Getting Started with DPDK and UHD

Contents

- 1 Application Note Number
- 2 Revision History
- 3 Overview
- 4 Abstract
- 5 Supported Devices
    - 5.1 USRPs
    - 5.2 Host Network Cards
- 6 References
- 7 Dependencies
- 8 Installing DPDK
- 9 Installing UHD
- 10 Enable hugepages
- 11 Preparing your UHD Configuration File
- 12 Additional Host Configuration for NIC Vendors
    - 12.1 Intel X520 / X710
    - 12.2 Mellanox NICs
    - 12.3 Running UHD Applications with DPDK
    - 12.4 Tuning Notes
        - 12.4.1 General Host Performance Tuning App Note
        - 12.4.2 Increasing num_recv_frames
        - 12.4.3 Full Rate Streaming
        - 12.4.4 Full Rate on X3xx
        - 12.4.5 Isolate CPUs
        - 12.4.6 Disable System Interrupts
        - 12.4.7 Disable Hyper-threading
    - 12.5 Additional Tuning Notes from Intel

AN-500

Date	Author	Details
2020-01-08	Nate Temple Alex Williams	Initial creation

This application note walks through the process to get started with the Data Plane Development Kit (DPDK) driver within UHD.

Up until now, UHD's only support for networked devices was backed by the kernel's sockets implementation. Every call to `send()` or `recv()` would cause a context switch and invite the kernel's scheduler to replace our thread with something else. Because the typical scheduler is optimized to distribute CPU time fairly across multiple loads, the timing-critical threads might sporadically be hit with sleeping time, and the thread might be migrated off its current CPU and forced to run on another. The overhead and random latency spikes make it difficult to enable reliable real-time streaming at higher rates.

DPDK is a high-speed packet processing framework that enables a kernel bypass for network drivers. By putting the entire driver in user space, avoiding context switches, and pinning I/O threads to cores, UHD and DPDK combine to largely prevent the latency spikes induced by the scheduler. In addition, the overall overhead for packet processing lowers.

DPDK is supported on the following USRP devices:

- N300 / N310
- N320 / N321
- X300 / X310
- E320

DPDK is supported on many Intel and Mellanox based 10Gb NICs. Below is a list of NICs Ettus Research has tested. For a full list of NICs supported by DPDK, please see the DPDK manual.

- Intel X520-DA1
- Intel X520-DA2
- Intel X710-DA2
- Intel X710-DA4
- Intel XL710
- Mellanox MCX4121A-ACAT ConnectX-4 Lx

- DPDK: https://www.dpdk.org/
- UHD Manual: https://files.ettus.com/manual/page_dpdk.html

- UHD 3.x requires DPDK 17.11, which is included in the default repos of Ubuntu 18.04.x
- DPDK support was added for the N3xx/E320 USRPs with UHD 3.13.x.x
- DPDK support was added for the X3xx with UHD 3.14.1.0

It is recommended to use the latest stable version of UHD, which at the time of this writing is UHD 3.15.0.0

On Ubuntu 18.04.x, it is possible to install DPDK 17.11 via `apt`:

```
sudo apt install dpdk dpdk-dev
```

Once the `dpdk` and `dpdk-dev` packages are installed, UHD will locate them during a build and you should see DPDK in the enabled components lists.

Edit your grub configuration file, `/etc/default/grub`, and add the follow parameters to `GRUB_CMDLINE_LINUX_DEFAULT`:

```
iommu=pt intel_iommu=on hugepages=2048
```

On a vanilla Ubuntu system it should look like this:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash iommu=pt intel_iommu=on hugepages=2048"
```

Close `/etc/default/grub` and at the command prompt, update your grub configuration with the command:

```
sudo update-grub
```

For these settings to take effect, reboot your host machine.

You should note the MAC addresses for your 10Gb NICs before proceeding.

The MAC addresses for your NICs can be found by running the command:

```
ip a
```

You should then create a UHD configuration file at the location `/root/.uhd/uhd.conf`.

```
sudo su
mkdir -p /root/.uhd
nano /root/.uhd/uhd.conf
```

An example `uhd.conf` file is listed below.

You should update the following fields for your configuration from this example:

- Update the MAC address variables, `dpdk-mac`, to match your NIC
- Update the `dpdk-driver` if the location is different on your system. `/usr/lib/x86_64-linux-gnu/dpdk-17.11-drivers/` is the default location on Ubuntu 18.04.x when `dpdk` is installed via `apt`.
- Update the `dpdk-corelist` and `dpdk-io-cpu` fields. In this example, a two port NIC is used. There should be one core for the main `dpdk` thread (in this example `core #2`), and then separate cores assigned to each NIC (in this example `core #3` for the first port on the NIC, `core #4` for the second port on the NIC)
- Update the `dpdk-ipv4` fields to your desired IP range.
  - `192.168.30.2`, `192.168.40.2` on a default X3xx system
  - `192.168.10.2`, `192.168.20.2` on a default N3xx system
  - `192.168.10.2` on a default E320 system

