NVIDIA DOCA OVS DOCA

User Guide

MLNX-15-060597 _v2.0.2 | May 2023

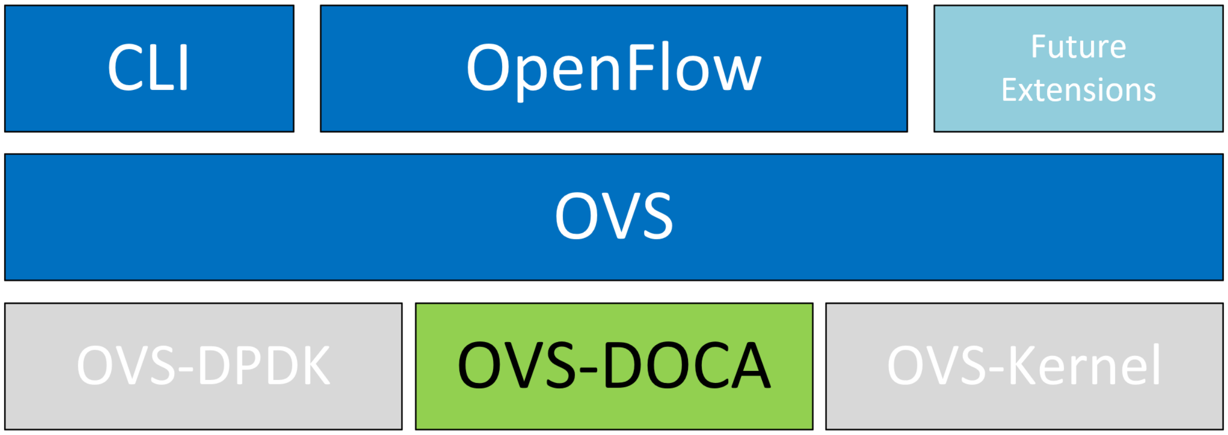
Table of Contents

Chapter 1. Introduction

The NVIDIA DOCA package includes an Open vSwitch (OVS) application designed to work with NVIDIA NICs and utilize ASAP^2 technology for data-path acceleration. This application supports three modes: OVS-Kernel and OVS-DPDK, which are the common modes, and an OVS-DOCA mode which leverages the DOCA Flow library to configure the e-switch and utilize hardware offload mechanisms and application techniques that are not available in the other two modes.

All three operation modes make use of flow offloads for hardware acceleration, but due to its architecture and use of DOCA libraries, the OVS-DOCA mode provides the most efficient performance and feature set among them.

NVIDIA Accelerated Switching And Packet Processing (ASAP^2) technology allows OVS offloading by handling OVS data-plane in the eSwitch hardware while maintaining OVS control-plane unmodified, resulting in higher OVS performance without the associated CPU load.



As seen in the diagram, the 3 OVS flavors preserve the same northbound API, OpenFlow, CLI and data interfaces (e.g., vDPA, VF passthrough).

This document expands on the usage of OVS-DOCA mode. For more information on OVS-Kernel and OVS-DPDK, please refer to NVIDIA MLNX_OFED Documentation.

Chapter 2. OVS-DOCA Mode

The following subsections provide the necessary steps to launch/deploy OVS DOCA.

2.1. Configuring OVS-DOCA

To configure OVS DOCA HW offloads:

1. Unbind the VFs:

   ```
   echo 0000:04:00.2 > /sys/bus/pci/drivers/mlx5_core/unbind
   echo 0000:04:00.3 > /sys/bus/pci/drivers/mlx5_core/unbind
   ```

   Note: VMs with attached VFs must be powered off to be able to unbind the VFs.

2. Change the e-switch mode from `legacy` to `switchdev` on the PF device (make sure all VFs are unbound):

   ```
   echo switchdev > /sys/class/net/enp4s0f0/compat/devlink/mode
   ```

   Note: This command also creates the VF representor netdevices in the host OS.

