

参会记录：2020 Intel Packet Processing Virtual Summit

1 会议信息

- (1) 会议时间：11月16日-21日
- (2) 参与时间：11月16日，11月17日
- (3) 会议日程：

日期 Date	时间 Time	主题 Subject	演讲嘉宾 Speaker
16-Nov	20:00 - 20:45	E810 DPDK PMD 简介 E810 DPDK PMD Introduction	张祺 英特尔资深网络软件工程师 Qi Zhang, Intel Senior Network Software Engineer
	20:45 - 21:30	DPDK -iAVF后端设备虚拟化 Device Virtualization in DPDK -iAVF Backend	王潇 英特尔资深网络软件工程师 Xiao Wang, Intel Senior Network Software Engineer
17-Nov	20:00 - 20:45	CPU动态负载均衡模块--来自Intel的新eventdev设备 CPU dynamic load balancing module-new eventdev device from Intel	卢秀春 英特尔资深网络软件工程师 Xiuchun Lu, Intel Senior Network Software Engineer
	20:45 - 21:30	面向数据中心的Intel Big Spring Canyon (BSC) SmartNIC Intel Big Spring Canyon (BSC) SmartNIC for Cloud Data Center	王栋 英特尔资深平台应用工程师 Dong Wang, Intel Senior Platform Application Engineer
18-Nov	20:00 - 20:45	从低级加密指令到高性能IPsec/TLS：安全传输加速的全面覆盖 From low level crypto instructions to high performance IPsec/TLS : a full coverage of secure transportation acceleration	胡雪焜 英特尔资深平台应用工程师 Xuekun Hu, Intel Senior Platform Application Engineer
	20:45 - 21:30	面向云友好的IPsec大象流：使用QAT或多核软件调度引擎的VPP异步加密 Towards Cloud-friendly IPsec Elephant Flow: VPP Asynchronous Crypto with QAT or Multi-core SW scheduler engines	许炜华 英特尔资深平台应用工程师 Rosen Xu, Intel Senior Platform Application Engineer
19-Nov	20:00 - 20:45	HDSL B简介：高密度可扩展负载均衡器 Introduction to HDSL B: High Density Scalable Load Balancer	刘勇 英特尔资深网络软件工程师 Yong Liu, Intel Senior Network Software Engineer
	20:45 - 21:30	基于DDP的Intel NIC的高级功能和5G UPF加速 Advanced features of Intel NIC and 5G UPF Acceleration based on DDP	虞平 英特尔资深网络软件工程师 Ping Yu, Intel Senior Network Software Engineer
21-Nov	09:00 - 09:45	Intel OpenNESS边缘计算平台参考设计及Testbed实践 Intel OpenNESS edge computing platform reference design and Testbed practice	张宇巍 英特尔资深网络软件工程师 Yuwei Zhang, Intel Senior Network Software Engineer
	09:45 - 10:30	圆桌会议 Panel Discussion	张帆 英特尔资深网络软件工程师 Fan Zhang, Intel Senior Network Software Engineer

2 会议内容

2.1 E810 DPDK PMD 简介

1) E810 DPDK PMD 文档: https://doc.dpdk.org/guides/rel_notes/release_20_11.html#tested-platforms

