual machine disk image can be accessed through shared network storage. Currently using NFS
- Linux kernel supports virtio-net, hypervisor supports vhost-user server
- Network connection
- The destination Host does not have the same virtual machine as the source Host (migration conflict failed)

10.2 Live Migration Networking Diagram

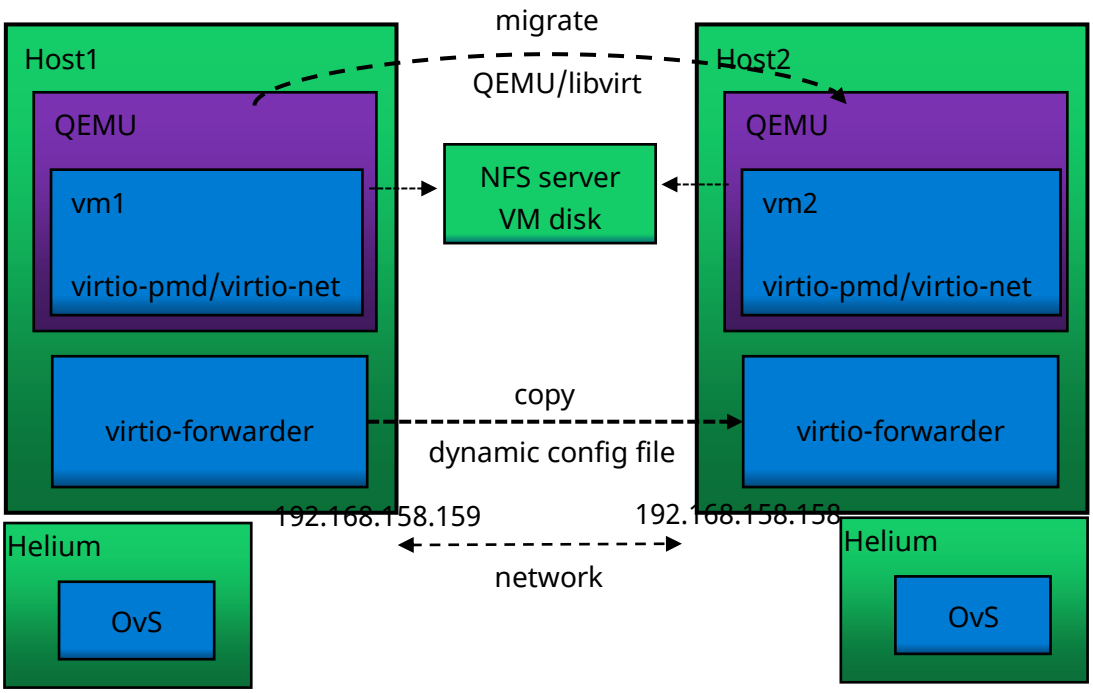


Figure 10-1 Networking Diagram of vhost-user Live Migration Test

According to the above constraints, the live migration test network is shown in above. The virtual machine disk images of the source Host1 and the target Host2 are shared through NFS, and the Host1 ip: 192.168.158.159 and Host2 ip: 192.168.158.158 are connected to the network. The hypervisor uses QEMU + libvirt.

Versions currently tested:

```
[root@localhost ~]# virsh version
Compiled against library: libvirt 4.5.0
Using library: libvirt 4.5.0
Using API: QEMU 4.5.0
Running hypervisor: QEMU 4.2.0

[root@localhost ~]# qemu-img --version
qemu-img version 4.2.0
Copyright (c) 2003-2019 Fabrice Bellard and the QEM
[root@localhost ~]#
```

10.3 NFS Setup

NFS is divided into a client and a server. Currently, Host1 is used as the NFS server and Host2 is used as the client.

Step1: Host1 configures the nfs server and shares the /opt/nfs directory (pay attention to the directory permissions)

```
# yum install nfs-utils rpcbind -y
# vim /etc/exports:
  /opt/nfs *(rw,sync,no_subtree_check,no_root_squash)
# systemctl start rpcbind.service nfs-server.service
# systemctl start nfs-lock nfs-idmap
```

Set to start the nfs server. (optional)

```
# systemctl enable rpcbind.service nfs-server.service
```

Step2: Host2 configures the nfs client and mounts the '/opt/nfs' directory

```
# yum -y install nfs-utils
# showmount -e 192.168.158.159
```

```
[root@pc3 ~]# showmount -e 192.168.158.159
Export list for 192.168.158.159:
/opt/nfs *
[root@pc3 ~]#
```

```
# mkdir /opt/nfs
# mount -t nfs 192.168.158.159:/opt/nfs   /opt/nfs
# chmod -R 777 /opt/nfs
```

```
[root@pc3 ~]# mount -t nfs 192.168.158.159:/opt/nfs  /opt/nfs
[root@pc3 ~]# ls /opt/nfs
copy_vm.sh  create_vm.sh  vv1.qcow2  vv2.qcow2
[root@pc3 ~]# ls /opt/nfs -alh
total 37G
drwxrwxrwx  2 root root   78 Sep 13 15:32 .
drwxrwxrwx. 5 root root   53 May 21 09:18 ..
-rwxr-xr-x  1 root root  393 Sep 13 15:32 copy_vm.sh
-rwxr-xr-x  1 root root  577 Sep 13 15:30 create_vm.sh
-rwxrwxrwx  1 root root  28G Sep 13 19:56 vv1.qcow2
-rwxr-xr-x  1 root root 4.9G Sep 13 19:45 vv2.qcow2
[root@pc3 ~]#
```

Step3: Host2 configures the client to automatically mount the nfs shared directory when it is powered on (optional)

```
# vim /etc/fstab
add  192.168.158.159:/opt/kvm   /opt/kvm   nfs    defaults   0 0
# mount -a
```

10.4 Live Migration based on Libvirt

10.4.1 Migrating Configuration in Libvirt Tcp Mode

Step1: Configure libvirt tcp listening mode

vim /etc/libvirt/libvirtd.conf, modified as follows

```
listen_tls = 0
listen_tcp = 1
tcp_port = "16509"
listen_addr = "0.0.0.0"
auth_tcp = "none"
```

```
[root@pc3 /opt/kvm]# cat /etc/libvirt/libvirtd.conf | grep -v -E "^$|#"
listen_tls = 0
listen_tcp = 1
tcp_port = "16509"
listen_addr = "0.0.0.0"
auth_tcp = "none"
[root@pc3 /opt/kvm]#
```

Step2: Libvirt service starts listening mode

vim /etc/sysconfig/libvirtd, modified as follows

LIBVIRTD_ARGS=”--listen”

```
[root@pc3 /opt/kvm]#
[root@pc3 /opt/kvm]# cat /etc/sysconfig/libvirtd | grep -v -E "^$|#"
LIBVIRTD_ARGS="--listen"
[root@pc3 /opt/kvm]#
```

Step3: Restart the libvirtd service to verify that the configuration is successful

```
# systemctl restart libvirtd.service
# netstat -anpt | grep libvirt
```

```
[root@pc3 ~/PKT_GEN]# netstat -anpt | grep libvirt
tcp        0      0 0.0.0.0:16509           0.0.0.0:*               LISTEN      1531/libvirtd
[root@pc3 ~/PKT_GEN]#
```

```
# virsh -c qemu+tcp://192.168.158.159/system list
```

```
[root@pc3 ~]# virsh list
 Id    Name                           State
----------------------------------------------------

[root@pc3 ~]# virsh list
 Id    Name                           State
----------------------------------------------------
 4     vv1                            running

[root@pc3 ~]# virsh -c qemu+tcp://192.168.158.159/system list
 Id    Name                           State
----------------------------------------------------
 4     vv2                            running

[root@pc3 ~]#
```

The above steps need to be configured on both Host1 and Host2.

10.4.2 Host1 Configuration to Start Running Virtual Machine

Step1: Enable hugepage memory

`# sysctl vm.nr_hugepages=2048`

Step2: Host1 loads the pf driver

- View the host-sid on the Helium DPU SNIC side
- Helium DPU SNIC loads and updates kernel modules such as pcie-ep/mgmt_net, and sets the obtained host-sid
- The Host side loads the pf driver and specifies the number of VF ports (1~64)
  See 7 Configuration of Virtual Port on the Host for details

Step3: Bind VF port to vfio-pci

`# modprobe vfio-pci`

`# dpdk-devbind.py -b vfio-pci 0000:02:00.0 0000:02:00.1 0000:02:00.2`

Step4: Run virtio-forwarder and use the configuration tool to dynamically add vhost and VF ports

- start virtio-forwarder service
  see 9.4 virtio-forwarder Configuration for details
- Use the virtio-forwarder dynamic tool to add VF and vhost ports

  `# /usr/local/lib/virtioforwarder/virtioforwarder_port_control add_sock --vhost-path="/tmp/vhost1.sock" --pci-addr="02:00.0" --tso=on --mtu=64500`

  `# /usr/local/lib/virtioforwarder/virtioforwarder_port_control add_sock --vhost-path="/tmp/vhost2.sock" --pci-addr="02:00.1" --tso=on --mtu=64500`

  `# /usr/local/lib/virtioforwarder/virtioforwarder_port_control add_sock --vhost-path="/tmp/vhost3.sock" --pci-addr="02:00.2" --tso=on --mtu=64500`
- Verify that the configuration is complete

