APPLICATION NOTE

Intel Corporation

Intel® Speed Select Technology – Base Frequency Priority CPU Management for Open vSwitch* (OVS*)

Authors

Ian Stokes

Chris MacNamara

David Hunt

1 Overview

This application note describes how to configure Open vSwitch* (OvS*) with the Data Plane Development Kit (DPDK) to use a specific Poll Mode Driver (PMD) mask to ensure "priority" CPUs (that is, CPUs that can operate at a guaranteed higher frequency) are used for packet processing activity. This document also demonstrates how the OVS DPDK's existing features can be used to assign PMDs to CPUs in a system that already takes advantage of SST-BF. Because of this, the setup or configuration of SST-BF is outside of the scope of this application note.

This application note is written for network administrators who want to use this OVS capability to specify the CPU PMD mask and receive queue isolation. Doing this maximizes the throughput on a platform that has the ability to provide priority and non-priority CPUs.

The configuration described in this document utilizes a CPU capability called Intel® Speed Select Technology – Base Frequency (Intel® SST-BF), which is available on select SKUs of 2nd generation Intel® Xeon® Scalable processor (5218N, 6230N, and 6252N). The placement of key workloads on higher frequency Intel® SST-BF enabled cores can result in an overall system workload increase and potential overall energy savings when compared to deploying the CPU with symmetric core frequencies. For more information about Intel® SST-BF, refer to *Intel® Speed Select Technology – Base Frequency - Enhancing Performance* and *Intel® Speed Select Technology - Base Frequency (Intel® SST-BF) with Kubernetes** in Table 2.

This document is part of the Network Transformation Experience Kit, which is available at: https://networkbuilders.intel.com/network-technologies/network-transformation-exp-kits

1.1 Required Software

For testing purposes, we used Open vSwitch v2.10 with DPDK v17.11.4.

The PMD mask specification and receive queue isolation feature is available for Open vSwitch v2.6 and subsequent releases.

If you want to access the most recent development features, use the OVS master branch, which can be downloaded from GitHub* at:
https://github.com/openvswitch/ovs/tree/master

When using the OVS master branch, refer to the installation steps for OVS with DPDK at:
https://docs.openvswitch.org/en/latest/intro/install/dpdk/

Note: Because the setup and component requirements differ for various OVS with DPDK releases, ensure that you follow the steps relevant to your OVS deployment.

Table of Contents

Figures

Tables

1.2 Terminology

Table 1. Terminology

ABBREVIATION	DESCRIPTION
DPDK	Data Plane Development Kit
NIC	Network Interface Card
NUMA	Non-Uniform Memory Access
OVS	Open vSwitch
PMD	Poll Mode Driver
SKU	Stock Keeping Unit
SST	Intel® Speed Select Technology
SST-BF	Intel® Speed Select Technology – Base Frequency

1.3 Reference Documents

Table 2. Reference Documents

REFERENCE	SOURCE
Configure Open vSwitch* with Data Plane Development Kit on Ubuntu* Server 17.04	https://software.intel.com/en-us/articles/set-up-open-vswitch-with-dpdk-on-ubuntu-server
Intel® Speed Select Technology (Intel® SST)	https://www.intel.com/content/www/us/en/architecture-and-technology/speed-select-technology-article.html
Intel® Speed Select Technology – Base Frequency - Enhancing Performance	https://builders.intel.com/docs/networkbuilders/intel-speed-select-technology-base-frequency-enhancing-performance.pdf
Intel® Speed Select Technology - Base Frequency (Intel® SST-BF) with Kubernetes*	https://builders.intel.com/docs/networkbuilders/intel-speed-select-technology-base-frequency-with-kubernetes-application-note.pdf
Open vSwitch* with DPDK Overview	https://software.intel.com/en-us/articles/open-vswitch-with-dpdk-overview

2 Identify Priority CPUs

In a symmetric core frequency deployment (default), all cores on a processor operate at the same frequency. When the Intel® SST-BF is activated and configured, it allows the CPU to operate in an asymmetric configuration with two frequency tiers. This enables users to boost performance of targeted applications at runtime by assigning them to the higher frequency cores. For details, refer to the reference documents in Table 2.