2) 首先介绍了 E810 DPDK 的发布版本及各版本特点，目前版本是 20.11，它对小包处理有 15-35%的性能提升，具体可以查看性能报告。

Intel® Ethernet 800 Series PMD Key Milestones

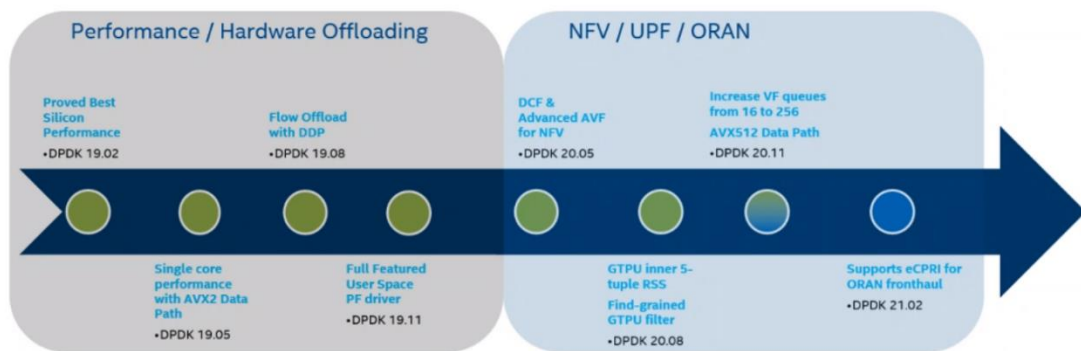


图 1 DPDK 发布版本及特点

然后介绍了 800 系列的特点，它有更多的硬件资源，可以有更多的 VF，Queue，可以卸载更多的 Flow。

From I40E PMD to ICE PMD

Function	Item	I40E	ICE
PF	Queue Pair Number	Max to 1024	Max to 2048
	VF Number	Max to 128	Max to 256
	Extract Match Flow	8K	20K ~ 30K
	Wildcard Match Flow	NO	512 ~ 2K
	Host Features for SR-IOV	Private API	Removed (DCF as replacement)
	DDP / PTYPE / Input Set	Private API	Generic Flow API (RTE_FLOW)
VF	Driver	I40EVF	IAVF
	Queue Pair Number	Max to 16	Max to 256
	Flow Offloading	NO	YES (resource shared with PF and other VFs)
Both	Performance	AVX2	AVX2 / AVX512

图 2 功能特性

接着介绍了 DPDK 的三种使用模式。

Deployment Mode

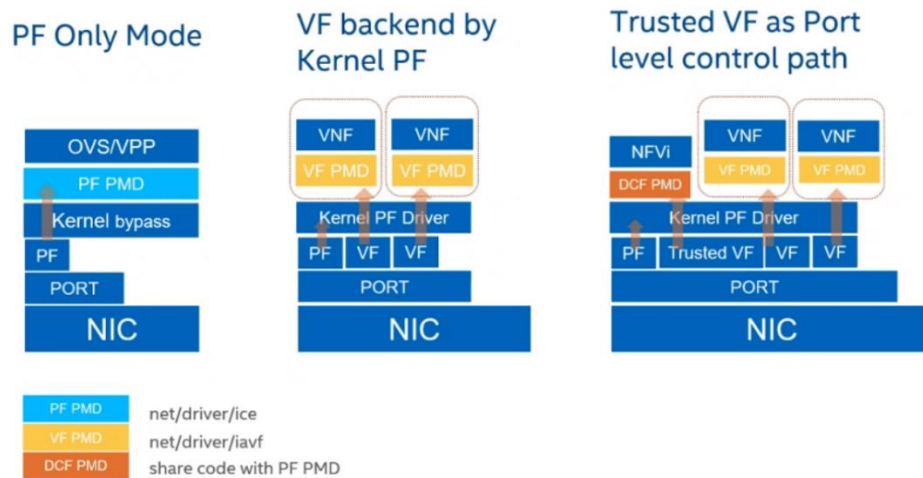
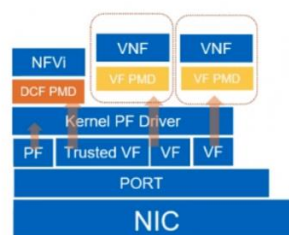


图 3 DPDK 使用模式

然后具体介绍了 DCF PMD 和 VF PMD。

Device Configure Function (DCF)

- A trusted VF driver backend by Kernel PF driver
- A complement for kernel tools (ethtool, ip, tc ...) to supported advanced flow offloading (Eg. PPPoE, PFCF, IPSec, L2TP ...)
- Use DPDK Hardware Acceleration Interface for NFVi, eg. RTE_FLOW to steering Package to VF



How to use DCF

- Create SR-IOV

```
#echo 4 > /sys/bus/pci/devices/0000\:\18\:\00.0/sriov_numvfs
```
- Turn on trust mode for VF0

```
#ip link set dev enp24s0f0 vf 0 trust on
```
- Use devargs "cap=dcf" to probe PMD

```
#testpmd -c 0x3 -n 4 -w 18:01.0,cap=dcf -- -i
```
- DCF Support RTE_FLOW

```
#flow create 0 priority 0 ingress pattern eth / ipv4 src is 192.168.0.2 dst is 192.168.0.3 / end
actions vf id 2 / end
```

NOTE:

DCF make sure rule still works after destination VF reset

图 4 DCF PMD 及其用法

然后举了一个 DPDK 的应用实例，即使用 800 系列优化 OVS (OpenvSwitch)：介绍 OVS 中 VXLAN overlay 的拓扑，然后概括 OVS 中的数据包处理流水线，之后介绍如何通过 SW 和 HW 的协同设计优化 OVS。