  `# /usr/local/lib/virtioforwarder/virtioforwarder_stats -d 0`

  or view the dynamic configuration file `# cat /etc/default/virtio-forwarder-dynamic`

```
[root@pc3 ~]# cat /etc/default/virtio-forwarder-dynamic

/tmp/vhost2.sock 0000:02:00.1 1 9000 0
/tmp/vhost1.sock 0000:02:00.0 1 9000 0
/tmp/vhost3.sock 0000:02:00.2 1 9000 0
[root@pc3 ~]#
```

```
|-------------------------------------------------|---------------
|pair|     port     |          vf          |status|   packets    |
-------------------------------------------------------------------
| 0  | 0000:02:00.1 |   /tmp/vhost2.sock   | DOWN |      0       |
| 1  | 0000:02:00.0 |   /tmp/vhost1.sock   | DOWN |      0       |
| 2  | 0000:02:00.2 |   /tmp/vhost3.sock   | DOWN |      0       |
```

Step5: Run OvS on the Helium

see 9.5 Configuration of OvS on Helium DPU SNIC for details

Step6: Configure the virtual machine vv1 and run

- Configure the vhost-user port of the virtual machine vv1 /tmp/vhost1.scok

  To ensure the reliability of live migration, close the cache of the virtual machine disk image of the vm

  Add the following configuration:

  <driver name='qemu' type='qcow2' **cache='none'**/>

- Run the virtual machine, and use the pkt-gen of the kernel in the virtual machine to continuously stream to the vhost-user port

  Configure vhost-user port ip

  # ifconfig ens6 192.168.33.33

```
root@vv1:~# ifconfig ens6 192.168.33.33
root@vv1:~# ifconfig
ens2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.122.109  netmask 255.255.255.0  broadcast 192.168.122.255
        inet6 fe80::5054:ff:fee1:f17f  prefixlen 64  scopeid 0x20<link>
        ether 52:54:00:e1:f1:7f  txqueuelen 1000  (Ethernet)
        RX packets 166  bytes 9456 (9.4 KB)
        RX errors 0  dropped 107  overruns 0  frame 0
        TX packets 19  bytes 1656 (1.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens6: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.33.33  netmask 255.255.255.0  broadcast 192.168.33.255
        inet6 fe80::5054:ff:fe7f:479a  prefixlen 64  scopeid 0x20<link>
        ether 52:54:00:7f:47:9a  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1  bytes 90 (90.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 612  bytes 43908 (43.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 612  bytes 43908 (43.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@vv1:~#
```

Use the kernel's pkt-gen to continuously stream to the vhost-user port

./pkt_gen.sh ens6

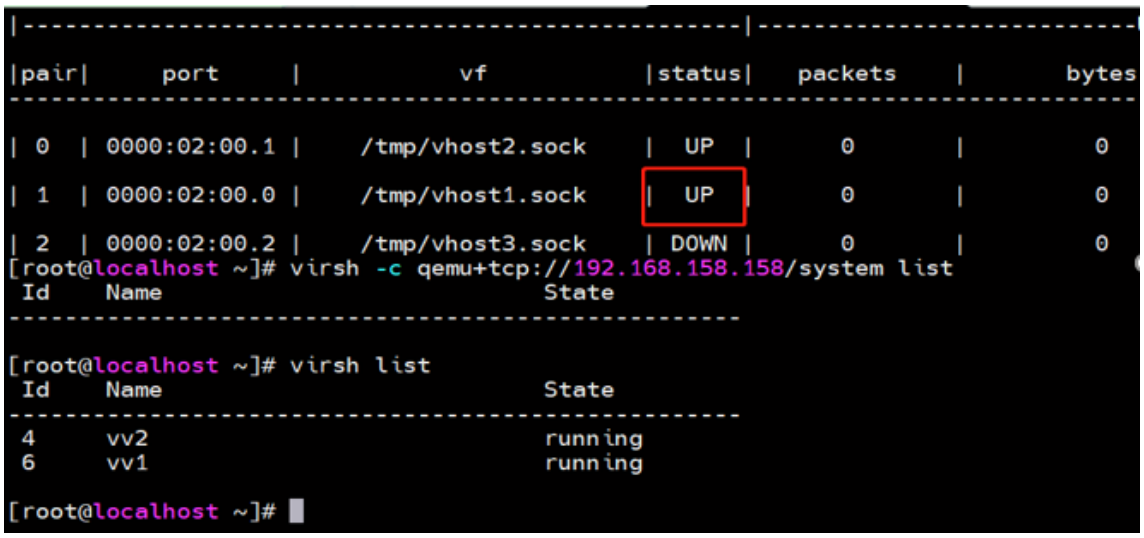
An example of pkt_gen.sh is as follows:

```
 7 #!/bin/bash
 8
 9 if [ $# -lt 1 ]; then
10         echo "input with no args use default ens6"
11         ENS=ens6
12 else
13         echo "input with args $1"
14         ENS=$1
15 fi
17
18 modprobe pktgen
19 echo "rem_device_all" > /proc/net/pktgen/kpktgend_0
20 echo "add_device $ENS" > /proc/net/pktgen/kpktgend_0
21 echo "pkt_size 512" > /proc/net/pktgen/$ENS
22 echo "count 0" > /proc/net/pktgen/$ENS
23 echo "delay 0" > /proc/net/pktgen/$ENS
24 echo "dst_mac 52:54:00:1d:40:00" > /proc/net/pktgen/$ENS
25 echo "flag MACDST_RND" > /proc/net/pktgen/$ENS
26 echo "flag IPDST_RND "     > /proc/net/pktgen/$ENS
27 echo "start" > /proc/net/pktgen/pgctrl
```

```
ens2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.122.109  netmask 255.255.255.0  broadcast 192.168.122.255
        inet6 fe80::5054:ff:fee1:f17f  prefixlen 64  scopeid 0x20<link>
        ether 52:54:00:e1:f1:7f  txqueuelen 1000  (Ethernet)
        RX packets 48  bytes 3304 (3.3 KB)
        RX errors 0  dropped 31  overruns 0  frame 0
        TX packets 9  bytes 1242 (1.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens6: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.33.33  netmask 255.255.255.0  broadcast 192.168.33.255
        inet6 fe80::5054:ff:fe7f:479a  prefixlen 64  scopeid 0x20<link>
        ether 52:54:00:7f:47:9a  txqueuelen 1000  (Ethernet)
        RX packets 25  bytes 3100 (3.1 KB)
        RX errors 0  dropped 25  overruns 0  frame 0
        TX packets 4  bytes 356 (356.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 208  bytes 15096 (15.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 208  bytes 15096 (15.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@vv1:~#
root@vv1:~#
root@vv1:~# ls
00.pcap  pkt-gen3.sh  pkt-gen.sh  socket  start.sh
root@vv1:~# ./pkt-gen.sh ens6
input with args ens6
```

View vhost1 port status UP through the virtio-forwarder tool on Host1

```
# /usr/local/lib/virtioforwarder/virtioforwarder_stats -d 0
```



10.4.3 Host2 Live Migration Preparation

Step1: Enable hugepage memory

Step2: Host2 loads the pf driver

Step3: Bind VF port to vfio-pci

Step 1-3 Same as 10.4.2 Step 1-3

Step4: Run virtio-forwarder

- The dynamic configuration file on Host1 side is copied to Host2

  # cd /etc/default/

  # scp root@192.168.158.159:/etc/default/virtio-forwarder-dynamic ./
- start virtio-forwarder
- Verify that the virtio-forwarder service starts successfully, and the dynamic configuration is loaded, and the port is DOWN

  # /usr/local/lib/virtioforwarder/virtioforwarder_stats -d 0

  # virsh -c qemu+tcp://192.168.158.159/system list

Step5: The Helium side of Host1 runs OvS

10.4.4 Migrating Virtual Machines from Host1 to Host2

Step1: Host1 uses libvirt command to migrate vv1 to Host2

```
# virsh migrate vv1 --live qemu+tcp://192.168.158.158/system --unsafe tcp://192.168.158.158
```

```
[root@localhost ~]#
[root@localhost ~]#
[root@localhost ~]# virsh migrate vv1 --live qemu+tcp://192.168.158.158/system --unsafe tcp://192.168.158.158

[root@localhost ~]# virsh list
 Id    Name                           State
----------------------------------------------------
 4     vv2                            running

[root@localhost ~]# virsh -c qemu+tcp://192.168.158.158/system list
 Id    Name                           State
----------------------------------------------------
 4     vv1                            running

[root@localhost ~]#
```

