Partial Offload Optimization and Performance on Intel Ethernet 700 Series NICs Using rte_flow

Irene Liew and Chenmin Sun (Intel)

(*) Presentation covered by Mesut Ali Ergin
OVS: hw-offload

- Experimental feature, utilizing rte_flow, was first available in OVS 2.10 / Aug 2018
  
  ```bash
  ovs-vsctl set Open_vSwitch . other_config:hw-offload=true
  ```

- Only supports *partial offloading*

- Allows OVS to skip costly OPs
  - MF Extraction
  - EMC Lookup
  - DPCLS Lookup

- Protocols supported:
  - L2: Ethernet, VLAN
  - L3: IPv4, IPv6
  - L4: TCP, UDP, SCTP, ICMP

(*) [https://patchwork.ozlabs.org/cover/916146/](https://patchwork.ozlabs.org/cover/916146/)
OVS: hw-offload

- Experimental feature, utilizing rte_flow, was first available in OVS 2.10 / Aug 2018
  - `ovs-vsctl set Open_vSwitch . other_config:hw-offload=true`
- Only supports *partial offloading*
- Allows OVS to skip costly OPs
  - MF Extraction
  - EMC Lookup
  - DPCLS Lookup
- Protocols supported:
  - L2: Ethernet, VLAN
  - L3: IPv4, IPv6
  - L4: TCP, UDP, SCTP, ICMP

(*) [https://patchwork.ozlabs.org/cover/916146/](https://patchwork.ozlabs.org/cover/916146/)
OVS: hw-offload

- Experimental feature, utilizing rte_flow, was first available in OVS 2.10 / Aug 2018*
  
  ```
  ovs-vsctl set Open_vSwitch . other_config:hw-offload=true
  ```

- Only supports *partial offloading*
- Allows OVS to skip costly OPs
  - MF Extraction
  - EMC Lookup
  - DPCLS Lookup
- Protocols supported:

  L2: Ethernet, VLAN - L3: IPv4, IPv6 - L4: TCP, UDP, SCTP, ICMP

(*) [https://patchwork.ozlabs.org/cover/916146/](https://patchwork.ozlabs.org/cover/916146/)
OVS: hw-offload

- Experimental feature, utilizing rte_flow, was first available in OVS 2.10 / Aug 2018*
  
  `ovs-vsctl set Open_vSwitch . other_config:hw-offload=true`

- Only supports *partial offloading*
- Allows OVS to skip costly OPs
  - MF Extraction
  - EMC Lookup
  - DPCLS Lookup
- Protocols supported:

  L2: Ethernet, VLAN - L3: IPv4, IPv6 - L4: TCP, UDP, SCTP, ICMP

(*) https://patchwork.ozlabs.org/cover/916146/
• Intel® Ethernet 700 Series Network Adapters (10/25/40GbE)
  • rte_flow driver in i40e PMD since flow API’s inception (2017)
  • Utilizes Intel® Ethernet Flow Director feature in the controller
  • QUEUE, PASSTHRU, DROP, FLAG, MARK actions were available
  • Allows for up to 8K rules in device memory

• NEW in DPDK 19.08
  • Support for MARK + RSS actions added to rte_flow

• NEW in DPDK 19.11
  • Flow Director support for i40e vector RX path (SSE4.2 and AVX2)

• OVS hw-offload works with Intel® Ethernet 700 Series Controllers
  • Try in dpdk-next branch (now)
  • Out-of-box with OVS 2.13 (when released in 03/2020)
OVS: hw-offload Test Setup

**Hardware**

- **Platform**: Supermicro X11DPH-T
- **CPU**: Intel(R) Xeon(R) Platinum 8180 CPU @ 2.50GHz
- **MEMORY**: Micron, DDR4 2666MHz, 16GB per Channel, 12 Channels, TOTAL: 192 GB
- **NIC**: Intel® Ethernet XL710 DA2 Adapter
- **BIOS**: American Megatrends Inc., version 3.1 dated 05/22/2019

**Software**

- **Host OS**: Ubuntu* 19.04
- **Host Kernel version**: 5.0.0-23-generic
- **BIOS settings**: P-state Disabled, C-States Disabled
- **SW Version**: DPDK 19.11-rc1, OVS 2.11.1 (dpdk-latest)
- **IXIA TEST**: RFC 2544 0.01% PACKET LOSS