OVS Packet Pipeline

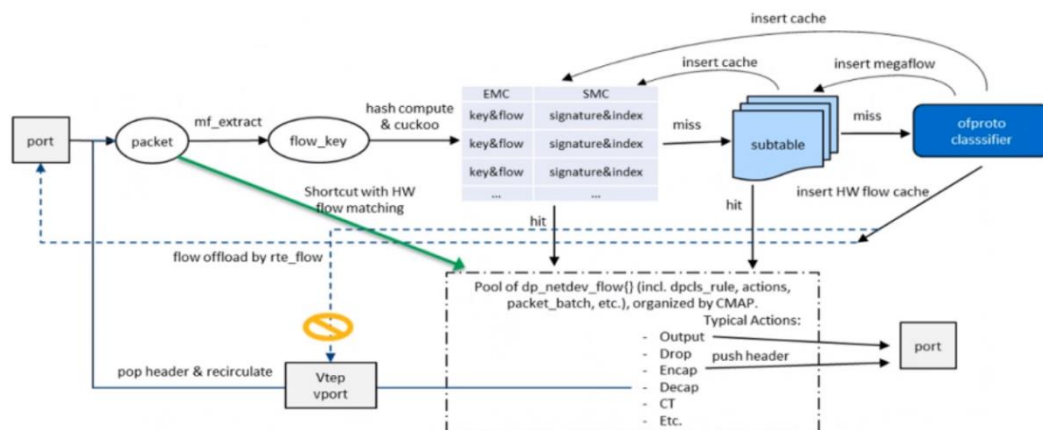


图 5 OVS 数据包流水线

Accumulative Improvement by SW Opti

- Adaptive polling instead of round robin
 - Avoids overhead of polling vhost ports that have no packets
 - Give more weight to active ports
- Reuse hash value for packets from the same microflow
 - Compute hash for first packet in a batch, use same hash for rest of the packets in a batch
- Compile time DPCLS lookup optimization for VXLAN
 - Leverage the helper MACRO provided by OVS
 - Adds two more dpcls lookup functions for VXLAN scenario
- A faster key extraction for common packet types (e.g. IP/UDP)
 - Mini flow extract is optimized for most common packet types
- Batching of header encap/decap
 - Encap and decap performed in batch, rather than on packet basis, same flow
- Queue size config for smaller memory footprint
 - Reduce LLC-load-miss events significantly

图 6 SW 优化

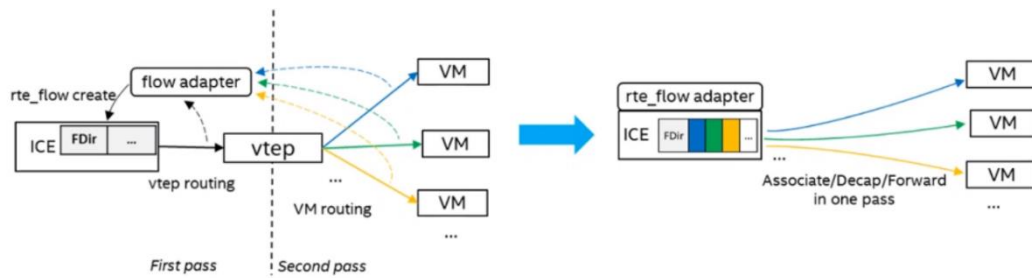
Rte_flow semantics of outer flow and inner flow

```
Attributes: ingress=1, egress=0, prio=0, group=0, transfer=0
rte flow eth pattern:
  Spec: src=10:00:00:00:00:02, dst=10:00:00:00:00:01,
  type=0x0800
  Mask: src=ff:ff:ff:ff:ff:ff, dst=ff:ff:ff:ff:ff:ff, type=0xffff
rte flow ipv4 pattern:
  Spec: tos=0x0, ttl=40, proto=0x11, src=172.1.0.200,
  dst=172.1.0.100
  Mask: tos=0x0, ttl=0, proto=0x0, src=0.0.0.0,
  dst=255.255.255.255
rte flow udp pattern:
  Spec: src_port=1000, dst_port=4789
  Mask: src_port=0x0, dst_port=0xffff
rte flow mark action:
  Mark: id=0
rte flow RSS action:
  RSS: queue_num=4
```

```
Attributes: ingress=1, egress=0, prio=0, group=0, transfer=0
rte flow eth pattern:
  Spec = null
  Mask = null
rte flow ipv4 pattern:
  Spec: tos=0x0, ttl=40, proto=0x0, src=172.1.0.0, dst=172.1.0.100
  Mask: tos=0xff, ttl=0, proto=0x0, src=255.255.255.255, dst=255.255.255.255
rte flow udp pattern:
  Spec: src_port=34233, dst_port=4789
  Mask: src_port=0x0, dst_port=0x0
rte flow vxlan pattern:
  Spec: vni=1001
  Mask: vni=0xfffff
rte flow eth pattern:
  Spec: src=a0:00:00:00:00:02, dst=a0:00:00:00:00:01, type=0x0800
  Mask: src=00:00:00:00:00:00, dst=ff:ff:ff:ff:ff:ff, type=0xffff
rte flow ipv4 pattern:
  Spec: tos=0x0, ttl=40, proto=0x11, src=192.1.0.200, dst=192.1.0.1
  Mask: tos=0x0, ttl=0, proto=0x0, src=0.0.0.0, dst=255.255.255.255
rte flow mark action:
  Mark: id=0
rte flow RSS action:
  RSS: queue_num=4
```