Step2: The migration is completed, the vv1 port corresponding to the virtio-forwarder on the Host1 side is down, and the virtual machine vv1 is down

```
# /usr/local/lib/virtioforwarder/virtioforwarder_stats -d 0
# virsh list
```

```
|-----------------------------------------------|-----------------------UP-----------------------|-----------------------DOWN-----------------------
|pair|    port     |          vf          |status|  packets  |    bytes    |  Mbps  |  Mpps  |  packets  |    bytes    |  Mbps  |  Mpps
-------------------------------------------------------------------------------------------------------------------------------------------
| 0  | 0000:02:00.1 |   /tmp/vhost2.sock   |  UP  |     0     |      0      |  0.00  |  0.00  |     0     |      0      |  0.00  |  0.00
| 1  | 0000:02:00.0 |   /tmp/vhost1.sock   | DOWN |    26     |     3224    |  0.00  |  0.00  |     0     |      0      |  0.00  |  0.00
| 2  | 0000:02:00.2 |   /tmp/vhost3.sock   | DOWN |     0     |      0      |  0.00  |  0.00  |     0     |      0      |  0.00  |  0.00
[root@localhost ~/sf5000/HOST_EC96/shell_for_auto_test/host]#
```

```
[root@pc3 ~]# virsh list
 Id    Name                           State
----------------------------------------------------

[root@pc3 ~]# virsh list
 Id    Name                           State
----------------------------------------------------
 4     vv1                            running

[root@pc3 ~]# virsh -c qemu+tcp://192.168.158.159/system list
 Id    Name                           State
----------------------------------------------------
 4     vv2                            running

[root@pc3 ~]#
```

Step3: The vv1 port corresponding to the virtio-forwarder on the Host2 side is up, and the virtual machine vv1 is up

The virtual machine vv1 runs on Host2 and runs pkt-gen without interruption. Check the port status on Host2. The vhost-user port corresponding to vv1 is up and has traffic.

```
# /usr/local/lib/virtioforwarder/virtioforwarder_stats -d 0
```

```
|-----------------------------------------------|-----------------------UP-----------------------|-----------------------DOWN-----------------------
|pair|    port     |          vf          |status|  packets  |    bytes    |  Mbps  |  Mpps  |  packets  |    bytes    |  Mbps  |  Mpps
-------------------------------------------------------------------------------------------------------------------------------------------
| 0  | 0000:02:00.1 |   /tmp/vhost2.sock   | DOWN |     0     |      0      |  0.00  |  0.00  |     0     |      0      |  0.00  |  0.00
| 1  | 0000:02:00.0 |   /tmp/vhost1.sock   |  UP  |     0     |      0      |  0.00  |  0.00  | 117484950 |  60152287368  | 1583201.62 |3092.19
| 2  | 0000:02:00.2 |   /tmp/vhost3.sock   | DOWN |     0     |      0      |  0.00  |  0.00  |     0     |      0      |  0.00  |  0.00
```

```
root@vv1:~# cat /proc/net/pktgen/ens6
Params: count 0  min_pkt_size: 512  max_pkt_size: 512
     frags: 0  delay: 0  clone_skb: 0  ifname: ens6
     flows: 0 flowlen: 0
     queue_map_min: 0  queue_map_max: 0
     dst_min:   dst_max:
     src_min:   src_max:
     src_mac: 52:54:00:7f:47:9a dst_mac: 52:54:00:1d:40:00
     udp_src_min: 9  udp_src_max: 9  udp_dst_min: 9  udp_dst_max: 9
     src_mac_count: 0  dst_mac_count: 0
     Flags: IPDST_RND  MACDST_RND
Current:
     pkts-sofar: 316131064  errors: 0
     started: 53324053us  stopped: 155870098us idle: 1836us
     seq_num: 316131065  cur_dst_mac_offset: 0  cur_src_mac_offset: 0
     cur_saddr: 192.168.33.33  cur_daddr: 0.0.0.0
     cur_udp_dst: 9  cur_udp_src: 9
     cur_queue_map: 0
     flows: 0
Result: OK: 102546045(c102544209+d1836) usec, 316131064 (512byte,0frags)
  3082820pps 12627Mb/sec (12627230720bps) errors: 0
root@vv1:~#
```

Step4: Define and save vv1 xml on the Host2 side

Although vv1 is migrated to Host2, the xml configuration file has not been migrated. Manually save the configuration and try to restart, and the restart is successful.

```
# virsh dumpxml vv1 > vv1.xml
# virsh define vv1.xml
```

```
[root@pc3 ~/code/sf5000-b/HOST_EC96/shell_for_auto_test/host]#
[root@pc3 ~/code/sf5000-b/HOST_EC96/shell_for_auto_test/host]# virsh dumpxml vv1 >vv1.xml
[root@pc3 ~/code/sf5000-b/HOST_EC96/shell_for_auto_test/host]# virsh define vv1.xml
Domain vv1 defined from vv1.xml

[root@pc3 ~/code/sf5000-b/HOST_EC96/shell_for_auto_test/host]#
[root@pc3 ~/code/sf5000-b/HOST_EC96/shell_for_auto_test/host]# virsh list
 Id    Name                           State
----------------------------------------------------
 1     vv1                            running

[root@pc3 ~/code/sf5000-b/HOST_EC96/shell_for_auto_test/host]# virsh shutdown vv1
Domain vv1 is being shutdown

[root@pc3 ~/code/sf5000-b/HOST_EC96/shell_for_auto_test/host]# virsh list
 Id    Name                           State
----------------------------------------------------
 1     vv1                            running

[root@pc3 ~/code/sf5000-b/HOST_EC96/shell_for_auto_test/host]# virsh list
 Id    Name                           State
----------------------------------------------------

[root@pc3 ~/code/sf5000-b/HOST_EC96/shell_for_auto_test/host]# virsh start vv1
Domain vv1 started

[root@pc3 ~/code/sf5000-b/HOST_EC96/shell_for_auto_test/host]# virsh list
 Id    Name                           State
----------------------------------------------------
 2     vv1                            running
```

10.4.5 Host1 Migrates Virtual Machines to Host2 (there is no vhost-user port on Host2)

Step1: Host2 live migration preparation

Host2 creates a new net named migrate, which is used for migration changes of Host1 vhost-user port

- Create a new netw.xml and set a net named migrate
  vim netw.xml :
  <network>
  <name>migrate</name>
  <bridge name='migvirbr0' stp='off' delay='0'/>
  </network>
- define migrate xml
  `# virsh net-define netw.xml`
- start migrate xml
  `# virsh net-start migrate`
- Set auto-start (optional)
  `# virsh net-autostart migrate`

```
[root@pc3 /opt/nfs]# virsh list
 Id    Name                           State
----------------------------------------------------

[root@pc3 /opt/nfs]# virsh edit vv2
error: failed to get domain 'vv2'
error: Domain not found: no domain with matching name 'vv2'

[root@pc3 /opt/nfs]# cat netw.xml
<network>
  <name>migrate</name>
  <bridge name='migvirbr0' stp='off' delay='0'/>
</network>
[root@pc3 /opt/nfs]# virsh net-list
 Name                 State      Autostart     Persistent
----------------------------------------------------------
 default              active     no            yes
 migrate              active     no            yes

[root@pc3 /opt/nfs]#
```

Step2: Host1 live migration preparation

- Host1 dumps the xml of the virtual machine vv2 to be migrated to vv21.xml
  `# virsh dumpxml vv2 > vv21.xml`

```
[root@localhost /opt/nfs]#
[root@localhost /opt/nfs]#
[root@localhost /opt/nfs]# virsh list
 Id    Name                           State
----------------------------------------------------

[root@localhost /opt/nfs]# virsh start vv2
Domain vv2 started

[root@localhost /opt/nfs]# virsh list
 Id    Name                           State
----------------------------------------------------
 13    vv2                            running

[root@localhost /opt/nfs]# virsh dumpxml vv2 >vv21.xml
[root@localhost /opt/nfs]#
```

➢ Modify the vhost-user type port

The vhost-user port type is changed to network, and the source-network is changed to the net:migrate currently set on the migration Host2

```
    <interface type='network'>
      <mac address='52:54:00:ac:b6:6a'/>
      <source network='default' bridge='virbr0'/>
      <target dev='vnet0'/>
      <model type='rtl8139'/>
      <alias name='net0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
    </interface>
    <interface type='vhostuser'>
      <mac address='52:54:00:31:82:88'/>
      <source type='unix' path='/tmp/vhost2.sock' mode='server'/>
      <model type='virtio'/>
      <alias name='net1'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>
    </interface>
    <serial type='pty'>
      <source path='/dev/pts/1'/>
```

```
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x1'/>
    </controller>
    <interface type='network'>
      <mac address='52:54:00:ac:b6:6a'/>
      <source network='default' bridge='virbr0'/>
      <target dev='vnet0'/>
      <model type='rtl8139'/>
      <alias name='net0'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
    </interface>
    <interface type='network'>
      <mac address='52:54:00:31:82:88'/>
      <source network='migrate'/>
      <model type='virtio'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>
    </interface>
    <serial type='pty'>
      <source path='/dev/pts/1'/>
      <target type='isa-serial' port='0'>
```

Step3: Host1 Live Migration

Host1 uses the libvirt migration command migrate to migrate vv2

`virsh migrate vv2 --live qemu+tcp://192.168.158.158/system --unsafe tcp://192.168.158.158 --xml vv2.xml`

```
[root@localhost /opt/nfs]# virsh migrate vv2 --live qemu+tcp://192.168.158.158/system --unsafe tcp://192.168.158.158  --xml vv2.xml

[root@localhost /opt/nfs]# virsh list
 Id    Name                           State
----------------------------------------------------

[root@localhost /opt/nfs]# virsh -c qemu+tcp://192.168.158.158/system list
 Id    Name                           State
----------------------------------------------------
 6     vv2                            running

[root@localhost /opt/nfs]#
```