The placement of key workloads on higher frequency Intel® SST-BF enabled cores can result in an overall system workload increase and potential overall energy savings when compared to deploying the CPU with symmetric core frequencies.

Priority CPUs can perform at a guaranteed higher frequency when compared to other CPUs on the same socket. On platforms, priority CPUs can differ in number and placement.

In this document, as an example, we assume that the priority CPUs are CPUs 2, 5, and 8 on socket 0 of a given system.

3 Isolate CPUs from the Linux* Process Scheduling

By default, the Linux kernel can potentially schedule processes on one of the given priority cores. If one of the CPUs is also used as a PMD for the receive queues of a network device, then the performance of the OVS* DPDK application can be impacted.

To avoid impacting the performance, you can complete the optional step of isolating the CPUs using the `isolcpus` kernel boot parameter:

1. Add or update the `isolcpus` kernel boot parameter:
   isolcpus =2,5,8
2. Reboot the system.
3. Use the following command to confirm that the CPUs are isolated:
   `cat /sys/devices/system/cpu/isolated`

The output should be similar to Figure 1.

```
-bash-4.4$ cat /sys/devices/system/cpu/isolated
2,5,8
```

Figure 1. Output of Command to Display Isolated CPUs

4 Configure the lcore Mask for OVS* with DPDK

The `dpdk-lcore-mask` specifies the CPU cores which the DPDK lcore threads are to spawn. The DPDK lcore threads are used for DPDK library tasks, such as library internal message processing, logging, etc. The value is specified in hexadecimal (hex) format.

If not specified, the value is determined by choosing the lowest CPU core from the initial CPU affinity list.

Note: The lcore is not responsible for the datapath processing. This task is handled by CPUs specified in the OVS DPDK PMD mask.

For performance reasons, it is best to set the lcore to a single CPU on the system, rather than allowing lcore threads to float.

Because the lcore is **not** responsible for packet processing on the OVS DPDK datapath, the best option is to set this lcore to a non-priority (lower frequency), non-isolated CPU, for example, CPU 1.

The following is an example command that shows how to achieve this:

```
ovs-vsctl set Open_vSwitch . other_config:dpdk-lcore-mask="0x2"
```

5 Configure the PMD Mask for OVS* with DPDK

`pmd-cpu-mask` specifies the CPU mask for setting the CPU affinity of the PMD threads. These threads are responsible for datapath processing (Rx/Tx) of the traffic within the OVS with DPDK. The value is specified in hexadecimal (hex) format.

If not specified, one PMD thread is created for each Non-Uniform Memory Access (NUMA) node and by default, pinned to any available CPU on the NUMA node.

For a platform with priority CPUs, it can be advantageous to specify CPUs with the higher frequency as part of the PMD mask. This allows for an increased level of packet processing on these CPUs when compared to non-priority CPUs, which have a lower frequency.

For example, with priority CPUs 2, 5, and 8, we can specify the core masks shown in Table 3.

Table 3. PMD Mask Examples for Use with CPUs 2, 5, and 8

CPU for Use by PMD	PMD Masks
2	0x4
5	0x20
8	0x100
2 and 5	0x24
2 and 8	0x104
2, 5, and 8	0x124

You can issue the following command to specify CPUs 2, 5, and 8 in the following PMD mask:

```
ovs-vsctl set Open_vSwitch . other_config:pmd-cpu-mask= "0x124"
```

The assumed setup is like that shown in Figure 2. The deployment reflects bridge `br0`, port `dpdk0`, and PMD mask `0x124`.

