NVIDIA DOCA Emulated Devices

User Guide

MLNX-15-060468 _v1.2 | January 2022

NVIDIA DOCA 模拟设备

用户指南

MLNX-15-060468 _v1.2 | 一月 2022

Table of Contents

目录

Chapter 1. Mediated Devices

NVIDIA mediated devices deliver flexibility in allowing to create accelerated devices without SR-IOV on the BlueField system. These mediated devices support NIC and RDMA and offer the same level of ASAP2 offloads as SR-IOV VFs. Mediates devices are supported using mlx5 sub-function acceleration technology.

Two sub-function devices are created on the BlueField device upon boot (one per port if the port is in switchdev mode) using commands from `/etc/mellanox/mlnx-sf.conf`:

```
/sbin/mlnx-sf -a create -d 0000:03:00.0 -u 61a59715-aeec-42d5-be83-f8f42ba8b049 --
mac 12:11:11:11:11:11
/sbin/mlnx-sf -a create -d 0000:03:00.1 -u 5b198182-1901-4c29-97a0-6623f3d02065 --
mac 12:11:11:11:11:12
```

The help menu for `mlnx-sf` is presented below:

```
Usage: mlnx-sf [ OPTIONS ]
OPTIONS:
-a, -action,     --action <action>              Perform action
    action:      { enable | create | configure | remove | show | set_max_mdevs |
 query_mdevs_num }

-d, -device,     --device <device>              PCI device
 <domain>:<bus>:<device>.<func> (E.g.: 0000:03:00.0)
-m, -max_mdevs, --max_mdevs <max mdevs number>  Set maximum number of MDEVs
-u, -uuid,       --uuid <uuid>                  UUID to create SF with
-M, -mac,        --mac <MAC>                    MAC to create SF with
-p, -permanent, --permanent [<conf file>]       Store configuration to be used after
 reboot and/or driver restart. Default (/etc/mellanox/mlnx-sf.conf).
-V, -version,    --version                      Display script version and exit
-D, -dryrun,     --dryrun                       Display commands only
-v, -verbose,    --verbose                      Run script in verbose mode (print
 out every step of execution)
-h, -help,       --help                         Display help

"/etc/mellanox/mlnx-sf.conf" can be updated manually or using "mlnx-sf" tool with "-
p" parameter.
```

1.1. Related Configuration

Interface names are configured using the UDEV rule under `/etc/udev/rules.d/82-net-setup-link.rules`.

```
SUBSYSTEM=="net", ACTION=="add", ATTR{phys_switch_id}!="", ATTR{phys_port_name}!="",
 \
        IMPORT{program}="/etc/infiniband/vf-net-link-name.sh $attr{phys_switch_id}
 $attr{phys_port_name}" \
        NAME="$env{NAME}", RUN+="/sbin/ethtool -L $env{NAME} combined 4"
# MDEV network interfaces
```

第 1 章. 中介设备

NVIDIA 中介设备在 BlueField 系统上允许创建无需 SR-IOV 的加速设备，提供了灵活性。这些中介设备支持网络接口卡(NIC)和 RDMA，并提供与 SR-IOV 虚拟功能(VFs)相同的 ASAP2 卸载级别。中介设备通过 mlx5 子功能加速技术得到支持。

启动时在 BlueField 设备上创建了两个子功能设备（如果端口处于 switchdev 模式，则每个端口一个）使用/etc/mellanox/mlnx-sf.conf 中的命令：

```
/sbin/mlnx-sf -a create -d 0000:03:00.0 -u 61a59715-aeec-42d5-be83-f8f42ba8b049 -mac 12:11:11:11:11:11
/sbin/mlnx-sf -a create -d 0000:03:00.1 -u 5b198182-1901-4c29-97a0-6623f3d02065 -mac 12:11:11:11:11:12
```

以下为 mlnx-sf 的帮助菜单：

```
用法：mlnx-sf [选项] 选项：-a, -action, --action <操作> 执行操作

 action: { enable | create | configure | remove | show | set_max_mdevs | query_mdevs_num }

-d, -device, --device <device> PCI 设备 <domain>:<bus>:<device>.<func> (例如：0000:03:00.0) -m, -max_mdevs, --
max_mdevs <max mdevs number> 设置 MDEVs 的最大数量 -u, -uuid, --uuid <uuid> 创建 SF 的 UUID -M, -mac, --mac <MAC>
创建 SF 的 MAC -p, -permanent, --permanent [<conf file>] 将配置存储到在重启和/或驱动程序重启后使用的配置文件中。
默认(/etc/mellanox/mlnx-sf.conf)。 -V, -version, --version 显示脚本版本并退出 -D, -dryrun, --dryrun 仅显示命令 -
v, -verbose, --verbose 以详细模式运行脚本（打印执行的每一步） -h, -help, --help 显示帮助
"/etc/mellanox/mlnx-sf.conf" 可以手动更新或使用带有 "p" 参数的 "mlnx-sf" 工具。
```

1.1. 相关配置

接口名称是通过在 /etc/udev/rules.d/82-net- 目录下的 UDEV 规则进行配置的
setup-link.rules .

```
SUBSYSTEM=="net", ACTION=="add", ATTR{phys_switch_id}!="", ATTR{phys_port_name}!="", \

 IMPORT{program}="/etc/infiniband/vf-net-link-name.sh $attr{phys_switch_id} $attr{phys_port_name}" \

名称="$env{NAME}", 运行+="/sbin/ethtool -L $env{NAME} combined 4" # MDEV 网络接口
```

```
ACTION=="add", SUBSYSTEM=="net", DEVPATH=="/devices/
pci0000:00/0000:00:00.0/0000:01:00.0/0000:02:02.0/0000:03:00.0/61a59715-aeec-42d5-
be83-f8f42ba8b049/net/eth[0-9]", NAME="p0m0"
ACTION=="add", SUBSYSTEM=="net", DEVPATH=="/devices/
pci0000:00/0000:00:00.0/0000:01:00.0/0000:02:02.0/0000:03:00.1/5b198182-1901-4c29-97a0-6623f3d0206
net/eth[0-9]", NAME="p1m0"
```

NVMe SNAP uses `p0m0` as its default interface. See `/etc/nvme_snap/sf1.conf`.

```
ACTION=="add", SUBSYSTEM=="net", DEVPATH=="/devices/ pci0000:00/0000:00:00.0/0000:01:00.0/0000:02:02.0/0000:03:00.0/61a59715-aeec-
42d5be83-f8f42ba8b049/net/eth[0-9]", NAME="p0m0" ACTION=="add", SUBSYSTEM=="net", DEVPATH=="/devices/
pci0000:00/0000:00:00.0/0000:01:00.0/0000:02:02.0/0000:03:00.1/5b198182-1901-4c29-97a0-6623f3d02065 net/eth[0-9]", NAME="p1m0"
```

NVMe SNAP 使用 p0m0 作为其默认接口。请参阅/etc/nvme_snap/sf1.conf。

Chapter 2. VirtIO-net Emulated Devices

VirtIO emulated devices enables users to create VirtIO-net emulated PCIe devices in the system where the NVIDIA® BlueField®-2 DPU is connected. This is done by the virtio-net-controller software module present in the DPU. Virtio-net emulated devices allow users to hot plug up to 16 virtio-net PCIe PF Ethernet NIC devices or 504 virtio-net PCI VF Ethernet NIC devices in the host system where the DPU is plugged in.