Step4: Host2 Migration Results

The virtual machine on Host2 is successfully migrated and continues to run. You can check that the vhost-user port has been changed to migrate through dumpxml.

```
# virsh list
# virsh dumpxml vv2 > migratevv2.xml
# vim migratevv2.xml
```

```
[root@pc3 /opt/nfs]# virsh list
 Id    Name                           State
----------------------------------------------------
 6     vv2                            running

[root@pc3 /opt/nfs]# virsh dumpxml vv2  > migratevv2.xml
[root@pc3 /opt/nfs]#
```

```
 83       <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x1'/>
 84     </controller>
 85     <interface type='network'>
 86       <mac address='52:54:00:ac:b6:6a'/>
 87       <source network='default' bridge='virbr0'/>
 88       <target dev='vnet0'/>
 89       <model type='rtl8139'/>
 90       <alias name='net0'/>
 91       <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
 92     </interface>
 93     <interface type='network'>
 94       <mac address='52:54:00:31:82:88'/>
 95       <source network='migrate'/>
 96       <model type='virtio'/>
 97       <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>
 98     </interface>
 99     <serial type='pty'>
100       <source path='/dev/pts/1'/>
101       <target type='isa-serial' port='0'>
```

10.5 QEMU based Live Migration

10.5.1 Host1 Configuration to Start Running Virtual Machine

The steps are basically the same as the steps in 10.4.2 Host1 Configuration to Start Running Virtual Machine. Use QEMU instead of libvirt to start the virtual machine.

Use the QEMU command line to start, you need to pay attention to the following features:

- Configure huge pages (vhost-user uses Host's huge page memory as data communication for vhost-user)
- Open vhost-user port and set unix socket

- Set tap type port (access Host + external network)

Configure /etc/qemu-ifup and /etc/qemu-ifdown to connect/disconnect the port to the Host bridge

```
[root@pc3 ~]# cat /etc/qemu-ifdown
#!/bin/bash

-
[root@pc3 ~]# cat /etc/qemu-ifup
#!/bin/sh
echo "ip link set $1 master virbr0"
ip link set $1 master virbr0

[root@pc3 ~]#
```

- It is recommended to use host-model type for cpu

It is best to use the same cpu types on both ends of the migration. If the difference between the cpu types is large, you can configure cpu config, or use custom cpu

Host1 side QEMU sample script qemu_start.sh:

```
# qemu-system-x86_64
KVM_PATH="/root/qemu/build/qemu-system-x86_64"
# Number of guest cpus
VCPUS_NR="4"
# Memory
MEM=2048

VIRTIO_OPTIONS="csum=off,gso=off,guest_tso4=off,guest_tso6=off,guest_ecn=off"
# Socket Path
SOCKET_PATH1="/tmp/vhost2.sock"
SOCKET_PATH2="/tmp/vhost3.sock"
# nfs mem Path
MEM_PATH="/opt/kvm/vm1.img"

$KVM_PATH \
 -enable-kvm \
 -m $MEM \
 -smp $VCPUS_NR \
 -cpu host \
 -name vv2 \
 -vnc :8 \
 -object memory-backend-file,id=ram-node0,mem-path=/dev/hugepages,share=yes,size=${MEM}M \
 -numa node,nodeid=0,cpus=0-3,memdev=ram-node0 \
 -hda $MEM_PATH \
 -chardev socket,id=chr0,path=$SOCKET_PATH1,server \
 -netdev type=vhost-user,id=net0,chardev=chr0,vhostforce \
 -device virtio-net-pci,netdev=net0,mac=CC:CB:BB:BB:BB:BA,$VIRTIO_OPTIONS \
 -chardev socket,id=chr1,path=$SOCKET_PATH2,server \
 -netdev type=vhost-user,id=net1,chardev=chr1,vhostforce \
 -device virtio-net-pci,netdev=net1,mac=CC:CB:BB:BB:BB:BB,$VIRTIO_OPTIONS \
 -netdev tap,id=netshare \
 -device virtio-net-pci,netdev=netshare,mac=52:54:00:2a:de:c5 \
 -monitor telnet::3333,server,nowait
```

Note: If the startup fails, pay attention to modifying the qemu-ifup permissions

```
[root@localhost /opt/nfs]# ls
copy_vm.sh  create_vm.sh  qemu_start.sh  vv1.qcow2  vv2.qcow2
[root@localhost /opt/nfs]# ./qemu_start.sh
qemu-system-x86_64: -chardev socket,id=chr0,path=/tmp/vhost1.sock,server: info: QEMU waiting for connection on: disconnected:unix:/tmp/vhost1.sock,server
qemu-system-x86_64: -chardev socket,id=chr1,path=/tmp/vhost2.sock,server: info: QEMU waiting for connection on: disconnected:unix:/tmp/vhost2.sock,server
qemu-system-x86_64: network script /etc/qemu-ifup failed with status 256
[root@localhost /opt/nfs]# chmod +x /etc/qemu-ifup
[root@localhost /opt/nfs]# chmod +x /etc/qemu-ifdown
[root@localhost /opt/nfs]# ./qemu_start.sh
qemu-system-x86_64: -chardev socket,id=chr0,path=/tmp/vhost1.sock,server: info: QEMU waiting for connection on: disconnected:unix:/tmp/vhost1.sock,server
qemu-system-x86_64: -chardev socket,id=chr1,path=/tmp/vhost2.sock,server: info: QEMU waiting for connection on: disconnected:unix:/tmp/vhost2.sock,server
ip link set tap0 master virbr0
```

10.5.2 Host2 Live Migration Preparation

Basically the same as step 10.4.3, use QEMU instead of libvirt to start the virtual machine

The QEMU startup of Host2 is basically the same as that of Host1, plus -incoming tcp:0:5555 to enable tcp 5555 port listening.

10.5.3 Migrating Virtual Machines from Host1 to Host2

Step1: Host1 migrates using the QEMU command

Enter QEMU monitor, use telnet mode (QEMU command-line configuration)

telnet localhost 3333 // 3333 is configured for the above QEMU startup

(qemu) migrate tcp:192.168.158.158:5555 // 5555 is configured for Host2 QEMU

A brief introduction to QEMU migrate:

(qemu) migrate -d -b tcp:dest_ip:8888

-d can query the migration status during the migration process, otherwise it can only be queried after the migration.

-b migrate virtual machine storage files

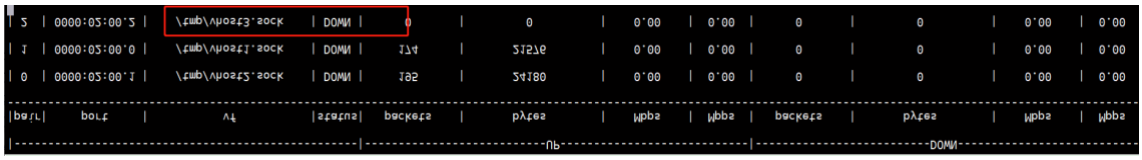
View Migration Status

(qemu) info migrate

```
[root@localhost ~]# telnet localhost 3333
Trying ::1...
Connected to localhost.
Escape character is '^]'.
QEMU 4.2.0 monitor - type 'help' for more information
(qemu) migrate tcp:192.168.158.158:5555
(qemu) info migra
migrate                 migrate_cache_size    migrate_capabilities
migrate_parameters
(qemu) info migrate
globals:
store-global-state: on
only-migratable: off
send-configuration: on
send-section-footer: on
decompress-error-check: on
clear-bitmap-shift: 18
Migration status: completed
total time: 19041 milliseconds
downtime: 166 milliseconds
setup: 8 milliseconds
transferred ram: 767968 kbytes
throughput: 330.67 mbps
remaining ram: 0 kbytes
total ram: 2114888 kbytes
duplicate: 339483 pages
skipped: 0 pages
normal: 190873 pages
normal bytes: 763492 kbytes
dirty sync count: 3
page size: 4 kbytes
multifd bytes: 0 kbytes
pages-per-second: 10230
(qemu)
(qemu)
```

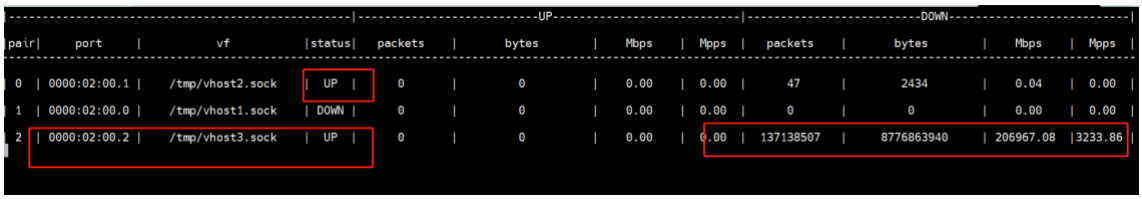
Step2: After Host1 migration is completed, vv2 and virtio-forwarder related port should be down on Host1

```
# /usr/local/lib/virtio-forwarder/virtioforwarder_stats -d 0
```