```
[use_dpdk=1]
dpdk-mtu=9000
dpdk-driver=/usr/lib/x86_64-linux-gnu/dpdk-17.11-drivers/
dpdk-corelist=2,3,4
dpdk-num-mbufs=4095
dpdk-mbufs-cache-size=315

[dpdk-mac=aa:bb:cc:dd:ee:f1]
dpdk-io-cpu = 3
dpdk-ipv4 = 192.168.10.1/24

[dpdk-mac=aa:bb:cc:dd:ee:f2]
dpdk-io-cpu = 4
dpdk-ipv4 = 192.168.20.1/24
```

Note: Additional information on the UHD configuration file can be found here:
https://files.ettus.com/manual_archive/v3.15.0.0/html/page_dpdk.html#dpdk_nic_config

The process for this step is different for Intel and Mellanox NICs and is detailed in individual sections below.

The Intel based NICs will use the `vfio-pci` driver which must be loaded:

```
sudo modprobe vfio-pci
```

Next, you will need to rebind the NIC to the `vfio-pci` drivers.

First, identify the PCI address your NIC is at:

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

Note the PCI address that your NIC is connected to for the next step.

Before the next step, you will need to turn off the NIC first before doing the rebind.

In Ubuntu under `System` -> `Network` -> click the switches to `off` for the 10Gb ports, then run the `dpdk-devbind` commands:

Note: Your PCI address will likely be different than `02:00.0` as shown in the example below.

```
sudo dpdk-devbind --bind=vfio-pci 02:00.0
sudo dpdk-devbind --bind=vfio-pci 02:00.1
```

Now if you run `dpdk-devbind --status` again, you should see the NICs listed under DPDK devices

```
# dpdk-devbind --status

Network devices using DPDK-compatible driver
============================================
0000:02:00.0 '82599ES 10-Gigabit SFI/SFP+ Network Connection 10fb' drv=vfio-pci unused=ixgbe
0000:02:00.1 '82599ES 10-Gigabit SFI/SFP+ Network Connection 10fb' drv=vfio-pci unused=ixgbe
```

Note: More info can be found here on the rebinding process:
https://doc.dpdk.org/guides-17.11/linux_gsg/linux_drivers.html#binding-and-unbinding-network-ports-to-from-the-kernel-modules

The Mellanox NICs do not require rebinding using the `vfio-pci` driver. Mellanox provides additional drivers for DPDK.

Install and activate the Mellanox drivers:

```
sudo apt install librte-pmd-mlx5-17.11
sudo modprobe -a ib_uverbs mlx5_core mlx5_ib
```

UHD based application (including GNU Radio flowgraphs) can now be ran using a DPDK transport by passing in the Device Argument: `use_dpdk=1`.

Important Note: In order for UHD to use DPDK, the UHD application *must* be ran as the `root` user. Using `sudo` will not work, you should switch to the `root` user by running `sudo su`.

For example, running the `benchmark_rate` utility:

```
# cd /usr/local/lib/uhd/examples

# ./benchmark_rate --rx_rate 125e6 --rx_subdev "A:0 B:0" --rx_channels 0,1 --tx_rate 125e6 --tx_subdev "A:0 B:0" --tx_channels 0,1 --args "ad