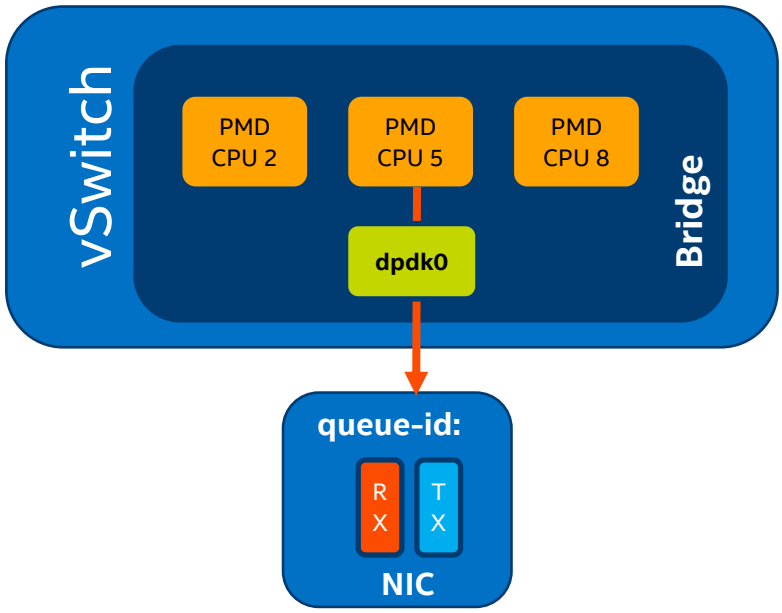

Figure 2. OVS with DPDK Deployment

You can confirm that the CPUs are used as part of the PMD mask with the following command:
`ovs-appctl dpif-netdev/pmd-rxq-show`

Executing this command produces an output similar to Table 3.

```
-bash-4.4$ sudo $OVS_DIR/utilities/ovs-appctl dpif-netdev/pmd-rxq-show
pmd thread numa_id 0 core_id 2:
  isolated: false
pmd thread numa_id 0 core_id 5:
  isolated: false
  port: dpdk0             queue-id:  0  pmd usage: NOT AVAIL
pmd thread numa_id 0 core_id 8:
  isolated: false
-bash-4.4$
```

Figure 3. pmd-rxq-show Command – Sample Output

The key points in this output are the CPU IDs being reported by OVS with DPDK, that is, **`core_id 2`**, **`core_id 5`**, and **`core_id 8`**. These IDs correspond to the PMD mask value specified, `0x124`, which corresponds to the priority cores on the platform.

6 Isolate Receive Queue

In our example, we assumed that all CPUs added as part of the PMD mask correspond to priority cores that operate at a higher frequency.

However, it is possible that of the three PMDs we specify, only one CPU is a priority CPU, and the other CPUs, defined as part of the PMD mask, are lower priority.

Let's assume CPU 2 is a priority CPU and that CPUs 5 and 8 are lower priority. It is possible that a port (for example, a 10 Gbit/s NIC, `dpdk0`) has high traffic throughput and therefore should be assigned to the priority CPU (CPU 2).

However, by default, OVS* with DPDK distributes receive queues for devices among PMDs automatically.

There are two methods to achieve this:

- Method 1: Use a *cycles* assignment, where the receive queues are ordered by their measured processing cycles, and then evenly assigned in descending order to PMDs based on an up/down walk of the PMDs.
- Method 2: Use a *round-robin* assignment, where the receive queues are assigned to PMDs in a round-robin scheme.

In both cases, there is a risk that a receive queue for a port is not assigned to a priority CPU, which can be mitigated by isolating the receive queues.

To ensure that the receive queue for a port is serviced by the priority CPU, you can isolate the receive queues to the specific CPU using the following command:

```
ovs-vsctl set Open_vSwitch . other_config:pmd-rxq-assign=<assignment>
```

In our example, we want to isolate receive queue 0 of the `dpdk0` interface on CPU 2 with the following command:

```
# Set receive queue 0 of dpdk0 to core 2
ovs-vsctl set interface dpdk0 other_config:pmd-rxq-affinity="0:2"
```

The configuration is now similar to Figure 4.