Step3: vv2 runs on Host2, the virtio-forwarder port is up, and the service continues to run on vv2

```
# /usr/local/lib/virtio-forwarder/virtioforwarder_stats -d 0
```



pair	port	vf	status	UP packets	UP bytes	UP Mbps	UP Mpps	DOWN packets	DOWN bytes	DOWN Mbps	DOWN Mpps
0	0000:02:00.1	/tmp/vhost2.sock	UP	0	0	0.00	0.00	47	2434	0.04	0.00
1	0000:02:00.0	/tmp/vhost1.sock	DOWN	0	0	0.00	0.00	0	0	0.00	0.00
2	0000:02:00.2	/tmp/vhost3.sock	UP	0	0	0.00	0.00	137138507	8776863940	206967.08	3233.86

Appendix A : Special Description of the Helium SNIC Application Scenarios

Special Instructions for Helium DPU SNIC Application Scenario 1:

- In theory, the restart of the Helium DPU SNIC needs to uninstall all related drivers from the Host side

If the corresponding driver and VF have been loaded on the Host side, restart the Helium DPU SNIC directly instead of uninstalling the corresponding driver on the Host side:

1、 At this time, since the driver on the Host side is not uninstalled, the Helium driver cannot be driven on the Host after the Helium restart.
2、 If the host_sid of the Helium DPU SNIC on the Host side is 0x00, the Helium can be restarted successfully.
3、 If the host_sid of the Helium DPU SNIC on the Host side is not 0x00, the Helium DPU SNIC may not start successfully or the Helium will be very stuck after startup.
Recovery method: uninstall mgmt_net.ko on the Host side

```
# ifconfig mvmgmt0 down >/dev/null
# rmmod mgmt_net
```

If it still cannot be recovered, it is recommended to log in to the serial port to check whether the Helium hangs in the system selection interface, press Ctl+D, and continue to run.

4、 If there is a need to restart the Helium network card continuously, and temporarily ignore the driver on the Host side (that is, the Host side is only used for power supply)
Confirm that the current Helium board is on the host_sid on the Host side, enter the boot interface of the Helium network card through the serial port, and enter the following configuration on the boot interface of the Helium network card:

```
> setenv bootargs 'console=pci0 console=ttyAMA0,115200n8 earlycon=pl011,0x87e028000000 maxcpus=24 rootwait rw root=/dev/mmcblk0p3 coherent_pool=16M pcie_ep.host_sid=0x33a00'
> saveenv
```

If you replace the PCIe slot, please pay attention to restore the configuration, or change the host_sid option to the correct value !!!

Special Instructions for Helium DPU SNIC Application Scenario 2:

- The application A runs on the Helium DPU SNIC. When A loaded multiple ports of Helium, the first loaded port does not support unloading from A, while other ports do not have this restriction, that is: the

first loaded port will be bound to the corresponding application A.

Take the OvS application as an example:

1、If the Host side loaded 4 VF ports, the Helium will have 8 available network ports, 4 eth + 4 SDP

2、OvS starts to load these 8 ports by default, and eth0 is the first port to be loaded

3、All these ports can be configured/added via OvS, including eth0

4、All these ports can be removed via OvS and added via OvS reconfiguration

5、You can delete all these ports through OvS and use all other ports other than eth0 (the first to be loaded) for other applications (such as rebinding to kernel drivers or processes), without affecting traffic between other ports in use.

6、It is not possible to delete the eth0 port through OvS for use by other non-OvS processes (that is, the OvS process does not exit, and the first loaded port cannot be used by other applications)

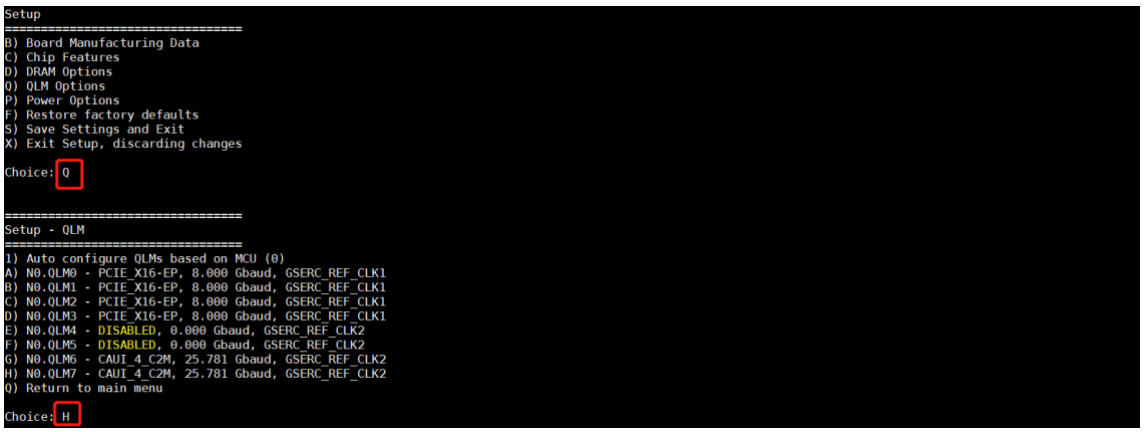
Appendix B : EC2002P 100G port - 1 divided into 4 and restore

EC2002P 100G - 1 port divided into 4 operations:

You need to operate the configuration in the board boot interface. The specific operations are as follows:

Step1: Connect the EC2002P to the serial port, then restart the network card, press B to enter the BDK, and press S to enter the setting mode

Step2: Press Q to enter QLM settings
Press H to configure port mode for panel port 1
If it is to configure the port mode of port 2 of the panel, you can press the G option here



Step3: Press 2 to configure all lanes for this port

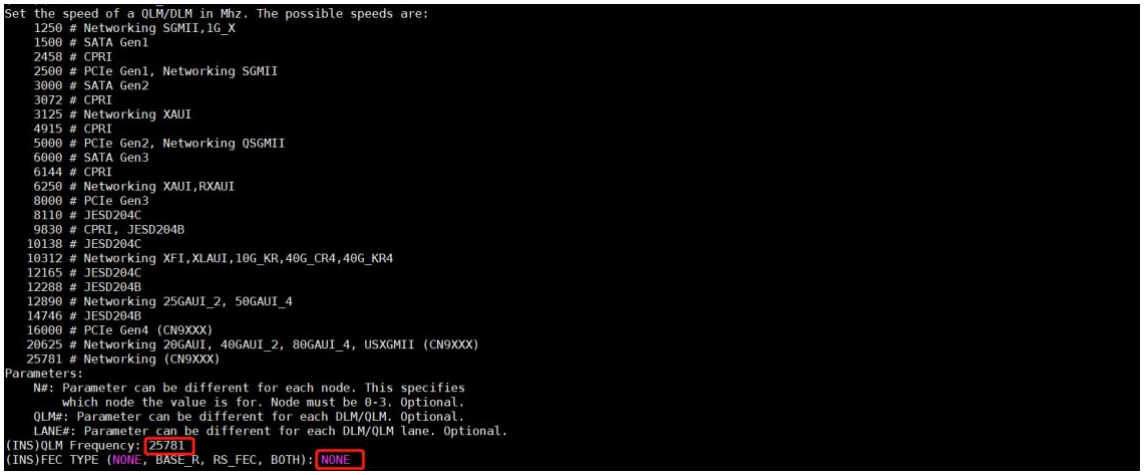
```
==============================
Setup - QLM Lanes and CLK
==============================
1) Configure the QLM_CLK
2) Configure all lanes with the same QLM_MODE and QLM_FREQ
A) N0.QLM7.0 - CAUI_4_C2M, 25.781 Gbaud, GSERC_REF_CLK2
B) N0.QLM7.1 - CAUI_4_C2M, 25.781 Gbaud, GSERC_REF_CLK2
C) N0.QLM7.2 - CAUI_4_C2M, 25.781 Gbaud, GSERC_REF_CLK2
D) N0.QLM7.3 - CAUI_4_C2M, 25.781 Gbaud, GSERC_REF_CLK2
Q) Return to main menu

Choice: 2

Set the mode of a QLM/DLM. The format of the key is:
QLM-MODE.N0.QLM#.LANE#
Parameters:
    N0: Parameter can be different for each node. This specifies
        which node the value is for. Node must be 0.
    QLM#: Parameter can be different for each DLM/QLM. See table
        below for encoding per chip.
    LANE#: Parameter can be different for each DLM/QLM lane. Some
        chips support mixing ethernet protocols on the same DLM/QLM.
        When this is desired, specify the lane. Otherwise don't
        specify the lane.
Possible mode values:
    PCIE_X1, PCIE_X2, PCIE_X4, PCIE_X8, PCIE_X16,
    PCIE_X4-EP, PCIE_X8-EP, PCIE_X16-EP,
    SATA,
    SGMII, 1G_X, QSGMII, XAUI, RXAUI,
    XFI, SFI, XLAUI, XLAUI_C2M,
    10G_KR, 40G_CR4, 40G_KR4,
    20GAUI_C2C,
    25GAUI_C2C, 25GAUI_C2M, 25G_CR, 25G_KR,
    25GAUI_2_C2C,
    40GAUI_2_C2C,
    50GAUI_2_C2C, 50GAUI_2_C2M, 50G_CR2, 50G_KR2,
    50GAUI_4_C2C,
    80GAUI_4_C2C,
    CAUI_4_C2C, CAUI_4_C2M, 100G_CR4, 100G_KR4,
    USXGMII_4X1, USXGMII_2X1, USXGMII_1X1
CN8XXX QLM/DLM Naming:
    CN83XX:
        HRM     Key                 Values
        QLM0    QLM-MODE.N0.QLM0    PCIE_X4,PCIE_X8
```