Note: Currently, only virtio specifications v1.0 and above are supported.

DPU software also enables users to create virtio block PCIe PF and SR-IOV PCIe VF devices. This is covered in the *NVIDIA BlueField SNAP and virtio-blk SNAP Documentation*.

2.1. VirtIO-net Controller

Virtio-net-controller is a systemd service running on the DPU, with a user interface frontend to communicate with the background service. An SF representor is created for each virtio-net device created on the host. Virtio-net controller only uses an SF number ≥1000.

Note: It is important to note that since the controller provides hardware resources and ACKs the request from the host's VirtIO driver, in order to reboot the DPU and host OS, it is necessary to reboot the host OS first, and only then reboot the DPU.

Note: SF representor name is determined by udev rules. The default name is in the format of `<prefix><pf_num><sf_num>`. For example: `en3f0pf0sf1001`.

2.1.1. SystemD Service

Controller systemd service is enabled by default and runs automatically if `VIRTIO_NET_EMULATION_ENABLE` is true from mlxconfig.

1. To check controller service status, run:
   ```
   systemctl status virtio-net-controller.service
   ```
2. To reload the service, make sure to unload virtio-net/virtio-pcie drivers on host. Then run:
   ```
   systemctl restart virtio-net-controller.service
   ```
3. To monitor log output of the controller service, run:
   ```
   journalctl -u virtio-net-controller
   ```

第二章. VirtIO-net 模拟设备

VirtIO 模拟设备使用户能够在连接 NVIDIA® BlueField®-2 DPU 的系统中创建 VirtIO-net 模拟 PCIe 设备。这是通过 DPU 中存在的 virtio-netcontroller 软件模块实现的。VirtIO-net 模拟设备允许用户在插入 DPU 的主机系统中热插拔最多 16 个 virtio-net PCIe PF 以太网网卡设备或 504 个 virtio-net PCI VF 以太网网卡设备。

注意：目前仅支持 virtio 规范 v1.0 及以上版本。

DPU 软件还允许用户创建 virtio 块 PCIe PF 和 SR-IOV PCIe VF 设备。
这在 NVIDIA BlueField SNAP 和 virtio-blk SNAP 文档中有详细说明。

2.1. VirtIO-net 控制器

Virtio-net 控制器是运行在 DPU 上的 systemd 服务，具有与后台服务通信的用户界面前端。对于主机上创建的每个 virtio-net 设备，都会创建一个 SF 表示器。Virtio-net 控制器仅使用 SF 编号≥1000。

注意：由于控制器提供硬件资源并确认来自主机 VirtIO 驱动程序的要求，因此为了重启 DPU 和主机操作系统，必须首先重启主机操作系统，然后才能重启 DPU。

注意：SF 代表者名称由 udev 规则确定。默认名称的格式为<prefix><pf_num><sf_num>。例如：en3f0pf0sf1001。

2.1.1. 系统服务

控制器 systemd 服务默认启用，如果从 mlxconfig 中
VIRTIO_NET_EMULATION_ENABLE 为 true，则自动运行。

1. 检查控制器服务状态，请运行：systemctl status virtio-net-controller.service

2. 要重新加载服务，请确保在主机上卸载 virtio-net/virtio-pcie 驱动程序。然后运行：
   ```
   systemctl restart virtio-net-controller.service
   ```
3. 要监控控制器服务的日志输出，请运行：
   ```
   journalctl -u 虚拟 io 网络控制器
   ```

The controller service has an optional configuration file which allows users to customize several parameters. The configuration file should be defined on the DPU at the following path `/opt/mellanox/mlnx_virtnet/virtnet.conf`.

This file will be read every time the controller starts. Dynamic change of virtnet.conf is not supported. It is defined as a JSON format configuration file. The currently supported options are:

- `ib_dev_p0` – RDMA device (e.g., `mlx5_0`) used to create SF on port 0. This port is the EMU manager when `is_lag` is 0. Default value is `mlx5_0`.
- `ib_dev_p1` – RDMA device (e.g., `mlx5_1`) used to create SF on port 1. Default value is `mlx5_1`.
- `ib_dev_lag` – RDMA LAG device (e.g., `mlx5_bond_0`) used to create SF on LAG. Default value is `mlx5_bond_0`. This port is EMU manager when `is_lag` is 1. `ib_dev_lag` and `ib_dev_p0`/`ib_dev_p1` cannot be configured simultaneously.
- `ib_dev_for_static_pf` – the RDMA device (e.g, mlx5_0) which the static VirtIO PF is created on
- `is_lag` - whether or not LAG is used. Note that if LAG is used, make sure to use the correct IB dev for static PF.
- `pf_mac` – base MAC address for static PFs. MACs are automatically assigned with the following pattern: `pf_mac`→`pf_0`, `pf_mac`+1→`pf_1`, etc.

> **Note:** Note that the controller does not validate the MAC address (other than its length). The user must ensure MAC is valid and unique.

- `recovery` – specifies whether recovery is enabled. If unspecified, recovery is enabled by default. To disable it, set `recovery` to 0.
- `sf_pool_percent` – determines the initial SF pool size as the percentage of `PF_TOTAL_SF` of `mlxconfig`. Valid range: [0, 100]. For instance, if the value is 5, it means an SF pool with 5% of `PF_TOTAL_SF` is created. 0 means no SF pool is reserved beforehand (default).

> **Note:** `PF_TOTAL_SF` is shared by all applications. User must ensure the percent request is guaranteed or else the controller will not be able to reserve the requested SFs resulting in failure.

- `sf_pool_force_destroy` – specifies whether to destroy the SF pool. When set to 1, the controller destroys the SF pool when stopped/restarted (and the SF pool is recreated if `sf_pool_percent` is not 0 when starting), otherwise it does not. Default value is 0.

For example, the definition below has all static PFs using mlx5_0 (port 0) as the data path device in a non-lag configuration.

```
{
  "ib_dev_p0": "mlx5_0",
  "ib_dev_p1": "mlx5_1",
  "ib_dev_for_static_pf": "mlx5_0",
  "is_lag": 0,
  "pf_mac": "00:11:22:33:44:55",
  "recovery": 1,
  "sf_pool_percent": 0,
  "sf_pool_force_destroy": 0
}
```

控制器服务有一个可选的配置文件，允许用户自定义多个参数。该配置文件应在 DPU 上的以下路径定义
/opt/mellanox/mlnx_virtnet/virtnet.conf.