**Configurations**

- **Flows (Millions)**: 1, 10
- **Rules (Thousands)**: 1, 7, 10
- **Packet Size (Bytes)**: 64, 256, 512, 1024
- **Protocols Matched**: IP, UDP, TCP
- **CPU Cores Used**: 1
OVS: hw-offload Throughput Experiment Results

For 1M Flows with/offload:
- 64 Packets/sec: Throughput ( Millions )
- 256 Packets/sec: Throughput ( Millions )
- 512 Packets/sec: Throughput ( Millions )
- 1024 Packets/sec: Throughput ( Millions )

For 1M Flows without/offload:
- 64 Packets/sec: Throughput ( Millions )
- 256 Packets/sec: Throughput ( Millions )
- 512 Packets/sec: Throughput ( Millions )
- 1024 Packets/sec: Throughput ( Millions )
OVS: hw-offload Throughput Experiment Results

**1K Rules**

Throughput (Packets/sec) Millions

- Throughput with offload
- Throughput without offload

Packet Size: 64, 256, 512, 1024

**7K Rules**

Throughput (Packets/sec) Millions

- Throughput with offload
- Throughput without offload

Packet Size: 64, 256, 512, 1024

**10K Rules**

Throughput (Packets/sec) Millions

- Throughput with offload
- Throughput without offload

Packet Size: 64, 256, 512, 1024

**PHY-PHY TESTS**

**PHY-VM-PHY TESTS**
OVS: hw-offload Throughput Experiment Results

- **1K Rules**
  - 1M Flows w/offload
  - 10M Flows w/offload
  - 1M Flows no offload
  - 10M Flows no offload
  - 1K Rules

- **7K Rules**
  - 1M Flows w/offload
  - 10M Flows w/offload
  - 1M Flows no offload
  - 10M Flows no offload
  - 7K Rules

- **10K Rules**
  - 1M Flows w/offload
  - 10M Flows w/offload
  - 1M Flows no offload
  - 10M Flows no offload
  - 10K Rules

**Throughput (Packets/sec) Millions**

**Packet Size**

**PHY-PHY TESTS**

**PHY-VM-PHY TESTS**
OVS: hw-offload Throughput Experiment Results

1K Rules

- 1M Flows w/offload
- 1M Flows no offload
- 10M Flows w/offload
- 10M Flows no offload

Throughput (Packets/sec): Millions

Packet Size: 64, 256, 512, 1024

7K Rules

Throughput (Packets/sec): Millions

Packet Size: 64, 256, 512, 1024

10K Rules

Throughput (Packets/sec): Millions

Packet Size: 64, 256, 512, 1024
OVS: hw-offload Throughput Experiment Results

Throughput Performance Improvements Up To
- 80% in PHY-PHY tests
- 25% in PHY-VM-PHY tests
OVS: hw-offload Core Scaling Experiment Results

1K Rules (64B)

PHY-PHY TEST

PHY-VM-PHY TEST
OVS: hw-offload Core Scaling Experiment Results

1K Rules (64B)

<table>
<thead>
<tr>
<th></th>
<th>1M flows w/o offload</th>
<th>1M Flows w/offload</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Throughput (Million Packets/sec)</td>
<td></td>
</tr>
<tr>
<td>1C1T - 2RXQ per core</td>
<td>4.7</td>
<td>8.5</td>
</tr>
<tr>
<td>2C2T - 2RXQ per core</td>
<td>10.0</td>
<td>15.5</td>
</tr>
<tr>
<td>2C2T - 4RXQ per core</td>
<td>9.9</td>
<td>14.6</td>
</tr>
<tr>
<td>4C4T - 2RXQ per core</td>
<td>23.4</td>
<td>22.9</td>
</tr>
<tr>
<td>4C4T - 4RXQ per core</td>
<td>19.5</td>
<td>19.0</td>
</tr>
</tbody>
</table>

