DPDK

DATA PLANE DEVELOPMENT KIT

DPDK Tools User Guides

Release 17.02.1

June 02, 2017

CONTENTS

1 dpdk-procinfo Application **1**
1.1 Running the Application 1

2 dpdk-pdump Application **2**
2.1 Running the Application 2
2.2 Example 3

3 dpdk-pmdinfo Application **4**
3.1 Running the Application 4

4 dpdk-devbind Application **5**
4.1 Running the Application 5
4.2 OPTIONS 5
4.3 Examples 6

5 dpdk-test-crypto-perf Application **7**
5.1 Limitations 7
5.2 Compiling the Application 7
5.3 Running the Application 8
5.4 Examples 12

CHAPTER

ONE

DPDK-PROCINFO APPLICATION

The dpdk-procinfo application is a Data Plane Development Kit (DPDK) application that runs as a DPDK secondary process and is capable of retrieving port statistics, resetting port statistics and printing DPDK memory information. This application extends the original functionality that was supported by dump_cfg.

1.1 Running the Application

The application has a number of command line options:

```
./$(RTE_TARGET)/app/dpdk-procinfo -- -m | [-p PORTMASK] [--stats | --xstats |
--stats-reset | --xstats-reset]
```

1.1.1 Parameters

-p PORTMASK: Hexadecimal bitmask of ports to configure.

–stats The stats parameter controls the printing of generic port statistics. If no port mask is specified stats are printed for all DPDK ports.

–xstats The xstats parameter controls the printing of extended port statistics. If no port mask is specified xstats are printed for all DPDK ports.

–stats-reset The stats-reset parameter controls the resetting of generic port statistics. If no port mask is specified, the generic stats are reset for all DPDK ports.

–xstats-reset The xstats-reset parameter controls the resetting of extended port statistics. If no port mask is specified xstats are reset for all DPDK ports.

-m: Print DPDK memory information.

CHAPTER

TWO

DPDK-PDUMP APPLICATION

The `dpdk-pdump` tool is a Data Plane Development Kit (DPDK) tool that runs as a DPDK secondary process and is capable of enabling packet capture on dpdk ports.

Note:

- The `dpdk-pdump` tool can only be used in conjunction with a primary application which has the packet capture framework initialized already.
- The `dpdk-pdump` tool depends on libpcap based PMD which is disabled by default in the build configuration files, owing to an external dependency on the libpcap development files which must be installed on the board. Once the libpcap development files are installed, the libpcap based PMD can be enabled by setting CONFIG_RTE_LIBRTE_PMD_PCAP=y and recompiling the DPDK.

2.1 Running the Application

The tool has a number of command line options:

```
./build/app/dpdk-pdump --
                       --pdump '(port=<port id> | device_id=<pci id or vdev name>),
                                (queue=<queue_id>),
                                (rx-dev=<iface or pcap file> |
                                 tx-dev=<iface or pcap file>),
                                [ring-size=<ring size>],
                                [mbuf-size=<mbuf data size>],
                                [total-num-mbufs=<number of mbufs>]'
                       [--server-socket-path=<server socket dir>]
                       [--client-socket-path=<client socket dir>]
```

The `--pdump` command line option is mandatory and it takes various sub arguments which are described in below section.

Note:

- Parameters inside the parentheses represents mandatory parameters.
- Parameters inside the square brackets represents optional parameters.
- Multiple instances of `--pdump` can be passed to capture packets on different port and queue combinations.

The `--server-socket-path` command line option is optional. This represents the server socket directory. If no value is passed default values are used i.e. `/var/run/.dpdk/` for root users and `~/.dpdk/` for non root users.

The `--client-socket-path` command line option is optional. This represents the client socket directory. If no value is passed default values are used i.e. `/var/run/.dpdk/` for root users and `~/.dpdk/` for non root users.

2.1.1 The `--pdump` parameters

`port`: Port id of the eth device on which packets should be captured.

`device_id`: PCI address (or) name of the eth device on which packets should be captured.

Note:

- As of now the `dpdk-pdump` tool cannot capture the packets of virtual devices in the primary process due to a bug in the ethdev library. Due to this bug, in a multi process context, when the primary and secondary have different ports set, then the secondary process (here the `dpdk-pdump` tool) overwrites the `rte_eth_devices[]` entries of the primary process.

`queue`: Queue id of the eth device on which packets should be captured. The user can pass a queue value of `*` to enable packet capture on all queues of the eth device.

`rx-dev`: Can be either a pcap file name or any Linux iface.