此文件将在控制器启动时每次读取。不支持 virtnet.conf 的动态更改。它被定义为 JSON 格式的配置文件。当前支持选项包括：

ib_dev_p0 – 用于在端口 0 上创建 SF 的 RDMA 设备（例如，mlx5_0）。当 is_lag 为 0 时，此端口是 EMU 管理器。默认值为 mlx5_0。

ib_dev_p1 – 用于在端口 1 上创建 SF 的 RDMA 设备（例如，mlx5_1）。默认值为 mlx5_1。

ib_dev_lag – 用于在 LAG 上创建 SF 的 RDMA LAG 设备（例如，mlx5_bond_0）。默认值为 mlx5_bond_0。当 is_lag 为 1 时，此端口是 EMU 管理器。ib_dev_lag 和 ib_dev_p0/ib_dev_p1 不能同时配置。

ib_dev_for_static_pf – 静态 VirtIO PF 创建在其上的 RDMA 设备（例如，mlx5_0）

是否使用 LAG。注意，如果使用 LAG，请确保使用正确的 IB 设备进行静态 PF。

静态 PF 的基准 MAC 地址。MAC 地址将自动按照以下模式分配：pf_mac→pf_0，pf_mac+1→pf_1，等等。

注意：控制器不会验证 MAC 地址（除了长度之外）。
用户必须确保 MAC 地址有效且唯一。

恢复 – 指定是否启用恢复。如果未指定，则默认启用恢复。要禁用它，请将恢复设置为 0。

‣sf_pool_percent – 确定初始 SF 池大小为 mlxconfig 中 PF_TOTAL_SF 的百分比。有效范围：[0, 100]。例如，如果值为 5，则表示创建一个占 PF_TOTAL_SF 5%的 SF 池。
5%的 PF_TOTAL_SF 被创建。0 表示事先不预留 SF 池（默认）。

注意：PF_TOTAL_SF 被所有应用程序共享。用户必须确保请求的百分比得到保证，否则控制器将无法保留请求的 SFs，从而导致失败。

‣sf_pool_force_destroy – 指定是否销毁 SF 池。当设置为 1 时，控制器在停止/重启时销毁 SF 池（如果 sf_pool_percent 在启动时不是 0，则重新创建 SF 池），否则不销毁。默认值为 0。

例如，以下定义中所有静态 PF 使用 mlx5_0（端口 0）作为非 LAG 配置的数据路径设备。

```
{ "ib_dev_p0": "mlx5_0", "ib_dev_p1": "mlx5_1",
"ib_dev_for_static_pf": "mlx5_0", "is_lag": 0,
"pf_mac": "00:11:22:33:44:55", "recovery": 1,
"sf_pool_percent": 0, "sf_pool_force_destroy": 0
}
```

The following is an example for LAG configuration:

```
{
  "ib_dev_lag": "mlx5_bond_0",
  "ib_dev_for_static_pf": "mlx5_bond_0",
  "is_lag": 1,
  "pf_mac": "00:11:22:33:44:55",
  "recovery": 1,
  "sf_pool_percent": 0,
  "sf_pool_force_destroy": 0
}
```

2.1.2. User Frontend

To communicate with the service, a user frontend program, virtnet, is installed on the DPU. Run the following command to check its usage:

```
# virtnet -h
usage: virtnet [-h] [-v] {hotplug,unplug,list,query,modify,log} ...

Nvidia virtio-net-controller command line interface v1.0.9

positional arguments:
  {hotplug,unplug,list,query,modify,log}
                        ** Use -h for sub-command usage
    hotplug             hotplug virtnet device
    unplug              unplug virtnet device
    list                list all virtnet devices
    query               query all or individual virtnet device(s)
    modify              modify virtnet device
    log                 set log level

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit
```

Note that each positional argument has its own help menu as well. For example:

```
# virtnet log -h
usage: virtnet log [-h] -l {info,err,debug}
optional arguments:
  -h, --help            show this help message and exit
  -l {info,err,debug}, --level {info,err,debug}
                        log level: info/err/debug
```

To operate a particular device, either the VUID or device index can be used to locate the device. Both attributes can be fetched from command "virtnet list". For example, to modify the MAC of a specific VF, you may run either of the following commands:

```
# virtnet modify -p 0 -v 0 device -m 0C:C4:7A:FF:22:98
```

Or:

```
# virtnet modify -u <VUID-string> device -m 0C:C4:7A:FF:22:98
```

2.1.3. Controller Recovery

It is possible to recover the control and data planes if communications are interrupted so the original traffic can resume.

Recovery depends on the JSON files stored in `/opt/mellanox/mlnx_virtnet/recovery` where there is a file that corresponds to each device (either PF or VF). The following is an example of the data stored in these files:

```
{
  "port_ib_dev": "mlx5_0",
```

以下是一个 LAG 配置的示例：

```
{ "ib_dev_lag": "mlx5_bond_0", "ib_dev_for_static_pf":
"mlx5_bond_0", "is_lag": 1, "pf_mac":
"00:11:22:33:44:55", "recovery": 1, "sf_pool_percent":
0, "sf_pool_force_destroy": 0 }
```

2.1.2. 用户前端

与服务通信时，需要在 DPU 上安装用户前端程序 virtnet。运行以下命令以检查其使用情况：

```
virtnet -h 使用方法：virtnet [-h] [-v] {hotplug,unplug,list,query,modify,log} ...

Nvidia virtio-net-controller 命令行界面 v1.0.9

位置参数：{hotplug,unplug,list,query,modify,log}
                                             ** 使用 -h 查看子命令用法
        热插拔 热插拔 virtnet 设备 断开 断开 virtnet 设备 列表 列出所有 virtnet 设备 查询
        查询所有或单个 virtnet 设备 修改 修改 virtnet 设备 日志 设置日志级别

可选参数：
   -h, --help 显示此帮助信息并退出 -v, --version 显示程序版本信息并退出
```

注意，每个位置参数都有自己的帮助菜单。例如：

```
# virtnet log -h 使用方法：virtnet log [-h] -l {info,err,debug} 可选参数：-
h, --help 显示此帮助信息并退出 -l {info,err,debug}, --level
{info,err,debug}
                                             日志级别：info/err/debug
```

要操作特定设备，可以使用 VUID 或设备索引来定位设备。这两个属性都可以从"virtnet list"命令中获取。例如，要修改特定 VF 的 MAC 地址，可以运行以下任一命令：

```
# virtnet modify -p 0 -v 0 device -m 0C:C4:7A:FF:22:98
```

Or:

```
# virtnet modify -u <VUID-string> device -m 0C:C4:7A:FF:22:98
```

2.1.3. 控制器恢复

如果通信中断，则可以恢复控制面和数据面，以便原始流量可以继续。

恢复依赖于存储在/opt/mellanox/mlnx_virtnet/recovery 中的 JSON 文件，其中每个设备（无论是 PF 还是 VF）都对应一个文件。以下是在这些文件中存储的数据示例：