   To revert to SR-IOV `legacy` mode:

   ```
   echo legacy > /sys/class/net/enp4s0f0/compat/devlink/mode
   ```

3. Bind the VFs:

   ```
   echo 0000:04:00.2 > /sys/bus/pci/drivers/mlx5_core/bind
   echo 0000:04:00.3 > /sys/bus/pci/drivers/mlx5_core/bind
   ```

4. Configure huge pages:

   ```
   mkdir -p /hugepages
   mount -t hugetlbfs hugetlbfs /hugepages
   echo 4096 > /sys/devices/system/node/node0/hugepages/hugepages-2048kB/
   nr_hugepages
   ```

5. Run the Open vSwitch service:

   ```
   systemctl start openvswitch
   ```

6. Enable hardware offload (disabled by default):

   ```
   ovs-vsctl set Open_vSwitch . other_config:dpdk-extra="-a 0000:00:00.0"
   ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true
   ovs-vsctl --no-wait set Open_vSwitch . other_config:doca-init=true
   ovs-vsctl set Open_vSwitch . other_config:hw-offload=true
   ```

7. Restart the Open vSwitch service. This step is required for HW offload changes to take effect.

   ```
   systemctl restart openvswitch
   ```

8. Create OVS-DPDK bridge:

```
ovs-vsctl --no-wait add-br br0-ovs -- set bridge br0-ovs datapath_type=netdev
```

9. Add PF to OVS:

```
ovs-vsctl add-port br0-ovs pf -- set Interface pf type=dpdk options:dpdk-
devargs=0000:88:00.0,dv_flow_en=2,dv_xmeta_en=4
```

10. Add representor to OVS:

```
ovs-vsctl add-port br0-ovs representor -- set Interface representor
 type=dpdk options:dpdk-devargs=0000:88:00.0,representor=[<vf-
number>],dv_flow_en=2,dv_xmeta_en=4
```

Note: Note that `<vf-number>` must be replaced by the number of the VF.

2.2. Offloading VXLAN Encapsulation/ Decapsulation Actions

vSwitch in userspace rather than kernel-based Open vSwitch requires an additional bridge. The purpose of this bridge is to allow use of the kernel network stack for routing and ARP resolution.

The datapath must look up the routing table and ARP table to prepare the tunnel header and transmit data to the output port.

VXLAN encapsulation/decapsulation offload configuration is done with:

- PF on `0000:03:00.0` PCIe and MAC `98:03:9b:cc:21:e8`
- Local IP `56.56.67.1` – the `br-phy` interface is configured to this IP
- Remote IP `56.56.68.1`

To configure OVS DOCA VXLAN:

1. Create a `br-phy` bridge:

```
ovs-vsctl add-br br-phy -- set Bridge br-phy datapath_type=netdev -- br-set-
external-id br-phy bridge-id br-phy -- set bridge br-phy fail-mode=standalone
 other_config:hwaddr=98:03:9b:cc:21:e8
```

2. Attach PF interface to `br-phy` bridge:

```
ovs-vsctl add-port br-phy p0 -- set Interface p0 type=dpdk options:dpdk-
devargs=0000:03:00.0,dv_flow_en=2,dv_xmeta_en=4
```

3. Configure IP to the bridge:

```
ip addr add 56.56.67.1/24 dev br-phy
```

4. Create a `br-ovs` bridge:

```
ovs-vsctl add-br br-ovs -- set Bridge br-ovs datapath_type=netdev -- br-set-
external-id br-ovs bridge-id br-ovs -- set bridge br-ovs fail-mode=standalone
```

5. Attach representor to `br-ovs`:

```
ovs-vsctl add-port br-ovs pf0vf0 -- set Interface pf0vf0 type=dpdk options:dpdk-
devargs=0000:03:00.0,representor=[0],dv_flow_en=2,dv_xmeta_en=4
```

6. Add a port for the VXLAN tunnel:

```
ovs-vsctl add-port ovs-sriov vxlan0 -- set interface vxlan0 type=vxlan
 options:local_ip=56.56.67.1 options:remote_ip=56.56.68.1 options:key=45
 options:dst_port=4789
```

2.3. Offloading Connection Tracking

Connection tracking enables stateful packet processing by keeping a record of currently open connections.

OVS flows utilizing connection tracking can be accelerated using advanced NICs by offloading established connections.