`tx-dev`: Can be either a pcap file name or any Linux iface.

Note:

- To receive ingress packets only, `rx-dev` should be passed.
- To receive egress packets only, `tx-dev` should be passed.
- To receive ingress and egress packets separately `rx-dev` and `tx-dev` should both be passed with the different file names or the Linux iface names.
- To receive ingress and egress packets together, `rx-dev` and `tx-dev` should both be passed with the same file name or the same Linux iface name.

`ring-size`: Size of the ring. This value is used internally for ring creation. The ring will be used to enqueue the packets from the primary application to the secondary. This is an optional parameter with default size 16384.

`mbuf-size`: Size of the mbuf data. This is used internally for mempool creation. Ideally this value must be same as the primary application's mempool's mbuf data size which is used for packet RX. This is an optional parameter with default size 2176.

`total-num-mbufs`: Total number mbufs in mempool. This is used internally for mempool creation. This is an optional parameter with default value 65535.

2.2 Example

```
$ sudo ./build/app/dpdk-pdump -- --pdump 'port=0,queue=*,rx-dev=/tmp/rx.pcap'
```

CHAPTER

THREE

DPDK-PMDINFO APPLICATION

The `dpdk-pmdinfo` tool is a Data Plane Development Kit (DPDK) utility that can dump a PMDs hardware support info.

3.1 Running the Application

The tool has a number of command line options:

```
dpdk-pmdinfo [-hrtp] [-d <pci id file] <elf-file>

-h, --help            Show a short help message and exit
-r, --raw             Dump as raw json strings
-d FILE, --pcidb=FILE Specify a pci database to get vendor names from
-t, --table           Output information on hw support as a hex table
-p, --plugindir       Scan dpdk for autoload plugins
```

Note:

- Parameters inside the square brackets represents optional parameters.

CHAPTER

FOUR

DPDK-DEVBIND APPLICATION

The `dpdk-devbind` tool is a Data Plane Development Kit (DPDK) utility that helps binding and unbinding devices from specific drivers. As well as checking their status in that regard.

4.1 Running the Application

The tool has a number of command line options:

```
dpdk-devbind [options] DEVICE1 DEVICE2 ....
```

4.2 OPTIONS

- `--help, --usage`

  Display usage information and quit

- `-s, --status`

  Print the current status of all known network interfaces. For each device, it displays the PCI domain, bus, slot and function, along with a text description of the device. Depending upon whether the device is being used by a kernel driver, the `igb_uio` driver, or no driver, other relevant information will be displayed: - the Linux interface name e.g. `if=eth0` - the driver being used e.g. `drv=igb_uio` - any suitable drivers not currently using that device e.g. `unused=igb_uio` NOTE: if this flag is passed along with a bind/unbind option, the status display will always occur after the other operations have taken place.

- `-b driver, --bind=driver`

  Select the driver to use or “none” to unbind the device

- `-u, --unbind`

  Unbind a device (Equivalent to `-b none`)

- `--force`

  By default, devices which are used by Linux - as indicated by having routes in the routing table - cannot be modified. Using the `--force` flag overrides this behavior, allowing active links to be forcibly unbound. WARNING: This can lead to loss of network connection and should be used with caution.

Warning: Due to the way VFIO works, there are certain limitations to which devices can be used with VFIO. Mainly it comes down to how IOMMU groups work. Any Virtual Function device can be used with VFIO on its own, but physical devices will require either all ports bound to VFIO, or some of them bound to VFIO while others not being bound to anything at all.
If your device is behind a PCI-to-PCI bridge, the bridge will then be part of the IOMMU group in which your device is in. Therefore, the bridge driver should also be unbound from the bridge PCI device for VFIO to work with devices behind the bridge.

Warning: While any user can run the `dpdk-devbind.py` script to view the status of the network ports, binding or unbinding network ports requires root privileges.