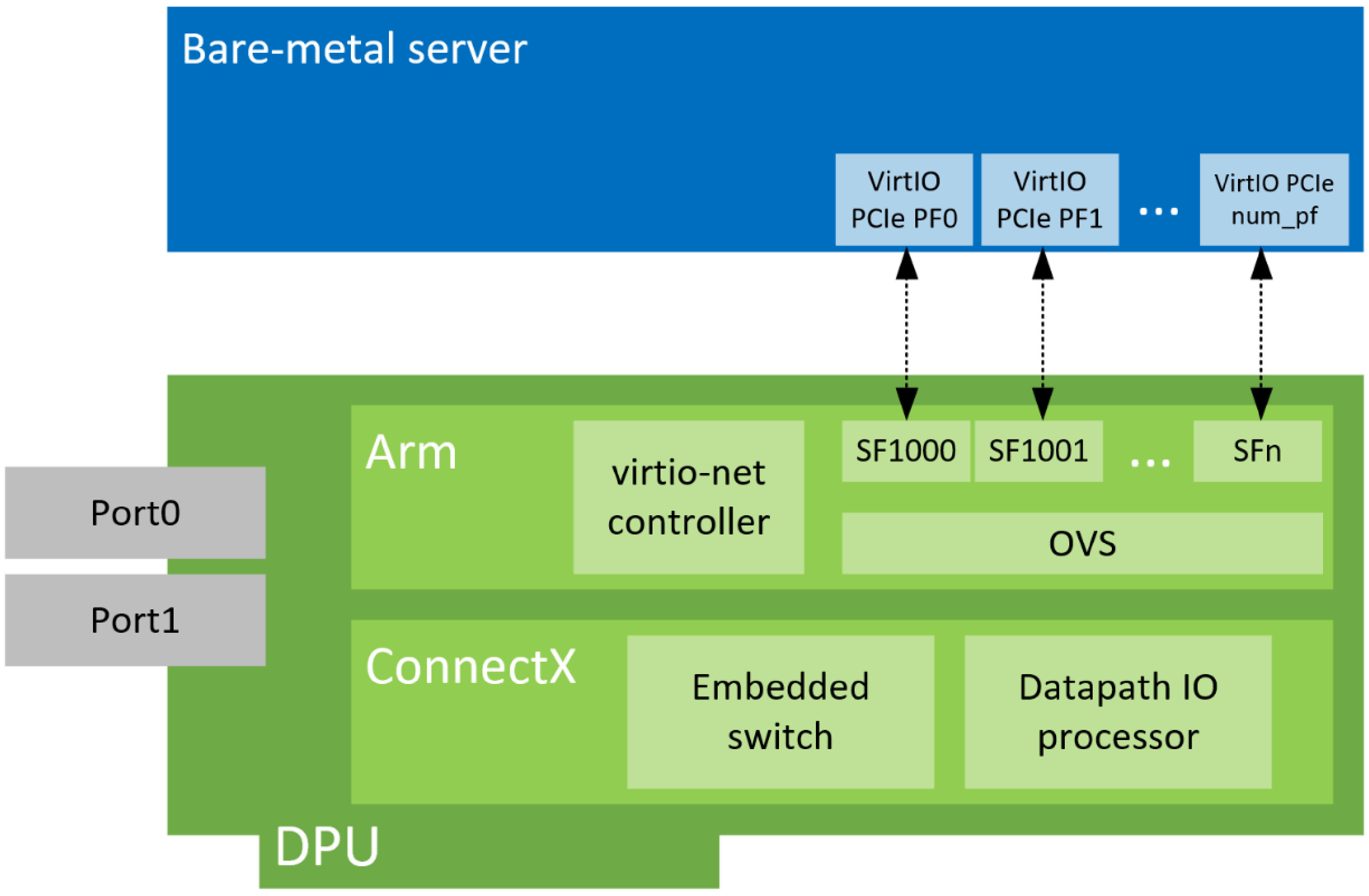
```
{ "port_ib_dev": "mlx5_0",
  "pf_id": 0,
  "function_type": "pf",
  "bdf_raw": 26624,
  "device_type": "hotplug",
  "mac": "0c:c4:7a:ff:22:93",
  "pf_num": 0,
  "sf_num": 2000,
  "mq": 1
}
```

These files should not be modified under normal circumstances. However, if necessary, advanced users may tune settings to meet their requirements. Users are responsible for the validity of the recovery files and should only perform this when the controller is not running.

Note: Controller recovery is enabled by default and does not need user configuration or intervention unless a system reset is needed or BlueField configuration is changed (i.e. any of the mlxconfig options `PCI_SWITCH_EMULATION_NUM_PORT`, `VIRTIO_NET_EMULATION_NUM_VF`, or `VIRTIO_NET_EMULATION_NUM_PF`). To this end, the files under `/opt/mellanox/mlnx_virtnet/recovery` must be deleted.

2.2. VirtIO-net PF Devices

This section covers managing virtio-net PCIe PF devices using virtio-net-controller.



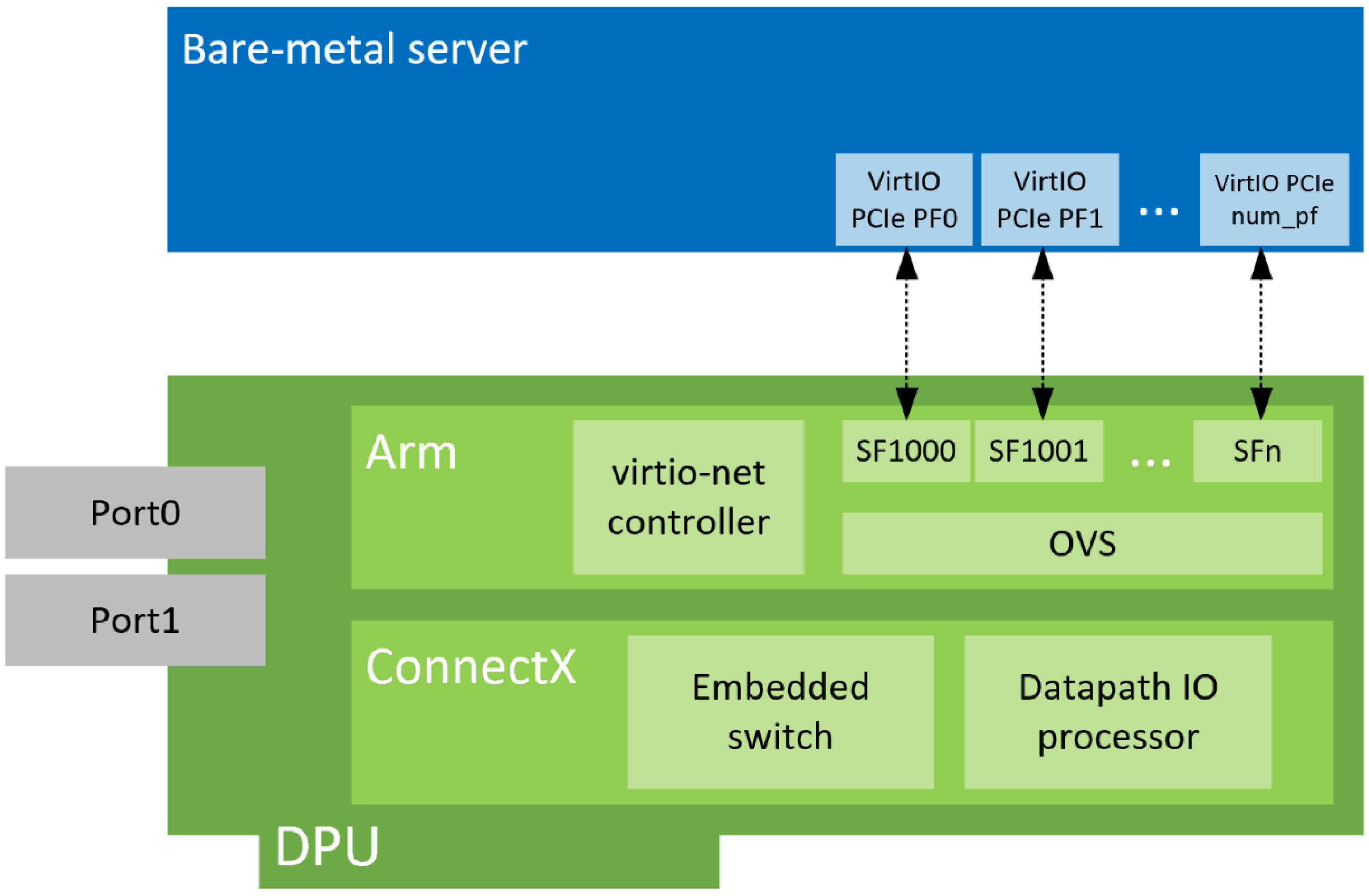
```
"pf_id": 0, "function_type": "pf",
"bdf_raw": 26624, "device_type":
"hotplug", "mac": "0c:c4:7a:ff:22:93",
"pf_num": 0, "sf_num": 2000, "mq": 1 }
```