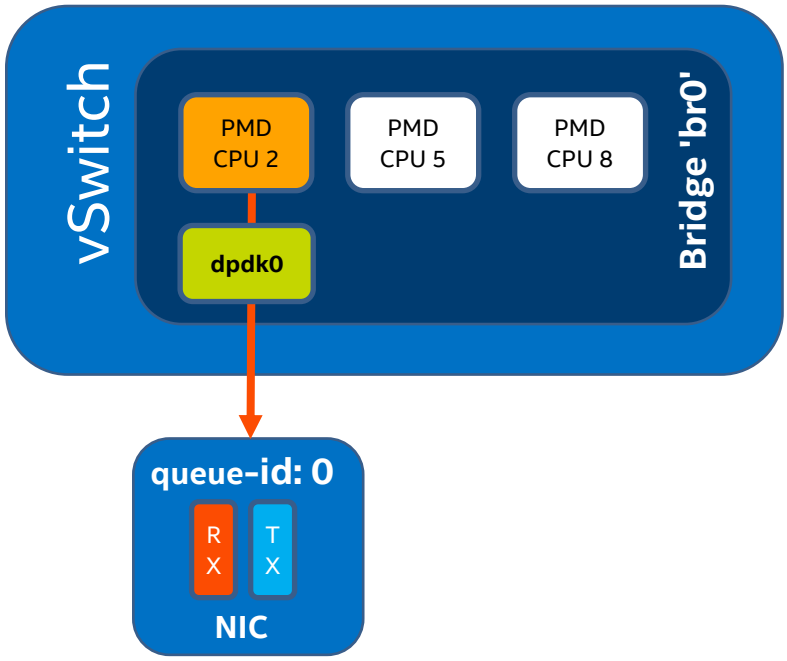

Figure 4. dpdk0 queue-id 0 Affinitized and Isolated to the PMD on CPU 2

Confirm this configuration by executing the following command:

```
ovs-appctl dpif-netdev/pmd-rxq-show
```

Figure 5 shows sample output.

```
-bash-4.4$ sudo $OVS_DIR/utilities/ovs-appctl dpif-netdev/pmd-rxq-show
pmd thread numa_id 0 core_id 2:
  isolated: true
  port: dpdk0             queue-id:  0  pmd usage: NOT AVAIL
pmd thread numa_id 0 core_id 5:
  isolated: false
pmd thread numa_id 0 core_id 8:
  isolated: false
-bash-4.4$
```

Figure 5. pmd-rxq-show Command – Sample Output with Core 2 Isolated

The key point in this output is that queue-id `0` for the `dpdk0` port is now affinitized to `core-id 2`. The isolated flag is now equal to `true` for core 2.

This means that OVS DPDK does not assign any other receive queues from other ports to CPU 2, ensuring that PMD workloads on priority CPU 2 are not interrupted by other port receive queue workloads.

7 Conclusion

In this application note, we demonstrated how to isolate priority CPUs at the kernel process scheduling level using the "`isolcpu`" kernel boot parameter, ensuring the kernel tasks will not be scheduled on a CPU that is used as a PMD in the OVS DPDK (which could otherwise affect performance of the PMD). We examined how to configure an OVS DPDK PMD mask using the "`pmd-cpu-mask`" parameter to correspond to specific CPUs, allowing CPUs with higher based frequencies to be used as a PMD for improved performance (up to a 1.25x-1.58x NFV workload performance improvement). Finally, we described how it is possible to isolate the receive queues for a given port to a specific PMD using the "`pmd-rxq-affinity`" argument, along with its associated parameters, ensuring receive queues with high throughput can be mapped and to a high frequency CPU in a deployment where both low and high frequency CPUs are being used for PMDs in OVS DPDK.

8 Additional Information

Have a question? Feel free to follow up with the query on the Open vSwitch* discussion mailing thread.

To learn more about Open vSwitch with DPDK in general, we encourage you to check out the following videos and articles in the Intel® Developer Zone and the Intel® Network Builders University:

- http://www.openvswitch.org/
- http://docs.openvswitch.org/en/latest/intro/install/dpdk/
- http://docs.openvswitch.org/en/latest/topics/dpdk/pmd/

1019/DN/PTI/PDF

341171-001US