4.3 Examples

To display current device status:

```
dpdk-devbind --status
```

To bind eth1 from the current driver and move to use igb_uio:

```
dpdk-devbind --bind=igb_uio eth1
```

To unbind 0000:01:00.0 from using any driver:

```
dpdk-devbind -u 0000:01:00.0
```

To bind 0000:02:00.0 and 0000:02:00.1 to the ixgbe kernel driver:

```
dpdk-devbind -b ixgbe 02:00.0 02:00.1
```

To check status of all network ports, assign one to the igb_uio driver and check status again:

```
# Check the status of the available devices.
dpdk-devbind --status
Network devices using DPDK-compatible driver
============================================
<none>

Network devices using kernel driver
===================================
0000:0a:00.0 '82599ES 10-Gigabit' if=eth2 drv=ixgbe unused=


# Bind the device to igb_uio.
sudo dpdk-devbind -b igb_uio 0000:0a:00.0


# Recheck the status of the devices.
dpdk-devbind --status
Network devices using DPDK-compatible driver
============================================
0000:0a:00.0 '82599ES 10-Gigabit' drv=igb_uio unused=
```

CHAPTER

FIVE

DPDK-TEST-CRYPTO-PERF APPLICATION

The `dpdk-test-crypto-perf` tool is a Data Plane Development Kit (DPDK) utility that allows measuring performance parameters of PMDs available in the crypto tree. There are available two measurement types: throughput and latency. User can use multiply cores to run tests on but only one type of crypto PMD can be measured during single application execution. Cipher parameters, type of device, type of operation and chain mode have to be specified in the command line as application parameters. These parameters are checked using device capabilities structure.

5.1 Limitations

On hardware devices the cycle-count doesn't always represent the actual offload cost. The cycle-count only represents the offload cost when the hardware accelerator is not fully loaded, when loaded the cpu cycles freed up by the offload are still consumed by the test tool and included in the cycle-count. These cycles are consumed by retries and inefficient API calls enqueuing and dequeuing smaller bursts than specified by the cmdline parameter. This results in a larger cycle-count measurement and should not be interpreted as an offload cost measurement.

On hardware devices the throughput measurement is not necessarily the maximum possible for the device, e.g. it may be necessary to use multiple cores to keep the hardware accelerator fully loaded and so measure maximum throughput.

5.2 Compiling the Application

Step 1: PMD setting

The `dpdk-test-crypto-perf` tool depends on crypto device drivers PMD which are disabled by default in the build configuration file `common_base`. The crypto device drivers PMD which should be tested can be enabled by setting:

```
CONFIG_RTE_LIBRTE_PMD_<name>=y
```

Setting example for open ssl PMD:

```
CONFIG_RTE_LIBRTE_PMD_OPENSSL=y
```

Step 2: Linearization setting

It is possible linearized input segmented packets just before crypto operation for devices which doesn't support scatter-gather, and allows to measure performance also for this use case.

To set on the linearization options add below definition to the `cperf_ops.h` file:

```
#define CPERF_LINEARIZATION_ENABLE
```

Step 3: Build the application

Execute the `dpdk-setup.sh` script to build the DPDK library together with the `dpdk-test-crypto-perf` applcation.

Initially, the user must select a DPDK target to choose the correct target type and compiler options to use when building the libraries. The user must have all libraries, modules, updates and compilers installed in the system prior to this, as described in the earlier chapters in this Getting Started Guide.

5.3 Running the Application

The tool application has a number of command line options:

```
dpdk-test-crypto-perf [EAL Options] -- [Application Options]
```

5.3.1 EAL Options

The following are the EAL command-line options that can be used in conjunction with the `dpdk-test-crypto-perf` applcation. See the DPDK Getting Started Guides for more information on these options.

- `-c <COREMASK>`

  Set the hexadecimal bitmask of the cores to run on.

- `-w <PCI>`

  Add a PCI device in white list.

- `--vdev <driver><id>`

  Add a virtual device.

5.3.2 Appication Options

The following are the appication command-line options:

- `--ptest type`

  Set test type, where `type` is one of the following:

  ```
  throughput
  latency
  ```

- `--silent`

  Disable options dump.

- `--pool-sz <n>`

  Set the number of mbufs to be allocated in the mbuf pool.

- `--total-ops <n>`

Set the number of total operations performed.

- `--burst-sz <n>`

  Set the number of packets per burst.

- `--buffer-sz <n>`

  Set the size of single packet (plaintext or ciphertext in it).

- `--segments-nb <n>`

  Set the number of segments per packet.

- `--devtype <name>`

  Set device type, where `name` is one of the following:

  ```
  crypto_null
  crypto_aesni_mb
  crypto_aesni_gcm
  crypto_openssl
  crypto_qat
  crypto_snow3g
  crypto_kasumi
  crypto_zuc
  ```

- `--optype <name>`

  Set operation type, where `name` is one of the following:

  ```
  cipher-only
  auth-only
  cipher-then-auth
  auth-then-cipher
  aead
  ```

- `--sessionless`

  Enable session-less crypto operations mode.

- `--out-of-place`

  Enable out-of-place crypto operations mode.