在正常情况下，这些文件不应被修改。但是，如果需要，高级用户可以调整设置以满足其需求。用户应负责恢复文件的有效性，并且只有在控制器未运行时才应执行此操作。

注意：控制器恢复默认启用，无需用户配置或干预，除非需要系统重置或更改 BlueField 配置（即 mlxconfig 选项 PCI_SWITCH_EMULATION_NUM_PORT、VIRTIO_NET_EMULATION_NUM_VF 或 VIRTIO_NET_EMULATION_NUM_PF 中的任何一项）。为此，必须删除/opt/mellanox/mlnx_virtnet/recovery 下的文件。

2.2. VirtIO-net PF 设备

本节介绍使用 virtio-net 控制器管理 virtio-net PCIe PF 设备。



2.2.1. VirtIO-net PF Device Configuration

1. Run the following command on the DPU:
```
mlxconfig -d /dev/mst/mt41686_pciconf0 s INTERNAL_CPU_MODEL=1
```
2. Add the following kernel boot parameters to the Linux boot arguments:
```
intel_iommu=on iommu=pt pci=realloc
```
3. Cold reboot the host system.
4. Apply the following configuration on the DPU:
```
$ mst start

$ mlxconfig -d /dev/mst/mt41686_pciconf0 s PF_BAR2_ENABLE=0 PER_PF_NUM_SF=1

$ mlxconfig -d /dev/mst/mt41686_pciconf0 s \
PCI_SWITCH_EMULATION_ENABLE=1 \
PCI_SWITCH_EMULATION_NUM_PORT=16 \
VIRTIO_NET_EMULATION_ENABLE=1 \
VIRTIO_NET_EMULATION_NUM_VF=0 \
VIRTIO_NET_EMULATION_NUM_PF=0 \
VIRTIO_NET_EMULATION_NUM_MSIX=10 \
VIRTIO_NET_EMULATION_VENDOR_ID=0x1af4 \
VIRTIO_NET_EMULATION_DEVICE_ID=0x1041 \
VIRTIO_NET_EMULATION_CLASS_CODE=0x028000 \
ECPF_ESWITCH_MANAGER=1 \
ECPF_PAGE_SUPPLIER=1 \
SRIOV_EN=0 \
PF_SF_BAR_SIZE=10 \
PF_TOTAL_SF=64

$ mlxconfig -d /dev/mst/mt41686_pciconf0.1 s \
PF_SF_BAR_SIZE=10 \
PF_TOTAL_SF=64
```
5. Cold reboot the host system a second time.

2.2.2. Creating Hotplug VirtIO-net PF Device

VirtIO emulated network PCIe devices are created and destroyed using virtio-net-controller application console. When this application is terminated, all created virtio-net emulated devices are hot unplugged.

1. Create a hotplug virtio-net device. Run:
```
virtnet hotplug -i mlx5_0 -f 0x0 -m 0C:C4:7A:FF:22:93 -t 1500 -n 3 -s 1024
```

Note: The maximum number of virtio-net queues is bound by the minimum of the following numbers:

- `VIRTIO_NET_EMULATION_NUM_MSIX` from the command `mlxconfig -d <mst_dev> q`
- `max_virtq` from the command `virtnet list`

This creates one hotplug virtio-net device with MAC address 0C:C4:7A:FF:22:93, MTU 1500, and 3 VirtIO queues with a depth of 1024 entries. This device is uniquely identified by its index. This index is used to query and update device attributes. If the device is created successfully, an output appears similar to the following:
```
{
  "bdf": "85:00.0",
```

2.2.1. VirtIO-net PF 设备配置

1. 在 DPU 上运行以下命令：
```
mlxconfig -d /dev/mst/mt41686_pciconf0 s INTERNAL_CPU_MODEL=1
```
2. 将以下内核引导参数添加到 Linux 引导参数中：
```
intel_iommu=on iommu=pt pci=realloc
```
3. 对主机系统进行冷重启。
4. 在 DPU 上应用以下配置：
```
$ mst start

$ mlxconfig -d /dev/mst/mt41686_pciconf0 s PF_BAR2_ENABLE=0 PER_PF_NUM_SF=1

$ mlxconfig -d /dev/mst/mt41686_pciconf0 s
PCI_SWITCH_EMULATION_ENABLE=1
PCI_SWITCH_EMULATION_NUM_PORT=16
VIRTIO_NET_EMULATION_ENABLE=1 VIRTIO_NET_EMULATION_NUM_VF=0

VIRTIO_NET_EMULATION_NUM_PF=0
VIRTIO_NET_EMULATION_NUM_MSIX=10
VIRTIO_NET_EMULATION_VENDOR_ID=0x1af4
VIRTIO_NET_EMULATION_DEVICE_ID=0x1041
VIRTIO_NET_EMULATION_CLASS_CODE=0x028000
ECPF_ESWITCH_MANAGER=1
ECPF_PAGE_SUPPLIER=1 \
SRIOV_EN=0 \ PF_SF_BAR_SIZE=10
\ PF_TOTAL_SF=64


$ mlxconfig -d /dev/mst/mt41686_pciconf0.1 s \
PF_SF_BAR_SIZE=10 \ PF_TOTAL_SF=64
```
5. 再次冷重启主机系统。

2.2.2. 创建热插拔 VirtIO-net PF 设备

VirtIO 虚拟网络 PCIe 设备通过 virtio-net-controller 应用程序控制台创建和销毁。当此应用程序终止时，所有创建的 virtio-net 虚拟设备都会进行热插拔断开。

1. 创建热插拔的 virtio-net 设备。运行：virtnet hotplug -i mlx5_0 -f 0x0 -m 0C:C4:7A:FF:22:93 -t 1500 -n 3 -s 1024

注意：virtio-net 队列的最大数量受以下数字中的最小值限制：

- 从命令 mlxconfig -d <mst_dev> q 中获取 VIRTIO_NET_EMULATION_NUM_MSIX

从命令行

这将创建一个具有 MAC 地址 0C:C4:7A:FF:22:93、MTU 1500 和 3 个深度为 1024 条目的 VirtIO 队列的热插拔 virtio-net 设备。该设备通过其索引唯一标识。此索引用于查询和更新设备属性。如果设备创建成功，输出将类似于以下内容：

```
{ "bdf": "85:00.0", }
```

```
  "vuid": "VNETS1D0F0",
  "id": 3,
  "sf_rep_net_device": "en3f0pf0sf2000",
  "mac": "0C:C4:7A:FF:22:93"
}
```

2. Add the representor port of the device to the OVS bridge and bring it up. Run:

   ```
   ip link set dev en3f0pf0sf1001 up
   ovs-vsctl add-port <bridge> en3f0pf0sf1001
   ```

   Once steps 1-3 are completed, virtio-net device should be available in the host system.
3. To query all the device configurations of virtio-net device that you created, run:

   ```
   virtnet query -p 1
   ```
4. To list all the virtio-net devices, run:

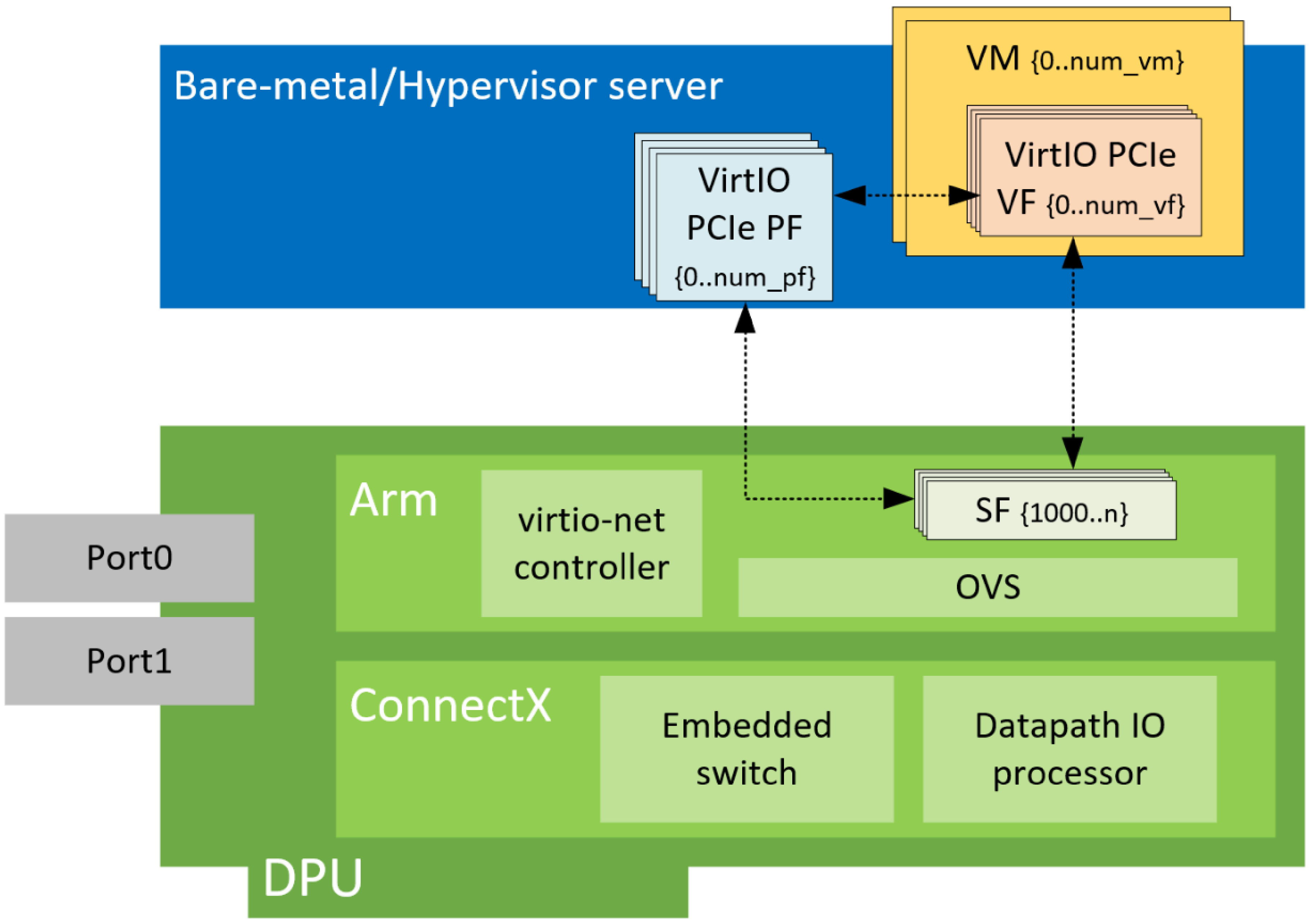
   ```
   virtnet list
   ```
5. To modify device attributes, for example, changing its MAC address, run:

   ```
   virtnet modify -p 0 device -m 0C:C4:7A:FF:22:98
   ```
6. Once usage is complete, to hot-unplug a VirtIO net device, run:

   ```
   virtnet unplug -p 1
   ```

2.3. VirtIO-net SR-IOV VF Devices

This section covers managing virtio-net PCIe SR-IOV VF devices using virtio-net-controller.

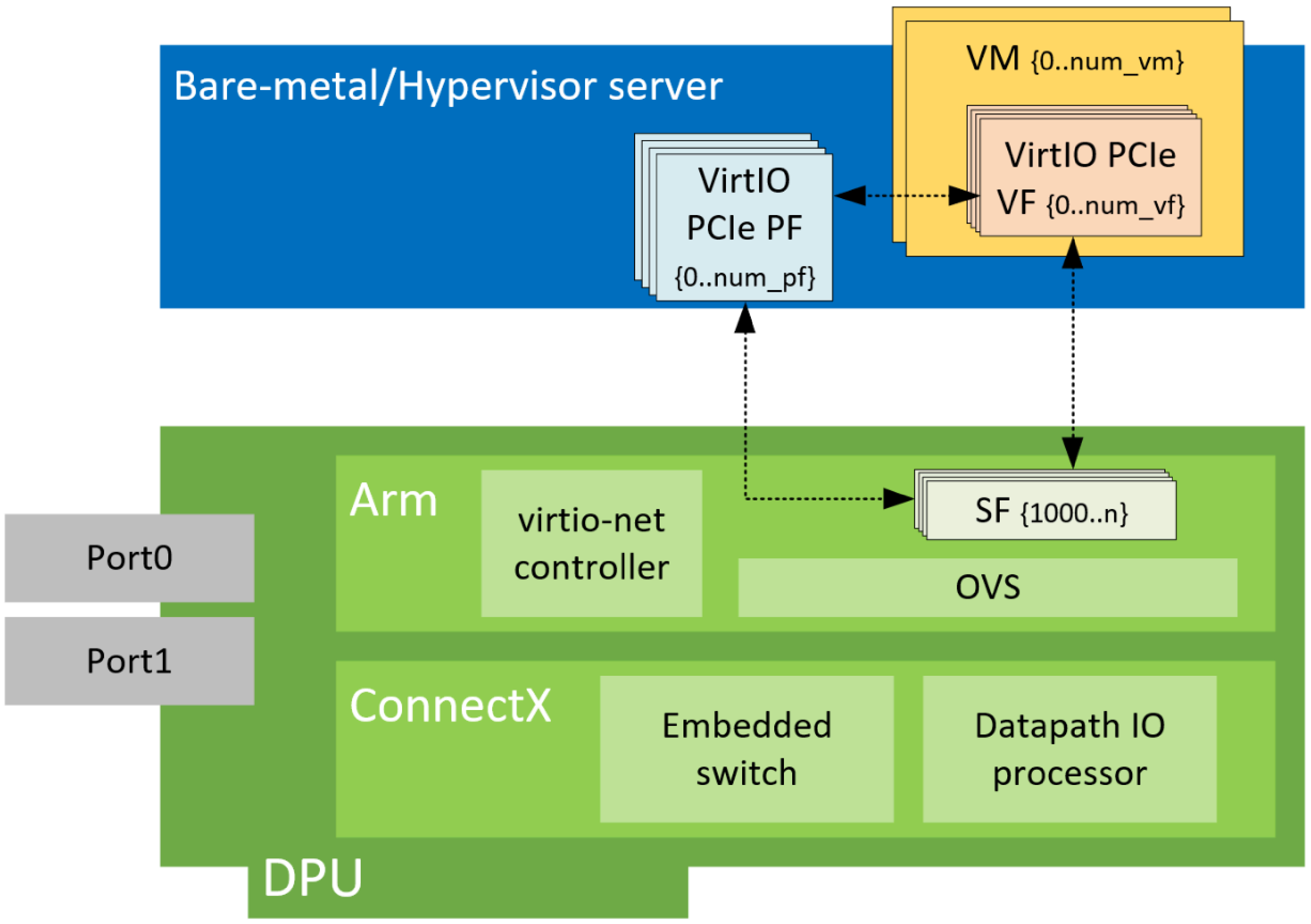


```
{"vuid": "VNETS1D0F0", "id": 3, "sf_rep_net_device":
"en3f0pf0sf2000", "mac": "0C:C4:7A:FF:22:93"}
```