PHY-PHY TEST

<table>
<thead>
<tr>
<th></th>
<th>1M flows w/o offload</th>
<th>1M Flows w/offload</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Throughput (Million Packets/sec)</td>
<td></td>
</tr>
<tr>
<td>1C1T - 4RXQ per core</td>
<td>2.0</td>
<td>2.5</td>
</tr>
<tr>
<td>2C2T - 2RXQ per core</td>
<td>4.2</td>
<td>5.3</td>
</tr>
<tr>
<td>4C4T - 2RXQ per core</td>
<td>7.9</td>
<td>9.7</td>
</tr>
<tr>
<td>8C8T - 2RXQ per core</td>
<td>16.1</td>
<td>17.4</td>
</tr>
</tbody>
</table>

PHY-VM-PHY TEST
• OVS DPDK hw-offload is
  • available on physical ports, now including Intel Ethernet 700 Series Devices
  • not supported on vhost-backend and virtio
  • able to offload rules matching on Ethernet, IP, TCP, and UDP protocols
  • Intel Ethernet 700 Series Devices can hold up to 8K rules in device memory

• Performance improvements are
  • due to cycles made available to flow access and action processing
  • consistent across packet sizes
  • proportional to the share of cycles for physical ports
  • diminishing as you use four or more cores per physical
The End

References:
DPDK Programmers Guide @ https://doc.dpdk.org/guides/prog_guide
Open vSwitch Hardware Acceleration Application Note @ https://bit.ly/2MbpJoA
HOWTO: Using OVS DPDK @ http://docs.openvswitch.org/en/latest/howto/dpdk

Acknowledgements:
Harry Van Haaren, Irene Liew, Chenmin Sun, Yipeng Wang, Charlie Tai, John McNamara, Edwin Verplanke

Contact us at:
npl@intel.com
OVS-DPDK Phy-Phy Core Scaling Test Configuration

- **2C2T**
  - Ubuntu-18.04
  - XVV710-DA2 2 x 25Gbps
  - 2RXQ per core

- **4C4T**
  - Ubuntu-18.04
  - XVV710-DA2 2 x 25Gbps
  - 4RXQ per core

- **8C8T**
  - Ubuntu-18.04
  - XVV710-DA2 2 x 25Gbps
  - 2RXQ per core
OVS-DPDK Phy-Phy Core Scaling Test Configuration

2C2T
- Ubuntu 18.04
- XVV710-DA2 2 x 25Gbps
- 2RXQ per core

4C4T
- Ubuntu 18.04
- XVV710-DA2 2 x 25Gbps
- 4RXQ per core

8C8T
- Ubuntu 18.04
- XVV710-DA2 2 x 25Gbps
- 2RXQ per core
OVS-DPDK Phy-VM-Phy Core Scaling Test Configuration

2C2T

Vhost0
OVS-DPDK
Vhost1
Core 2
Q0 Rx
Q1 Rx
dpdk0
Core 3
Q0 Rx
dpdk1
Ubuntu-18.04
P1
P2
XVV710-DA2 2 x 25Gbps

2RXQ per core

4C4T

Vhost0
OVS-DPDK
Vhost1
Core 2
Q0 Rx
Q1 Rx
Q2 Rx
Q3 Rx
dpdk0
Core 3
Q0 Rx
Q1 Rx
Q2 Rx
Q3 Rx
dpdk1
Core 4
Q0 Rx
Q1 Rx
Core 5
Q0 Rx
Ubuntu-18.04
P1
P2
XVV710-DA2 2 x 25Gbps

2RXQ per core

8C8T

Vhost0
Core 2
Q0 Rx
Q1 Rx
Q2 Rx
Core 3
Q3 Rx
Core 4
Q4 Rx
Core 5
Q5 Rx
Core 6
Q6 Rx
Core 7
Q7 Rx
Core 8
Q8 Rx
Core 9
Q9 Rx
Vhost1
dpdk0
Q0 Rx
Q1 Rx
Q2 Rx
Q3 Rx
dpdk1

Ubuntu-18.04
P1
P2
XVV710-DA2 2 x 25Gbps

2RXQ per core
OVS-DPDK Phy-VM-Phy Core Scaling Test Configuration

**2C2T**
- Ubuntu-18.04
- XVV710-DA2 2 x 25Gbps
- 2RXQ per core

**4C4T**
- Ubuntu-18.04
- XVV710-DA2 2 x 25Gbps
- 2RXQ per core

**8C8T**
- Ubuntu-18.04
- XVV710-DA2 2 x 25Gbps
- 2RXQ per core