DPDK

DATA PLANE DEVELOPMENT KIT

vDPA Device Drivers

Release 20.08.0

Aug 08, 2020

1 Overview of vDPA Drivers Features **2**

2 References **4**

3 Features Table **5**

4 IFCVF vDPA driver **7**
4.1 Pre-Installation Configuration 7
4.2 IFCVF vDPA Implementation 7
4.3 Features 8
4.4 Prerequisites 8
4.5 Limitations 8

5 MLX5 vDPA driver **9**
5.1 Design 9
5.2 Supported NICs 9
5.3 Prerequisites 10

The following are a list of vDPA (vHost Data Path Acceleration) device drivers, which can be used from an application through vhost API.

CHAPTER

ONE

OVERVIEW OF VDPA DRIVERS FEATURES

This section explains the supported features that are listed in the table below.

csum Device can handle packets with partial checksum.

guest csum Guest can handle packets with partial checksum.

mac Device has given MAC address.

gso Device can handle packets with any GSO type.

guest tso4 Guest can receive TSOv4.

guest tso6 Guest can receive TSOv6.

ecn Device can receive TSO with ECN.

ufo Device can receive UFO.

host tso4 Device can receive TSOv4.

host tso6 Device can receive TSOv6.

mrg rxbuf Guest can merge receive buffers.

ctrl vq Control channel is available.

ctrl rx Control channel RX mode support.

any layout Device can handle any descriptor layout.

guest announce Guest can send gratuitous packets.

mq Device supports Receive Flow Steering.

version 1 v1.0 compliant.

log all Device can log all write descriptors (live migration).

indirect desc Indirect buffer descriptors support.

event idx Support for avail_idx and used_idx fields.

mtu Host can advise the guest with its maximum supported MTU.

in_order Device can use descriptors in ring order.

IOMMU platform Device support IOMMU addresses.

packed Device support packed virtio queues.

proto mq Support the number of queues query.

proto log shmfd Guest support setting log base.

proto rarp Host can broadcast a fake RARP after live migration.

proto reply ack Host support requested operation status ack.

proto host notifier Host can register memory region based host notifiers.

proto pagefault Slave expose page-fault FD for migration process.

queue statistics Support virtio queue statistics query.

BSD nic_uio BSD `nic_uio` module supported.

Linux VFIO Works with `vfio-pci` kernel module.

Other kdrv Kernel module other than above ones supported.

ARMv7 Support armv7 architecture.

ARMv8 Support armv8a (64bit) architecture.

Power8 Support PowerPC architecture.

x86-32 Support 32bits x86 architecture.

x86-64 Support 64bits x86 architecture.

Usage doc Documentation describes usage, In `doc/guides/vdpadevs/`.

Design doc Documentation describes design. In `doc/guides/vdpadevs/`.

Perf doc Documentation describes performance values, In `doc/perf/`.

Note: Most of the features capabilities should be provided by the drivers via the next vDPA operations: `get_features` and `get_protocol_features`.

CHAPTER

TWO

REFERENCES

- OASIS: Virtual I/O Device (VIRTIO) Version 1.1
- QEMU: Vhost-user Protocol

CHAPTER

THREE

FEATURES TABLE

Table 3.1: Features availability in vDPA drivers

Feature	i f c v f	m l x 5
csum		Y
guest csum		Y
mac		
gso		
guest tso4		
guest tso6		
ecn		
ufo		
host tso4		Y
host tso6		Y
mrg rxbuf		
ctrl vq		
ctrl rx		
any layout		Y
guest announce		Y
mq		Y
version 1		Y
log all		Y
indirect desc		
event idx		
mtu		Y
in_order		
IOMMU platform		
packed		Y
proto mq		Y
proto log shmfd		Y
proto rarp		
proto reply ack		
proto host notifier		Y
proto pagefault		
queue statistics		Y
BSD nic_uio		
Linux VFIO		
Other kdrv		Y

Continued on next page

Table 3.1 – continued from previous page

Feature	i f c v f	m l x 5
ARMv7		
ARMv8		Y
Power8		Y
x86-32	Y	Y
x86-64	Y	Y
Usage doc		Y
Design doc		Y
Perf doc		

Note: Features marked with "P" are partially supported. Refer to the appropriate driver guide in the following sections for details.