To view offloaded connections, run:

```
ovs-appctl dpctl/offload-stats-show
```

2.4. SR-IOV VF LAG

To configure OVS-DOCA SR-IOV VF LAG:

1. Enable SR-IOV on the NICs:
   ```
   mlxconfig -d <PCI> set SRIOV_EN=1
   ```
2. Allocate the desired number of VFs per port:
   ```
   echo $n > /sys/class/net/<net name>/device/sriov_numvfs
   ```
3. Unbind all VFs:
   ```
   echo <VF PCI> >/sys/bus/pci/drivers/mlx5_core/unbind
   ```
4. Change both NICs' mode to SwitchDev:
   ```
   devlink dev eswitch set pci/<PCI> mode switchdev
   ```
5. Create Linux bonding using kernel modules:
   ```
   modprobe bonding mode=<desired mode>
   ```

   Note: Other bonding parameters can be added here. The supported bond modes are Active-Backup, XOR, and LACP.
6. Bring all PFs and VFs down:
   ```
   ip link set <PF/VF> down
   ```
7. Attach both PFs to the bond:
   ```
   ip link set <PF> master bond0
   ```
8. Bring PFs and bond link up:
   ```
   ip link set <PF0> up
   ip link set <PF1> up
   ip link set bond0 up
   ```
9. To work with VF-LAG with OVS-DPDK, add the bond master (PF) to the bridge:
   ```
   ovs-vsctl add-port br-phy p0 -- set Interface p0 type=dpdk options:dpdk-
   devargs=0000:03:00.0,dv_flow_en=2,dv_xmeta_en=4  options:dpdk-lsc-interrupt=true
   ```
10. Add representor `$N` of PF0 or PF1 to a bridge:
    ```
    ovs-vsctl add-port br-phy rep$N -- set Interface rep$N type=dpdk options:dpdk-
    devargs=<PF0-PCI>,representor=pf0vf$N,dv_flow_en=2,dv_xmeta_en=4
    ```
    Or:
    ```
    ovs-vsctl add-port br-phy rep$N -- set Interface rep$N type=dpdk options:dpdk-
    devargs=<PF0-PCI>,representor=pf1vf$N,dv_flow_en=2,dv_xmeta_en=4
    ```

2.5. Offloading Geneve Encapsulation/ Decapsulation

Geneve tunneling offload support includes matching on extension header.

To configure OVS-DPDK Geneve encapsulation/decapsulation:

1. Create a `br-phy` bridge:

```
ovs-vsctl --may-exist add-br br-phy -- set Bridge br-phy datapath_type=netdev
 -- br-set-external-id br-phy bridge-id br-phy -- set bridge br-phy fail-
mode=standalone
```

2. Attach PF interface to `br-phy` bridge:

```
ovs-vsctl add-port br-phy pf -- set Interface pf type=dpdk options:dpdk-
devargs=<PF PCI>,dv_flow_en=2,dv_xmeta_en=4
```

3. Configure IP to the bridge:

```
ifconfig br-phy <$local_ip_1> up
```

4. Create a `br-int` bridge:

```
ovs-vsctl --may-exist add-br br-int -- set Bridge br-int datapath_type=netdev
 -- br-set-external-id br-int bridge-id br-int -- set bridge br-int fail-
mode=standalone
```

5. Attach representor to `br-int`:

```
ovs-vsctl add-port br-int rep$x -- set Interface rep$x type=dpdk options:dpdk-
devargs=<PF PCI>,representor=[$x],dv_flow_en=2,dv_xmeta_en=4
```

6. Add a port for the Geneve tunnel:

```
ovs-vsctl add-port br-int geneve0 -- set interface geneve0 type=geneve
 options:key=<VNI> options:remote_ip=<$remote_ip_1> options:local_ip=<
$local_ip_1>
```

Chapter 3. Known Limitations

- Only one insertion thread is supported (n-offload-threads=1).
- Only a single PF is currently supported.
- VF LAG in LACP mode is not supported.
- Geneve options are not supported.
- Only 250K connection are offloaded by CT offload.
- Only 8 CT zones are supported by CT offload.
- OVS restart on non-tunnel configuration is not supported. Remove all ports before restarting.

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com