2. 将设备的代表端口添加到 OVS 桥并启动。运行：
```
ip link set dev en3f0pf0sf1001 up ovs-vsctl add-port
<bridge> en3f0pf0sf1001
```
   完成步骤 1-3 后，virtio-net 设备应在主机系统中可用。
3. 要查询您创建的 virtio-net 设备的所有设备配置，请运行：
```
virtnet query  -p 1
```
4. 要列出所有 virtio-net 设备，请运行：
```
virtnet 列表
```
5. 要修改设备属性，例如更改其 MAC 地址，请运行：
```
virtnet modify -p 0 device -m 0C:C4:7A:FF:22:98
```
6. 使用完毕后，要热插拔 VirtIO 网络设备，请运行：
```
virtnet 断开连接 -p 1
```

2.3. VirtIO-net SR-IOV 虚拟功能设备

本节介绍使用 virtio-net 控制器管理 virtio-net PCIe SR-IOV 虚拟功能设备。



2.3.1. Virtio-net SR-IOV VF Device Configuration

Note: Virtio-net SR-IOV VF is only supported with statically configured PF, hot-plugged PF is not currently supported.

1. On the DPU, make sure virtio-net-controller service is enabled so that it starts automatically. Run:
   ```
   systemctl status virtio-net-controller.service
   ```
2. On the x86 host, enable SR-IOV. Please refer to MLNX_OFED documentation under Features Overview and Configuration > Virtualization > Single Root IO Virtualization (SR-IOV) > Setting Up SR-IOV for instructions on how to do that. Make sure the parameters `intel_iommu=on iommu=pt pci=realloc` exist in `grub.conf` file.
3. It is recommended to add `pci=assign-busses` to the boot command line when creating more than 127 VFs. Without this option, the following errors might appear from host and the virtio driver will not probe these devices.
   ```
   pci 0000:84:00.0: [1af4:1041] type 7f class 0xffffff
   pci 0000:84:00.0: unknown header type 7f, ignoring device
   ```
4. Run the following command on the DPU:
   ```
   mst start && mlxconfig -d /dev/mst/mt41686_pciconf0 s INTERNAL_CPU_MODEL=1
   ```
5. Cold reboot the host system.
6. Apply the following configuration on the DPU in three steps to support up to 125 VFs per PF (500 VFs in total).

   a).
   ```
   mst start && mlxconfig -d /dev/mst/mt41686_pciconf0 s PF_BAR2_ENABLE=0
    PER_PF_NUM_SF=1
   ```
   b).
   ```
   mlxconfig -d /dev/mst/mt41686_pciconf0 s \
   PCI_SWITCH_EMULATION_ENABLE=1 \
   PCI_SWITCH_EMULATION_NUM_PORT=16 \
   VIRTIO_NET_EMULATION_ENABLE=1 \
   VIRTIO_NET_EMULATION_NUM_VF=126 \
   VIRTIO_NET_EMULATION_NUM_PF=4 \
   VIRTIO_NET_EMULATION_NUM_MSIX=4 \
   VIRTIO_NET_EMULATION_VENDOR_ID=0x1af4 \
   VIRTIO_NET_EMULATION_DEVICE_ID=0x1041 \
   VIRTIO_NET_EMULATION_CLASS_CODE=0x028000 \
   ECPF_ESWITCH_MANAGER=1 \
   ECPF_PAGE_SUPPLIER=1 \
   SRIOV_EN=1 \
   PF_SF_BAR_SIZE=8 \
   PF_TOTAL_SF=508 \
   NUM_OF_VFS=0
   ```
   c).
   ```
   mlxconfig -d /dev/mst/mt41686_pciconf0.1 s PF_TOTAL_SF=1 PF_SF_BAR_SIZE=8
   ```
7. Cold reboot the host system.

2.3.2. Creating Virtio-net SR-IOV VF Devices

The virtio-net-controller application console must be kept alive to maintain the functionality of the static PF and its VFs.

1. On the host, make sure the static virtio network device presents. Run:
   ```
   # lspci | grep -i virtio
   85:00.3 Network controller: Red Hat, Inc. Virtio network device
   ```

2.3.1. Virtio-net SR-IOV VF 设备配置

注意：Virtio-net SR-IOV VF 仅支持静态配置的 PF，目前不支持热插拔 PF。

1. 在 DPU 上，请确保 virtio-net-controller 服务已启用，以便自动启动。运行：

```
systemctl 状态 virtio-net-controller.service
```

2. 在 x86 主机上启用 SR-IOV。请参考 MLNX_OFED 文档中的功能概述和配置 > 虚拟化 > 单根 IO 虚拟化（SRIOV） > 设置 SR-IOV 的说明，了解如何进行操作。确保参数

   intel_iommu=on iommu=pt pci=realloc 存在于 grub.conf 文件中。

3. 建议在创建超过 127 个 VF 时，将 pci=assign-busses 添加到引导命令行中。没有此选项，可能会从主机和 virtio 驱动程序中出现以下错误。

```
pci 0000:84:00.0: [1af4:1041] 类型 7f 类 0xffffff pci 0000:84:00.0: 未知头部类
型 7f，忽略设备
```

4. 在 DPU 上运行以下命令：

```
mst start && mlxconfig -d /dev/mst/mt41686_pciconf0 s INTERNAL_CPU_MODEL=1
```

5. 对主机系统进行冷重启。
6. 在 DPU 上分三步应用以下配置，以支持每个 PF 最多 125 个 VF（总计 500 个 VF）。

a).
```
                              /dev/mst/mt41686_pciconf0 s PF_BAR2_ENABLE=0
 PER_PF_NUM_SF=1
```

b).
```
/dev/mst/mt41686_pciconf0 s \
PCI_SWITCH_EMULATION_ENABLE=1 \
PCI_SWITCH_EMULATION_NUM_PORT=16 \
VIRTIO_NET_EMULATION_ENABLE=1 \
VIRTIO_NET_EMULATION_NUM_VF=126 \
VIRTIO_NET_EMULATION_NUM_PF=4 \
VIRTIO_NET_EMULATION_NUM_MSIX=4 \
VIRTIO_NET_EMULATION_VENDOR_ID=0x1af4 \
VIRTIO_NET_EMULATION_DEVICE_ID=0x1041 \
VIRTIO_NET_EMULATION_CLASS_CODE=0x028000 \
ECPF_ESWITCH_MANAGER=1 \
ECPF_PAGE_SUPPLIER=1 \
SRIOV_EN=1 \ PF_SF_BAR_SIZE=8 \
PF_TOTAL_SF=508 \ NUM_OF_VFS=0
```

c).
```
                    /dev/mst/mt41686_pciconf0.1 s PF_TOTAL_SF=1 PF_SF_BAR_SIZE=8
```