[INFO] [UHD] linux; GNU C++ version 7.3.0; Boost_106501; UHD_3.14.0.HEAD-0-gabf0db4e
EAL: Detected 8 lcore(s)
EAL: Some devices want iova as va but pa will be used because.. EAL: IOMMU does not support IOVA as VA
EAL: No free hugepages reported in hugepages-1048576kB
EAL: Probing VFIO support...
EAL: VFIO support initialized
EAL: PCI device 0000:02:00.0 on NUMA socket -1
EAL:   Invalid NUMA socket, default to 0
EAL:   probe driver: 8086:10fb net_ixgbe
EAL:   using IOMMU type 1 (Type 1)
EAL: Ignore mapping IO port bar(2)
EAL: PCI device 0000:02:00.1 on NUMA socket -1
EAL:   Invalid NUMA socket, default to 0
EAL:   probe driver: 8086:10fb net_ixgbe
EAL: Ignore mapping IO port bar(2)
PMD: ixgbe_dev_link_status_print():  Port 0: Link Down
EAL: Port 0 MAC: aa bb cc dd ee f1
EAL: Port 0 UP: 1
PMD: ixgbe_dev_link_status_print():  Port 1: Link Down
EAL: Port 1 MAC: aa bb cc dd ee f2
EAL: Port 1 UP: 1
EAL: Init DONE!
EAL: Starting I/O threads!
USER2: Thread 1 started
[00:00:00.000003] Creating the usrp device with: addr=192.168.10.2,second_addr=192.168.20.2,mgmt_addr=10.2.1.19,master_clock_rate=125e6,use_d
[INFO] [MPMD] Initializing 1 device(s) in parallel with args: mgmt_addr=10.2.1.19,type=n3xx,product=n310,serial=313ABDA,claimed=False,addr=19
[INFO] [MPM.PeriphManager] init() called with device args 'product=n310,time_source=internal,master_clock_rate=125e6,clock_source=internal,us
[INFO] [0/DmaFIFO_0] Initializing block control (NOC ID: 0xF1F0D000000000004)
[INFO] [0/DmaFIFO_0] BIST passed (Throughput: 1344 MB/s)
[INFO] [0/DmaFIFO_0] BIST passed (Throughput: 1341 MB/s)
[INFO] [0/DmaFIFO_0] BIST passed (Throughput: 1348 MB/s)
[INFO] [0/DmaFIFO_0] BIST passed (Throughput: 1347 MB/s)
[INFO] [0/Radio_0] Initializing block control (NOC ID: 0x12AD100000011312)
[INFO] [0/Radio_1] Initializing block control (NOC ID: 0x12AD100000011312)
[INFO] [0/DDC_0] Initializing block control (NOC ID: 0xDDC0000000000000)
[INFO] [0/DDC_1] Initializing block control (NOC ID: 0xDDC0000000000000)
[INFO] [0/DUC_0] Initializing block control (NOC ID: 0xD0C0000000000002)
[INFO] [0/DUC_1] Initializing block control (NOC ID: 0xD0C0000000000002)
Using Device: Single USRP:
  Device: N300-Series Device
  Mboard 0: ni-n3xx-313ABDA
  RX Channel: 0
    RX DSP: 0
    RX Dboard: A
    RX Subdev: Magnesium
  RX Channel: 1
    RX DSP: 0
    RX Dboard: B
    RX Subdev: Magnesium
  TX Channel: 0
    TX DSP: 0
    TX Dboard: A
    TX Subdev: Magnesium
  TX Channel: 1
    TX DSP: 0
    TX Dboard: B
    TX Subdev: Magnesium
```

```
[00:00:03.728707] Setting device timestamp to 0...
[INFO] [MULTI_USRP]     1) catch time transition at pps edge
[INFO] [MULTI_USRP]     2) set times next pps (synchronously)
[00:00:05.331920] Testing receive rate 125.000000 Msps on 2 channels
[00:00:05.610789] Testing transmit rate 125.000000 Msps on 2 channels
[00:00:15.878071] Benchmark complete.


Benchmark rate summary:
  Num received samples:     2557247854
  Num dropped samples:      0
  Num overruns detected:    0
  Num transmitted samples:  2504266704
  Num sequence errors (Tx): 0
  Num sequence errors (Rx): 0
  Num underruns detected:   0
  Num late commands:        0
  Num timeouts (Tx):        0
  Num timeouts (Rx):        0


Done!
```

The Application Note linked below covers general performance tuning tips that should be applied:

- https://kb.ettus.com/USRP_Host_Performance_Tuning_Tips_and_Tricks

If you experience `Overflows` at higher data rates, adding the device argument `num_recv_frames=512` can help.

If you're streaming data at the full master clock rate, and there is no interpolation or decimation being performed on the FPGA, you can skip the DUC and DDC blocks within the FPGA with the following parameters:

- `skip_ddc=1`
- `skip_duc=1`

If you're streaming two transmit channels at full rate (200e6) on the X3xx platform, you should additionally set the device arg:

- `enable_tx_dual_eth=1`

Isolating the CPUs that are used for DPDK can improve performance. This can be done by adding the `isolcpus` parameter to your `GRUB_CONFIG`

```
isolcpus=2,3,4
```

Disabling system interrupts can improve the jitter and performance generally by 1-3%. This can be done by adding the parameters below to your `GRUB_CONFIG`

```
nohz_full=2,3,4 rcu_nocbs=2,3,4
```

In some applications which require the highest possible CPU performance per core, disabling hyper-threading can provide roughly a 10% increase in core performance, at the cost of having less core threads. Hyper-threading can be disabled within the BIOs and varies by manufacturer.

- Performance report from Intel on DPDK 17.11: https://fast.dpdk.org/doc/perf/DPDK_17_11_Intel_NIC_performance_report.pdf
- How to get best performance with NICs on Intel platforms: https://doc.dpdk.org/guides/linux_gsg/nic_perf_intel_platform.html