NVIDIA DOCA OVS DOCA

User Guide

MLNX-15-060597 _v2.0.2 | May 2023

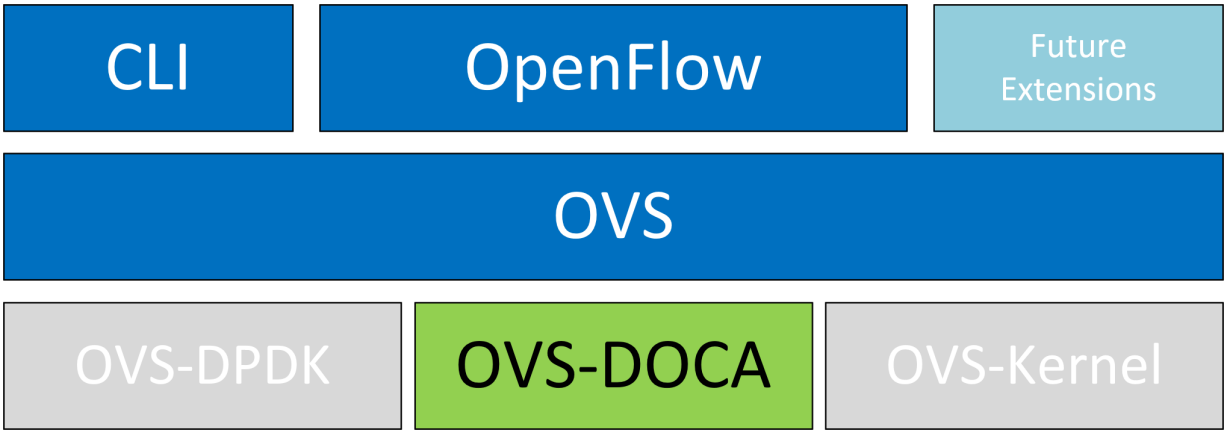
Table of Contents

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.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation nor any of its direct or indirect subsidiaries and affiliates (collectively: “NVIDIA”) make no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assume no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA, the NVIDIA logo, and Mellanox are trademarks and/or registered trademarks of Mellanox Technologies Ltd. and/or NVIDIA Corporation in the U.S. and in other countries. The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a world¬wide basis. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2023 NVIDIA Corporation & affiliates. All rights reserved.

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