7. 冷重启主机系统。

2.3.2. 创建 Virtio-net SR-IOV 虚拟功能设备。

虚拟 io 网络控制器应用程序控制台必须保持活跃，以维持静态 PF 及其 VF 的功能。

1. 在主机上，确保静态虚拟 io 网络设备存在。运行：# lspci | grep -i virtio

```
85:00.3 网络控制器：红帽公司 Virtio 网络设备
```

2. On the host, make sure virtio_pci and virtio_net are loaded. Run:

   ```
   # lsmod | grep virtio
   ```

   The net device should be created:

   ```
   # ethtool -i p7p3
   driver: virtio_net
   version: 1.0.0
   firmware-version:
   expansion-rom-version:
   bus-info: 0000:85:00.3
   supports-statistics: no
   supports-test: no
   supports-eeprom-access: no
   supports-register-dump: no
   supports-priv-flags: no
   ```

3. To create SR-IOV VF devices on the x86 host, run:

   ```
   # echo 2 > /sys/bus/pci/drivers/virtio-pci/0000\:85\:00.3/sriov_numvfs
   ```

   Note: When the number of VFs created is high, SR-IOV enablement may take several minutes.

   2 VFs should be created from the x86 host:

   ```
   # lspci | grep -i virt
   85:00.3 Network controller: Red Hat, Inc. Virtio network device
   85:04.5 Network controller: Red Hat, Inc. Virtio network device
   85:04.6 Network controller: Red Hat, Inc. Virtio network device
   ```

4. From the DPU virtio-net controller, run the following command to get VF information.

   ```
   # virnet list
   {
       "vf_id": 0,
       "parent_pf_id": 0,
       "function_type": "VF",
       "vuid": "VNETS0D0F2VF1",
       "bdf": "83:00.6",
       "sf_num": 3000,
       "sf_parent_device": "mlx5_0",
       "sf_rep_net_device": "en3f0pf0sf3000",
       "sf_rep_net_ifindex": 19,
       "sf_rdma_device": "mlx5_7",
       "sf_vhca_id": "0x192",
       "msix_config_vector": "0x0",
       "num_msix": 10,
       "max_queues": 4,
       "max_queues_size": 256,
       "net_mac": "5A:94:07:04:F6:1C",
       "net_mtu": 1500
   },
   ```

   You may use the pci-bdf to match the PF/VF on the x86 host to the information showing on DPU.

   To query all the device configurations of the virtio-net device of that VF, run:

   ```
   $ virtnet query -p 0 -v 0
   ```

   Add the corresponding SF representor to the OVS bridge and bring it up. Run:

   ```
   # ovs-vsctl add-port <bridge> en3f0pf0sf1004
   # ip link set dev en3f0pf0sf1004 up
   ```

2. 在主机上，请确保已加载 virtio_pci 和 virtio_net。运行：

```
# lsmod | grep virtio
```

网络设备应该被创建：

```
# ethtool -i p7p3
驱动程序: virtio_net
版本: 1.0.0
固件版本: 扩展固件版本: 总线信
息: 0000:85:00.3 支持统计: 否 支
持测试: 否


不支持 EEPROM 访问：否 不支持寄存器
转储：否 不支持私有标志：否
```

3. 在 x86 主机上创建 SR-IOV 虚拟功能（VF）设备，请运行：

```
# echo 2 > /sys/bus/pci/drivers/virtio-pci/0000:85:00.3/sriov_numvfs
```

注意：当创建的 VF 数量较多时，SR-IOV 启用可能需要几分钟。

应从 x86 主机创建 2 个虚拟函数：

```
# lspci | grep -i virt 85:00.3 网络控制器：红帽公司 Virtio 网络设备 85:04.5 网络控制
器：红帽公司 Virtio 网络设备 85:04.6 网络控制器：红帽公司 Virtio 网络设备
```

4. 从 DPU 的 virtio-net 控制器运行以下命令以获取虚拟函数信息。

```
# virnet list
{
     "vf_id": 0,
"parent_pf_id": 0, "function_type": "VF", "vuid":
"VNETS0D0F2VF1", "bdf": "83:00.6", "sf_num": 3000,
"sf_parent_device": "mlx5_0", "sf_rep_net_device":
"en3f0pf0sf3000", "sf_rep_net_ifindex": 19,
"sf_rdma_device": "mlx5_7", "sf_vhca_id": "0x192",
"msix_config_vector": "0x0", "num_msix": 10,
"max_queues": 4, "max_queues_size": 256, "net_mac":
"5A:94:07:04:F6:1C", "net_mtu": 1500 }
```

您可以使用 pci-bdf 来匹配 x86 主机上的 PF/VF 与 DPU 上显示的信息。

要查询该 VF 的 virtio-net 设备的所有设备配置，请运行：

```
$ virtnet 查询 -p 0 -v 0
```

将相应的 SF 表示器添加到 OVS 桥并启动。运行：

```
# ovs-vsctl add-port <桥> en3f0pf0sf1004 # ip link set dev
en3f0pf0sf1004 up
```

Now the VF should be functional.

Note: When port MTU (p0/p1 of the DPU) is changed after the controller is started, you must restart controller service. It is not recommended to use jumbo MTUs because that may lead to performance degradation.

5. To destroy SR-IOV VF devices on the host, run:

```
# echo 0 > /sys/bus/pci/drivers/virtio-pci/0000\:85\:00.3/sriov_numvfs
```

Note: When the command returns from the host OS, it does not necessarily mean the controller finished its operations. Look at controller log from the DPU and make sure you see a log like below before removing VirtIO kernel modules or recreate VFs.

```
# virtio-net-controller[3544]: [INFO]
 virtnet.c:617:virtnet_device_vfs_unload: PF(0): Unload (4) VFs finished
```

Once VFs are destroyed, created SFs from the DPU side are not destroyed but are saved into the SF pool to be reused later.

现在 VF 应该已经可用。

注意：在控制器启动后更改端口 MTU（DPU 的 p0/p1 端口）后，必须重新启动控制器服务。不建议使用巨帧 MTU，因为这可能会导致性能下降。

5. 要销毁主机上的 SR-IOV VF 设备，请运行：

```
# echo 0 > /sys/bus/pci/drivers/virtio-pci/0000:85:00.3/sriov_numvfs
```

注意：当命令从主机操作系统返回时，这并不一定意味着控制器已经完成了其操作。请查看 DPU 的控制台日志，确保您看到以下日志内容后再移除 VirtIO 内核模块或重新创建 VF。

```
# virtio-net-controller[3544]: [INFO] virtnet.c:617:virtnet_device_vfs_unload: PF(0): 卸载（4）VF
完成
```

一旦 VF 被销毁，从 DPU 侧创建的 SF 不会销毁，而是保存到 SF 池中以供以后重用。

NVIDIA Corporation | 加州圣克拉拉市桑托斯快速路 2788 号 95051
http://www.nvidia.com