- `--verify`

  Enable verify that all crypto operations were successful. The verification is done after the performance test.

- `--test-file <name>`

  Set test vector file path. See the Test Vector File chapter.

- `--test-name <name>`

  Set specific test name section in the test vector file.

- `--cipher-algo <name>`

  Set cipher algorithm name, where `name` is one of the following:

  ```
  3des-cbc
  3des-ecb
  3des-ctr
  aes-cbc
  aes-ccm
  ```

```
aes-ctr
aes-ecb
aes-gcm
aes-f8
aes-xts
arc4
null
kasumi-f8
snow3g-uea2
zuc-eea3
```

- `--cipher-op <mode>`

  Set cipher operation mode, where `mode` is one of the following:

  ```
  encrypt
  decrypt
  ```

- `--cipher-key-sz <n>`

  Set the size of cipher key.

- `--cipher-iv-sz <n>`

  Set the size of cipher iv.

- `--auth-algo <name>`

  Set authentication algorithm name, where `name` is one of the following:

  ```
  3des-cbc
  aes-cbc-mac
  aes-ccm
  aes-cmac
  aes-gcm
  aes-gmac
  aes-xcbc-mac
  md5
  md5-hmac
  sha1
  sha1-hmac
  sha2-224
  sha2-224-hmac
  sha2-256
  sha2-256-hmac
  sha2-384
  sha2-384-hmac
  sha2-512
  sha2-512-hmac
  kasumi-f9
  snow3g-uia2
  zuc-eia3
  ```

- `--auth-op <mode>`

  Set authentication operation mode, where `mode` is one of the following:

  ```
  verify
  generate
  ```

- `--auth-key-sz <n>`

  Set the size of authentication key.

- `--auth-digest-sz <n>`

Set the size of authentication digest.

- `--auth-aad-sz <n>`

  Set the size of authentication aad.

- `--csv-friendly`

  Enable test result output CSV friendly rather than human friendly.

5.3.3 Test Vector File

The test vector file is a text file contain information about test vectors. The file is made of the sections. The first section doesn't have header. It contain global information used in each test variant vectors - typicaly information about plaintext, ciphertext, cipher key, aut key, initial vector. All other sections begin header. The sections contain particular information typicaly digest.

Format of the file:

Each line beginig with sign '#' contain comment and it is ignored by parser:

```
# <comment>
```

Header line is just name in square bracket:

```
[<section name>]
```

Data line contain information tocken then sign '=' and a string of bytes in C byte array format:

```
<tocken> = <C byte array>
```

Tockens list:

- `plaintext`

  Original plaintext to be crypted.

- `ciphertext`

  Encrypted plaintext string.

- `cipher_key`

  Key used in cipher operation.

- `auth_key`

  Key used in auth operation.

- `iv`

  Initial vector.

- `aad`

  Additional data.

- `digest`

  Digest string.

5.4 Examples

Call application for performance throughput test of single Aesni MB PMD for cipher encryption aes-cbc and auth generation sha1-hmac, one milion operations, burst size 32, packet size 64:

```
dpdk-test-crypto-perf -c 0xc0 --vdev crypto_aesni_mb_pmd -w 0000:00:00.0 --
--ptest throughput --devtype crypto_aesni_mb --optype cipher-then-auth
--cipher-algo aes-cbc --cipher-op encrypt --cipher-key-sz 16 --auth-algo
sha1-hmac --auth-op generate --auth-key-sz 64 --auth-digest-sz 12
--total-ops 10000000 --burst-sz 32 --buffer-sz 64
```

Call application for performance latency test of two Aesni MB PMD executed on two cores for cipher encryption aes-cbc, ten operations in silent mode:

```
dpdk-test-crypto-perf -c 0xf0 --vdev crypto_aesni_mb_pmd1
--vdev crypto_aesni_mb_pmd2 -w 0000:00:00.0 -- --devtype crypto_aesni_mb
--cipher-algo aes-cbc --cipher-key-sz 16 --cipher-iv-sz 16
--cipher-op encrypt --optype cipher-only --silent
--ptest latency --total-ops 10
```

Call application for performance latency test of single open ssl PMD for cipher encryption aes-gcm and auth generation aes-gcm,ten operations in silent mode, test vector provide in file "test_aes_gcm.data" with packet verification:

```
dpdk-test-crypto-perf -c 0xf0 --vdev crypto_openssl -w 0000:00:00.0 --
--devtype crypto_openssl --cipher-algo aes-gcm --cipher-key-sz 16
--cipher-iv-sz 16 --cipher-op encrypt --auth-algo aes-gcm --auth-key-sz 16
--auth-digest-sz 16 --auth-aad-sz 16 --auth-op generate --optype aead
--silent --ptest latency --total-ops 10
--test-file test_aes_gcm.data --verify
```