CHAPTER

FOUR

IFCVF VDPA DRIVER

The IFCVF vDPA (vhost data path acceleration) driver provides support for the Intel FPGA 100G VF (IFCVF). IFCVF's datapath is virtio ring compatible, it works as a HW vhost backend which can send/receive packets to/from virtio directly by DMA. Besides, it supports dirty page logging and device state report/restore, this driver enables its vDPA functionality.

4.1 Pre-Installation Configuration

4.1.1 Config File Options

The following option can be modified in the `config` file.

- `CONFIG_RTE_LIBRTE_IFC_PMD` (default `y` for linux)

  Toggle compilation of the `librte_pmd_ifc` driver.

4.2 IFCVF vDPA Implementation

IFCVF's vendor ID and device ID are same as that of virtio net pci device, with its specific subsystem vendor ID and device ID. To let the device be probed by IFCVF driver, adding "vdpa=1" parameter helps to specify that this device is to be used in vDPA mode, rather than polling mode, virtio pmd will skip when it detects this message. If no this parameter specified, device will not be used as a vDPA device, and it will be driven by virtio pmd.

Different VF devices serve different virtio frontends which are in different VMs, so each VF needs to have its own DMA address translation service. During the driver probe a new container is created for this device, with this container vDPA driver can program DMA remapping table with the VM's memory region information.

The device argument "sw-live-migration=1" will configure the driver into SW assisted live migration mode. In this mode, the driver will set up a SW relay thread when LM happens, this thread will help device to log dirty pages. Thus this mode does not require HW to implement a dirty page logging function block, but will consume some percentage of CPU resource depending on the network throughput. If no this parameter specified, driver will rely on device's logging capability.

4.2.1 Key IFCVF vDPA driver ops

- ifcvf_dev_config: Enable VF data path with virtio information provided by vhost lib, including IOMMU programming to enable VF DMA to VM's memory, VFIO interrupt setup to route HW

interrupt to virtio driver, create notify relay thread to translate virtio driver's kick to a MMIO write onto HW, HW queues configuration.

This function gets called to set up HW data path backend when virtio driver in VM gets ready.

- ifcvf_dev_close: Revoke all the setup in ifcvf_dev_config.

  This function gets called when virtio driver stops device in VM.

4.2.2 To create a vhost port with IFC VF

- Create a vhost socket and assign a VF's device ID to this socket via vhost API. When QEMU vhost connection gets ready, the assigned VF will get configured automatically.

4.3 Features

Features of the IFCVF driver are:

- Compatibility with virtio 0.95 and 1.0.
- SW assisted vDPA live migration.

4.4 Prerequisites

- Platform with IOMMU feature. IFC VF needs address translation service to Rx/Tx directly with virtio driver in VM.

4.5 Limitations

4.5.1 Dependency on vfio-pci

vDPA driver needs to setup VF MSIX interrupts, each queue's interrupt vector is mapped to a callfd associated with a virtio ring. Currently only vfio-pci allows multiple interrupts, so the IFCVF driver is dependent on vfio-pci.

4.5.2 Live Migration with VIRTIO_NET_F_GUEST_ANNOUNCE

IFC VF doesn't support RARP packet generation, virtio frontend supporting VIRTIO_NET_F_GUEST_ANNOUNCE feature can help to do that.

CHAPTER

FIVE

MLX5 VDPA DRIVER

The MLX5 vDPA (vhost data path acceleration) driver library (**librte_pmd_mlx5_vdpa**) provides support for **Mellanox ConnectX-6**, **Mellanox ConnectX-6 Dx** and **Mellanox BlueField** families of 10/25/40/50/100/200 Gb/s adapters as well as their virtual functions (VF) in SR-IOV context.

Note: Due to external dependencies, this driver is disabled in default configuration of the "make" build. It can be enabled with `CONFIG_RTE_LIBRTE_MLX5_VDPA_PMD=y` or by using "meson" build system which will detect dependencies.

5.1 Design

For security reasons and robustness, this driver only deals with virtual memory addresses. The way resources allocations are handled by the kernel, combined with hardware specifications that allow to handle virtual memory addresses directly, ensure that DPDK applications cannot access random physical memory (or memory that does not belong to the current process).