Step4: Configure the Mode of the port, press Enter to confirm

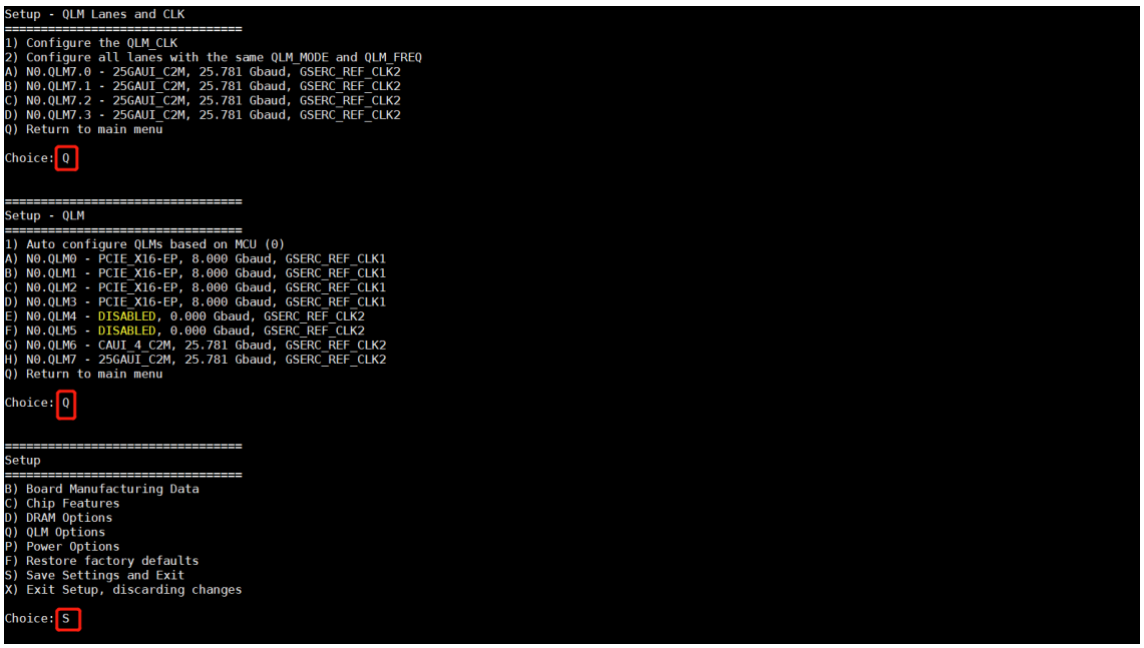
```
        GSERC0  QLM-MODE.N0.QLM2    Ethernet CGX1/CPRI0
        GSERC1  QLM-MODE.N0.QLM3    Ethernet CGX1/CPRI0
        GSERC2  QLM-MODE.N0.QLM4    Ethernet CGX2/CPRI1
        GSERC3  QLM-MODE.N0.QLM5    Ethernet CGX2/CPRI1
        GSERC4  QLM-MODE.N0.QLM6    Ethernet CGX3/CPRI2
F95MM:
        HRM     Key                 Values
        GSERR0  QLM-MODE.N0.QLM0    Ethernet CGX0
        GSERR1  QLM-MODE.N0.QLM1    Ethernet CGX1
        GSERJ0  QLM-MODE.N0.QLM2    JESD204C TOFC0
        GSERJ1  QLM-MODE.N0.QLM3    JESD204C TOFC0
        GSERJ2  QLM-MODE.N0.QLM4    JESD204C TOFC1
        GSERJ3  QLM-MODE.N0.QLM5    JESD204C TOFC1
        GSERJ4  QLM-MODE.N0.QLM6    JESD204C TOFC2
        GSERJ5  QLM-MODE.N0.QLM7    JESD204C TOFC2
        GSERJ6  QLM-MODE.N0.QLM8    JESD204C TOFC3
        GSERJ7  QLM-MODE.N0.QLM9    JESD204C TOFC3
CN98XX:
        HRM     Key                 Values
        GSERP0  QLM-MODE.N0.QLM0    PCIE_X4,PCIE_X8,PCIE_X16
                                    PCIE_X4-EP,PCIE_X8-EP,PCIE_X16-EP
        GSERP1  QLM-MODE.N0.QLM1    PCIE_X4,PCIE_X8,PCIE_X16
                                    PCIE_X4-EP,PCIE_X8-EP,PCIE_X16-EP
        GSERP2  QLM-MODE.N0.QLM2    PCIE_X4,PCIE_X8,PCIE_X16
                                    PCIE_X4-EP,PCIE_X8-EP,PCIE_X16-EP
        GSERP3  QLM-MODE.N0.QLM3    PCIE_X4,PCIE_X8,PCIE_X16
                                    PCIE_X4-EP,PCIE_X8-EP,PCIE_X16-EP
        GSERP4  QLM-MODE.N0.QLM4    PCIE_X4,PCIE_X8,PCIE_X16
                                    PCIE_X4-EP,PCIE_X8-EP,PCIE_X16-EP
        GSERP5  QLM-MODE.N0.QLM5    PCIE_X4,PCIE_X8,PCIE_X16
                                    PCIE_X4-EP,PCIE_X8-EP,PCIE_X16-EP
        GSERP6  QLM-MODE.N0.QLM6    PCIE_X4,PCIE_X8,PCIE_X16
                                    PCIE_X4-EP,PCIE_X8-EP,PCIE_X16-EP
        GSERP7  QLM-MODE.N0.QLM7    PCIE_X4,PCIE_X8,PCIE_X16
                                    PCIE_X4-EP,PCIE_X8-EP,PCIE_X16-EP
        GSERP8  QLM-MODE.N0.QLM8    PCIE_X4
        GSERR0  QLM-MODE.N0.QLM9    Ethernet CGX0
        GSERR1  QLM-MODE.N0.QLM10   Ethernet CGX1
        GSERR2  QLM-MODE.N0.QLM11   Ethernet CGX2
        GSERR3  QLM-MODE.N0.QLM12   Ethernet CGX3
        GSERR4  QLM-MODE.N0.QLM13   Ethernet CGX4
    Notes:
        1) For PCIe spanning multiple GSERP, each QLM must be specified.
(INS)QLM Mode: 25GAUI_C2M
```

Step5: Configure port requency and FEC type

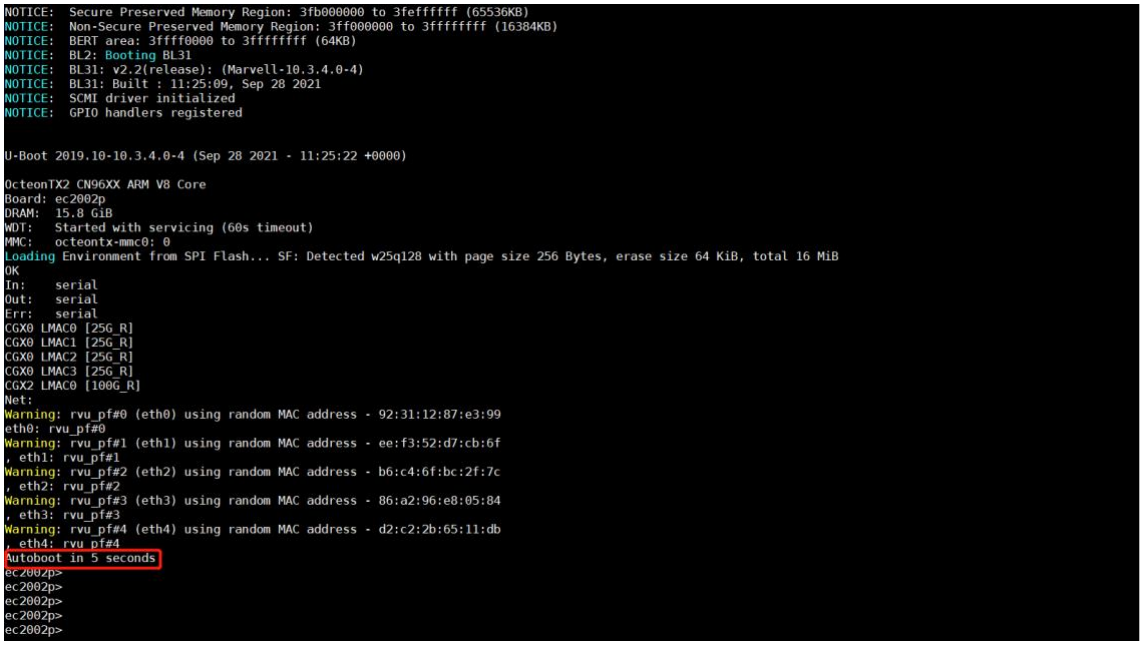
The default FEC type here is NONE, and press Enter to confirm



Step6: Press Q to exit the current panel port 1 setting, then press Q to exit the QLM setting, press S to save, and then it will restart automatically



Step7: After restarting, press s+t on the following interface to enter the uboot command line