Test vector file for cipher algorithm aes cbc 256 with authorization sha:

```
# Global Section
plaintext =
0xff, 0xca, 0xfb, 0xf1, 0x38, 0x20, 0x2f, 0x7b, 0x24, 0x98, 0x26, 0x7d, 0x1d, 0x9f, 0xb3, 0x93,
0xd9, 0xef, 0xbd, 0xad, 0x4e, 0x40, 0xbd, 0x60, 0xe9, 0x48, 0x59, 0x90, 0x67, 0xd7, 0x2b, 0x7b,
0x8a, 0xe0, 0x4d, 0xb0, 0x70, 0x38, 0xcc, 0x48, 0x61, 0x7d, 0xee, 0xd6, 0x35, 0x49, 0xae, 0xb4,
0xaf, 0x6b, 0xdd, 0xe6, 0x21, 0xc0, 0x60, 0xce, 0x0a, 0xf4, 0x1c, 0x2e, 0x1c, 0x8d, 0xe8, 0x7b
ciphertext =
0x77, 0xF9, 0xF7, 0x7A, 0xA3, 0xCB, 0x68, 0x1A, 0x11, 0x70, 0xD8, 0x7A, 0xB6, 0xE2, 0x37, 0x7E,
0xD1, 0x57, 0x1C, 0x8E, 0x85, 0xD8, 0x08, 0xBF, 0x57, 0x1F, 0x21, 0x6C, 0xAD, 0xAD, 0x47, 0x1E,
0x0D, 0x6B, 0x79, 0x39, 0x15, 0x4E, 0x5B, 0x59, 0x2D, 0x76, 0x87, 0xA6, 0xD6, 0x47, 0x8F, 0x82,
0xB8, 0x51, 0x91, 0x32, 0x60, 0xCB, 0x97, 0xDE, 0xBE, 0xF0, 0xAD, 0xFC, 0x23, 0x2E, 0x22, 0x02
cipher_key =
0xE4, 0x23, 0x33, 0x8A, 0x35, 0x64, 0x61, 0xE2, 0x49, 0x03, 0xDD, 0xC6, 0xB8, 0xCA, 0x55, 0x7A,
0xd0, 0xe7, 0x4b, 0xfb, 0x5d, 0xe5, 0x0c, 0xe7, 0x6f, 0x21, 0xb5, 0x52, 0x2a, 0xbb, 0xc7, 0xf7
auth_key =
0xaf, 0x96, 0x42, 0xf1, 0x8c, 0x50, 0xdc, 0x67, 0x1a, 0x43, 0x47, 0x62, 0xc7, 0x04, 0xab, 0x05,
0xf5, 0x0c, 0xe7, 0xa2, 0xa6, 0x23, 0xd5, 0x3d, 0x95, 0xd8, 0xcd, 0x86, 0x79, 0xf5, 0x01, 0x47,
0x4f, 0xf9, 0x1d, 0x9d, 0x36, 0xf7, 0x68, 0x1a, 0x64, 0x44, 0x58, 0x5d, 0xe5, 0x81, 0x15, 0x2a,
0x41, 0xe4, 0x0e, 0xaa, 0x1f, 0x04, 0x21, 0xff, 0x2c, 0xf3, 0x73, 0x2b, 0x48, 0x1e, 0xd2, 0xf7
iv =
0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F
# Section sha 1 hmac buff 32
[sha1_hmac_buff_32]
digest =
0x36, 0xCA, 0x49, 0x6A, 0xE3, 0x54, 0xD8, 0x4F, 0x0B, 0x76, 0xD8, 0xAA, 0x78, 0xEB, 0x9D, 0x65,
0x2C, 0xCA, 0x1F, 0x97
# Section sha 256 hmac buff 32
[sha256_hmac_buff_32]
digest =
```

```
0x1C, 0xB2, 0x3D, 0xD1, 0xF9, 0xC7, 0x6C, 0x49, 0x2E, 0xDA, 0x94, 0x8B, 0xF1, 0xCF, 0x96, 0x43,
0x67, 0x50, 0x39, 0x76, 0xB5, 0xA1, 0xCE, 0xA1, 0xD7, 0x77, 0x10, 0x07, 0x43, 0x37, 0x05, 0xB4
```