图 7 Rte_flow 的内部流和外部流示例

VXLAN Flow Offload with Flow Director



- First packet goes to slow path and trigger flow offload
- Subsequent packets are forwarded directly, reduce twice parsing and lookup
- 75% throughput up (bidirectional 1million flow) by SW + HW optimizations

图 8 使用 Flow Director 卸载 VXLAN 流

2.2 DLB

17 日主要听了一个多核转发模型 (DLB)，多核转发模型可以动态分配规则，识别所有报文类型，调度规则灵活，但占用额外的 CPU 核心。DLB 的主要组成部分包括队列、端口、事件。

多核转发模型：DLB

- ✓ 动态分配规则
- ✓ 可识别所有报文类型
- ✓ 调度规则灵活：按流/按包
- ✓ 硬件保序

□ 占用额外的CPU核心

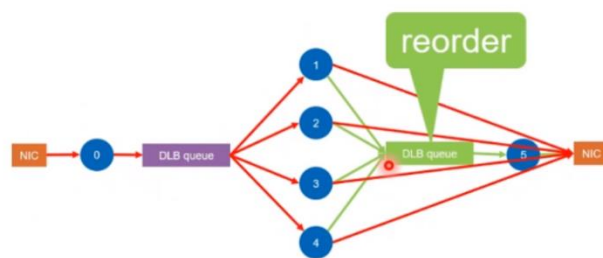


图 9 多核转发模型优缺点

DLB主要组成部分

▪ 队列

- a) 硬件资源，一个DLB包含多个队列，通过QID标识
- b) 有多种队列类型，调度方式不同
- c) 可同时使用多个队列且分别配置为不同类型，队列深度可配

▪ 端口

- a) 硬件资源，一个DLB包含多个端口，通过PortID标识
- b) 软件与队列交互的接口
- c) 需要与队列链接后才能使用，一个端口可以与多个队列链接（Direct Queue/Port除外）

▪ 事件（队列成员）

- a) 软件资源，共16字节，由下面2部分组成。
- b) 控制信息，由DLB使用，入队/出队时有不同的定义。
- c) 负载信息，可保存rte_mbuf指针或其他数据。

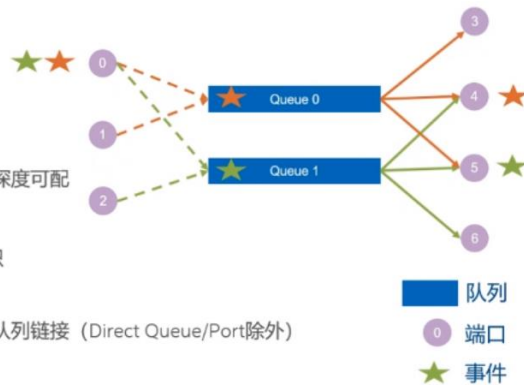


图 10 多和转发模型组成

3 心得体会

比起去年参加的 SD-WAN 好多了，至少知道在讲什么了，但是等想明白讲的这部分已经又讲过去好多内容了，继续努力吧。