Step8: Reset the port mode of eth0 in uboot, which is still 100G by default. It needs to be reset to restart to take effect

When only panel port 1 is set to 4*25G mode, use the following command:

```
set_ignore eth0 1
reset
```

When only panel port 2 is set to 4*25G mode, use the following command:

```
set_ignore eth1 1
reset
```

When both panel port 1 and panel port 2 are set to 4*25G mode, use the following command:

```
set_ignore eth0 1
set_ignore eth4 1
reset
```

Step9: After restarting, enter the Linux system and open the port used

When only panel port 1 is set to 4*25G mode, panel port 1 corresponds to four ports eth0~eth3:

```
i2cset -y 1 0x30 9 3   // enable all panel ports
ifconfig eth0 up
ifconfig eth1 up
ifconfig eth2 up
ifconfig eth3 up
```

When only panel port 2 is set to 4*25G mode, panel port 2 corresponds to four ports eth1~eth4:

```
i2cset -y 1 0x30 9 3   // enable all panel ports
ifconfig eth1 up
ifconfig eth2 up
ifconfig eth3 up
ifconfig eth4 up
```

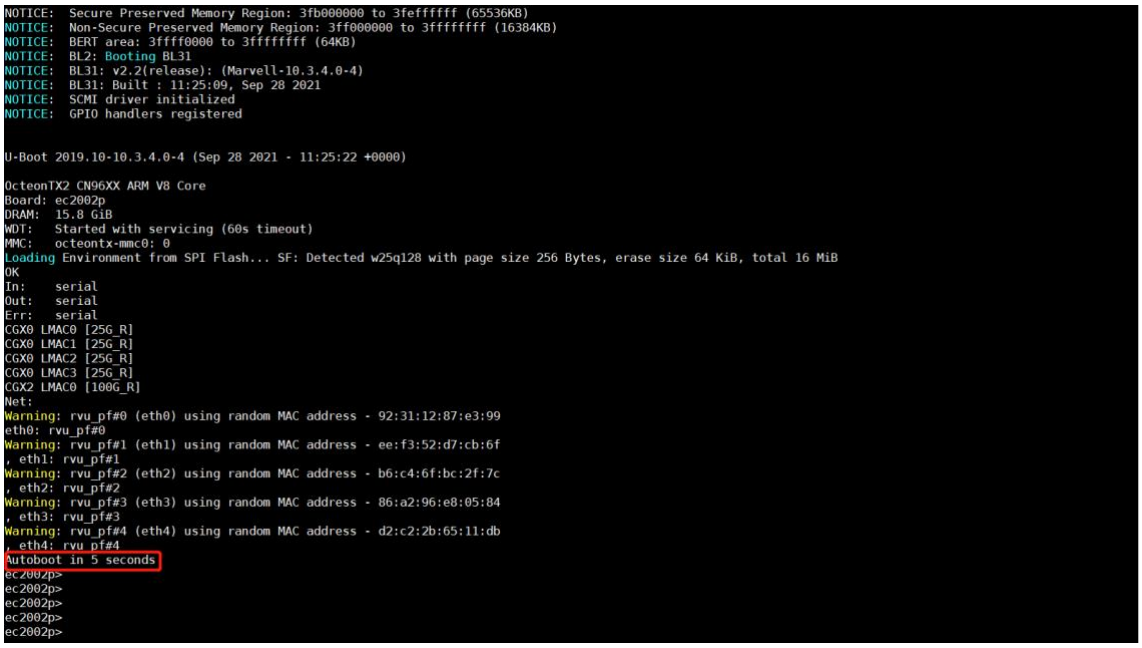
When both panel port 1 and panel port 2 are set to 4*25G mode, panel port 1 corresponds to four ports of eth0~eth3, and panel port 2 corresponds to four ports of eth4~eth7:

```
i2cset -y 1 0x30 9 3    // enable all panel ports
ifconfig eth0 up
ifconfig eth1 up
ifconfig eth2 up
ifconfig eth3 up
ifconfig eth4 up
ifconfig eth5 up
ifconfig eth6 up
ifconfig eth7 up
```

EC2002P 100G – Restore to 100G from 1 port divided into 4:

Verify that the current port configuration is a 1 in 4 state

Step1: After restarting, press s+t on the following interface to enter the uboot command line



Step2: Ignore the port mode of eth0-eth7 in uboot

When only panel port 1 is set to 4*25G mode, panel port 1 corresponds to four ports eth0~eth3:

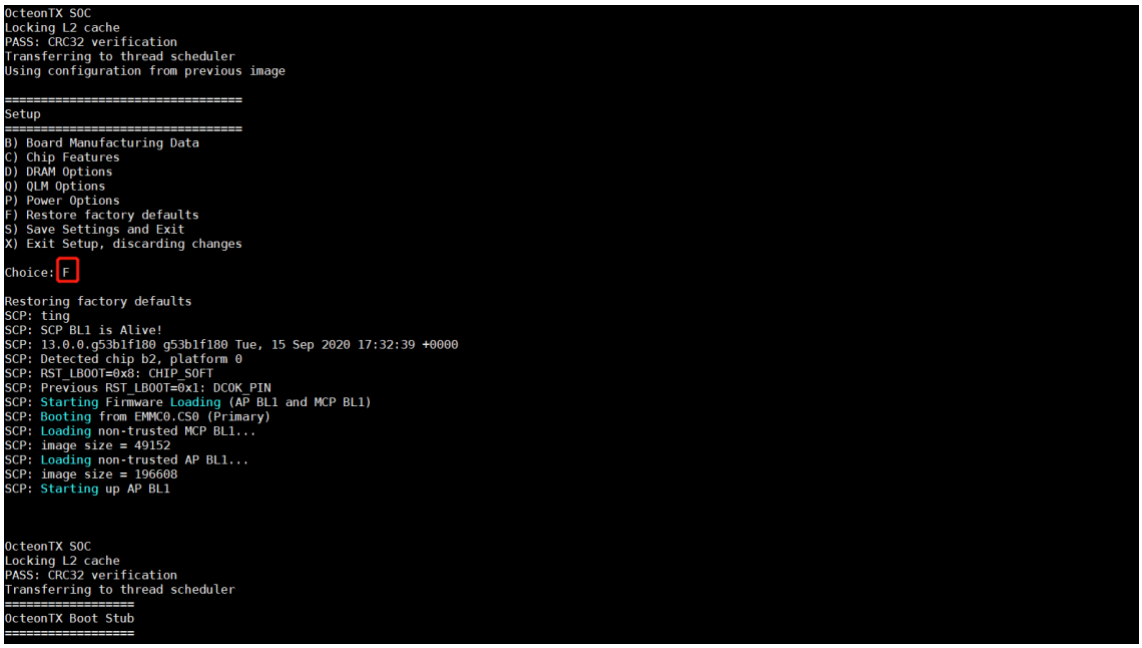
```
set_ignore eth0 1
set_ignore eth1 1
set_ignore eth2 1
set_ignore eth3 1
saveenv
reset
```

When both panel port 1 and panel port 2 are set to 4*25G mode, panel port 1 corresponds to four ports of eth0~eth3, and panel port 2 corresponds to four ports of eth4~eth7:

```
set_ignore eth0 1
set_ignore eth1 1
set_ignore eth2 1
set_ignore eth3 1
set_ignore eth4 1
set_ignore eth5 1
set_ignore eth6 1
set_ignore eth7 1
saveenv
reset
```

Step3: press B to enter the BDK, and press S to enter the setting mode

Step4: Press F to restore factory default



Step5: After restarting, enter the Linux system and open the port used

```
i2cset -y 1 0x30 9 3    //enable all panel ports
```

Appendix C : Helium NIC OS Installation

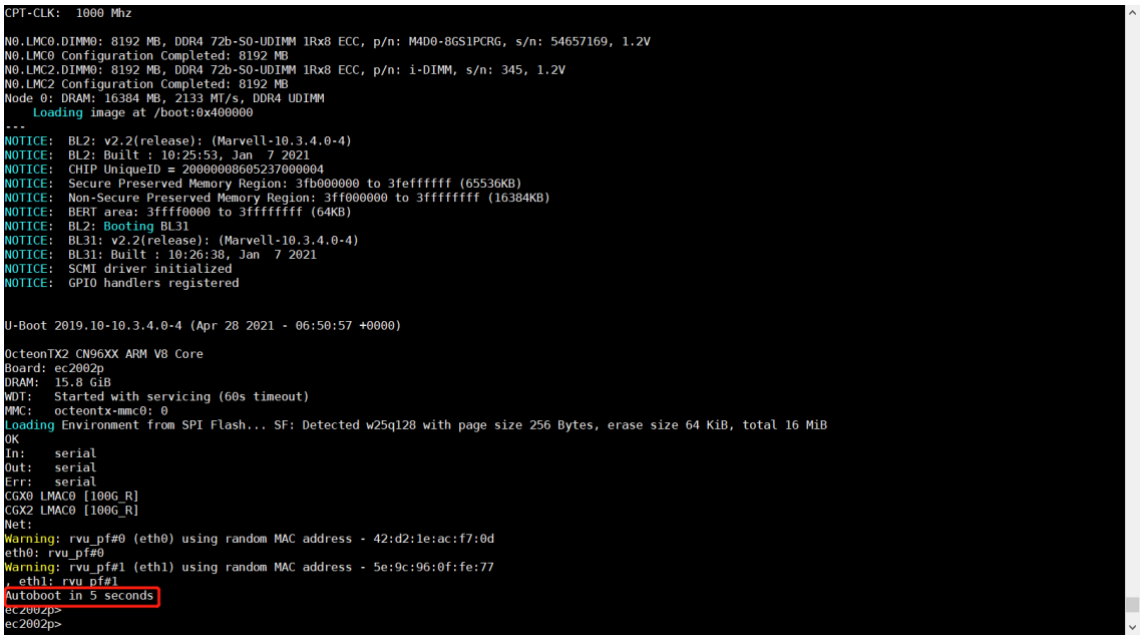
The installation process of Helium DPU SNIC operating system is as follows:

- Enter uboot through serial port
- Download and update the busybox image through tftp under uboot
- Download and install the operating system under busybox

Installation steps

Step1: Helium reboot and enter uboot

After logging in to the Helim through the serial port, enter `#reboot` to restart the Helium. After Autoboot in 5 seconds is displayed, press `s` and `t` to enter uboot



Step2: Verify that the management network port is activated in uboot mode

The current management network port eth2 corresponding to EC2002P, and eth4 corresponding to EC2004Y

Execute `> ethlist` to confirm whether there are eth2/eth4 ports. If not, enter the command `> usb reset` to activate the eth2/eth4 ports and execute `> ethlist` again to confirm.

Note: After each restart of the board, it is necessary to confirm whether the eth2/eth4 port is activated in uboot mode

Example:

Step3: Download and automatically enter busybox

Place the image of busybox Helium-busybox-Vx.yRz.img on the tftp server, configure the management network port ip and the tftp server ip

Set the ethact environment variable to eth2/eth4 (determined according to the model EC2002P/EC2004Y)

Download busybox image via tftp and automatically enter busybox

Example of command configuration:

> usb reset

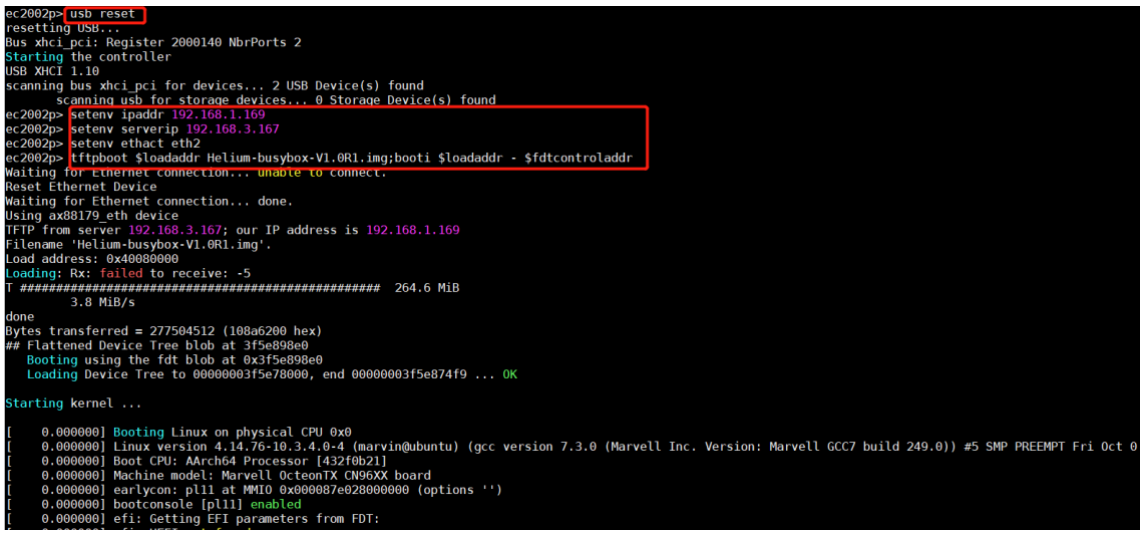
> setenv ipaddr { ip} //Configure the management network port ip to download the image package, which is in the same network segment as the tftp server ip

> setenv serverip {tftp_server_ip } //tftp server ip address

> setenv ethact eth2 // Set the management network port eth2

> tftpboot $loadaddr Helium-busybox-Vx.yRz.img;booti $loadaddr - $fdtcontroladdr // Download busybox image via tftp

Example:



If the tftp transfer fails, please check the problem points below 1 Confirm that the management network port eth2/eth4 is activated and set successfully, and check whether ethact is set to eth2/eth4 (EC2002P eth2 / EC2004Y eth4) 2 The port transmission rate of the Helium smart NIC should be the same as the rate corresponding to the tftp server. Currently, the speed used by the Helium smart NIC is gigabit

Step4: Enter the busybox terminal and download the Helium OS package

Enter the username #root to enter the terminal
Configure the management network port address to download the board system package Helium-Debian10-Vx.yRz.tar (the address of the management network port must be in the same network segment as the ip of the device where the system package is placed)

Example of command configuration:

#ifconfig eth2 <ip>
#route add default gw <gateway_ip> // gateway_ip is the gateway address, no route can be configured without it
#scp root@<service_ip>:~/Helium-Debian10-Vx.yRz.tar ./ // service_ip is the ip of the device that stores the system software package
example：

```
Welcome to Buildroot Marvell
marvell login: root
~# ifconfig eth2 192.168.1.169
[  362.812390] IPv6: ADDRCONF(NETDEV_UP): eth2: link is not ready
~# [  362.886276] ax88179_178a 2-1:1.0 eth2: ax88179 - Link status is: 1

~#
~# [  365.829970] ax88179_178a 2-1:1.0 eth2: ax88179 - Link status is: 1
[  365.841370] IPv6: ADDRCONF(NETDEV_CHANGE): eth2: link becomes ready

~# route add default gw 192.168.1.1
~#
~# scp tjc@192.168.3.167:/var/lib/tftpboot/Helium-Debian10-V1.0R1.tar ./
Ubuntu 16.04.6 LTS
tjc@192.168.3.167's password:
Helium-Debian10-V1.0R1.tar                    100%  401MB  54.7MB/s   00:07
~#
```

Step5: Unzip and install the OS system

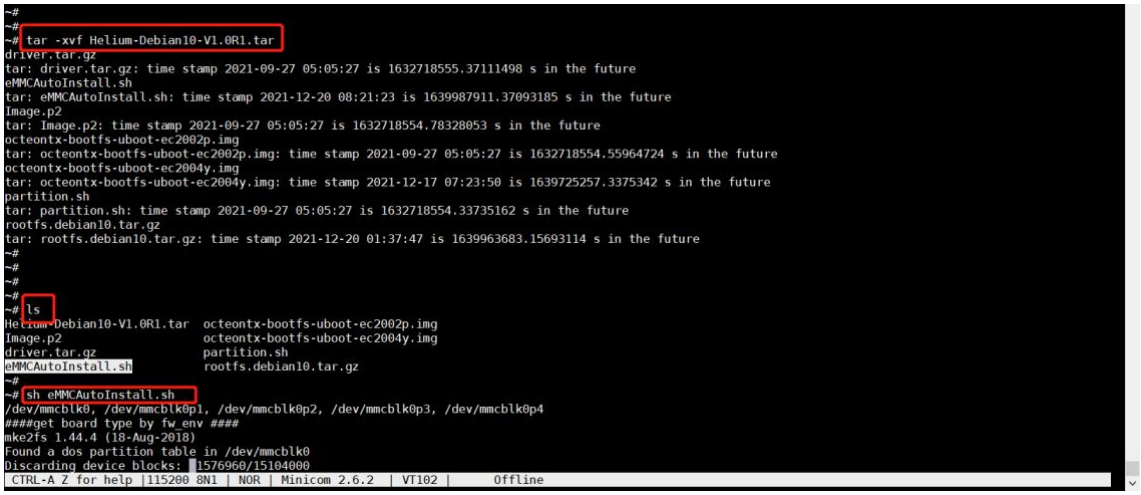
Unzip Helium-Debian10-Vx.yRz.tar and run eMMCAutoInstall.sh in the system package to install the system (the current system time under busybox may be different from the time of the compressed package. In order to prevent too many time alarms during decompression, it is recommended to Set the current system time)

Example of command configuration:

date -s “2023-2-1” // Configure system time
tar -xvf Helium-Debian10-Vx.yRz.tar //unzip

```
# sh eMMCAutoInstall.sh    //install OS
```

example:



Step6: Restart and complete the installation

```
# reboot
```

About Asterfusion

As a solution provider for next-generation cloud network architecture, Asterfusion helps its customers redefine their cloud network infrastructure with a leading generic software-defined-network solution and combining with the collaborations from its ecosystem partners, fulfills its ultimate goal of "Re-Engineering the Cloud Networks". With its patented packet processing and distributed routing technologies, a complete open-source network OS, a generic programmable hardware platform and a fabric controller closely aligned with cloud data center OS, Asterfusion provides completely open and ultra-high performance virtual network solutions to its cloud users along with the capabilities of application visibility, auto provisioning and deployment scheduling. For more information on Asterfusion's products and solutions, please visit www.asterfusion.com or follow us on Wechat public account. For sales inquiries, please send an email to sales@asterfusion.com.