The PMD can use libibverbs and libmlx5 to access the device firmware or directly the hardware components. There are different levels of objects and bypassing abilities to get the best performances:

- Verbs is a complete high-level generic API
- Direct Verbs is a device-specific API
- DevX allows to access firmware objects
- Direct Rules manages flow steering at low-level hardware layer

Enabling librte_pmd_mlx5_vdpa causes DPDK applications to be linked against libibverbs.

A Mellanox mlx5 PCI device can be probed by either net/mlx5 driver or vdpa/mlx5 driver but not in parallel. Hence, the user should decide the driver by the `class` parameter in the device argument list. By default, the mlx5 device will be probed by the net/mlx5 driver.

5.2 Supported NICs

- Mellanox® ConnectX®-6 200G MCX654106A-HCAT (2x200G)
- Mellanox® ConnectX®-6 Dx EN 25G MCX621102AN-ADAT (2x25G)
- Mellanox® ConnectX®-6 Dx EN 100G MCX623106AN-CDAT (2x100G)

- Mellanox® ConnectX®-6 Dx EN 200G MCX623105AN-VDAT (1x200G)
- Mellanox® BlueField SmartNIC 25G MBF1M332A-ASCAT (2x25G)

5.3 Prerequisites

- Mellanox OFED version: **5.0** see ../../nics/mlx5 guide for more Mellanox OFED details.

5.3.1 Compilation options

These options can be modified in the `.config` file.

- `CONFIG_RTE_LIBRTE_MLX5_VDPA_PMD` (default **n**)

  Toggle compilation of librte_pmd_mlx5 itself.

- `CONFIG_RTE_IBVERBS_LINK_DLOPEN` (default **n**)

  Build PMD with additional code to make it loadable without hard dependencies on **libibverbs** nor **libmlx5**, which may not be installed on the target system.

  In this mode, their presence is still required for it to run properly, however their absence won't prevent a DPDK application from starting (with `CONFIG_RTE_BUILD_SHARED_LIB` disabled) and they won't show up as missing with `ldd(1)`.

  It works by moving these dependencies to a purpose-built rdma-core "glue" plug-in which must either be installed in a directory whose name is based on `CONFIG_RTE_EAL_PMD_PATH` suffixed with `-glue` if set, or in a standard location for the dynamic linker (e.g. `/lib`) if left to the default empty string (`""`).

  This option has no performance impact.

- `CONFIG_RTE_IBVERBS_LINK_STATIC` (default **n**)

  Embed static flavor of the dependencies **libibverbs** and **libmlx5** in the PMD shared library or the executable static binary.

Note: For BlueField, target should be set to `arm64-bluefield-linux-gcc`. This will enable `CONFIG_RTE_LIBRTE_MLX5_VDPA_PMD` and set `RTE_CACHE_LINE_SIZE` to 64. Default armv8a configuration of make build and meson build set it to 128 then brings performance degradation.

5.3.2 Run-time configuration

- **ethtool** operations on related kernel interfaces also affect the PMD.

Driver options

- `class` parameter [string]

  Select the class of the driver that should probe the device. *vdpa* for the mlx5 vDPA driver.

- `event_mode` parameter [int]

  - 0, Completion queue scheduling will be managed by a timer thread which automatically adjusts its delays to the coming traffic rate.
  - 1, Completion queue scheduling will be managed by a timer thread with fixed delay time.
  - 2, Completion queue scheduling will be managed by interrupts. Each CQ burst arms the CQ in order to get an interrupt event in the next traffic burst.
  - Default mode is 0.

- `event_us` parameter [int]

  Per mode micro-seconds parameter - relevant only for event mode 0 and 1:

  - 0, A nonzero value to set timer step in micro-seconds. The timer thread dynamic delay change steps according to this value. Default value is 1us.
  - 1, A nonzero value to set fixed timer delay in micro-seconds. Default value is 100us.

- `no_traffic_time` parameter [int]

  A nonzero value defines the traffic off time, in seconds, that moves the driver to no-traffic mode. In this mode the timer events are stopped and interrupts are configured to the device in order to notify traffic for the driver. Default value is 2s.