DPDK
DATA PLANE DEVELOPMENT KIT

THE LINUX FOUNDATION

INTEL® 25GBE ADVANCED FEATURES FOR NFV

HELIN ZHANG, INTEL®

JINGJING WU, INTEL®

主办方： intel

参与方： 腾讯云 ZTE 美团云 Panabit 太一星晨 Balance Your Networks UnitedStack有云 云杉网络 Yunshan Networks

协办方： SDNLAB 专注网络创新技术　视频支持方： IT大咖说

Agenda

- Key Hardware Features
- Dynamic Device Personalization (DDP)
- Generic Flow API
- Virtual Function Daemon (VFD)
- Good Performance
- Adaptive Virtual Function (AVF)

Key Hardware Features

- PCIe v3.0, x8
- XXV710, 25GbE Link Speed
  - New addition to Intel® Ethernet 700 Series (10/25/40GbE)
- Network Virtualization offloads
  - VXLAN, NVGRE, GENEVE, VXLAN-GPE with NSH, MPLS, and more
- Input Set for RSS and Flow Director (FD)
  - Up to 24 of 56 words can be selected
- 3 HASH Algorithms
  - Toeplitz, Simple XOR, Symmetric Simple XOR

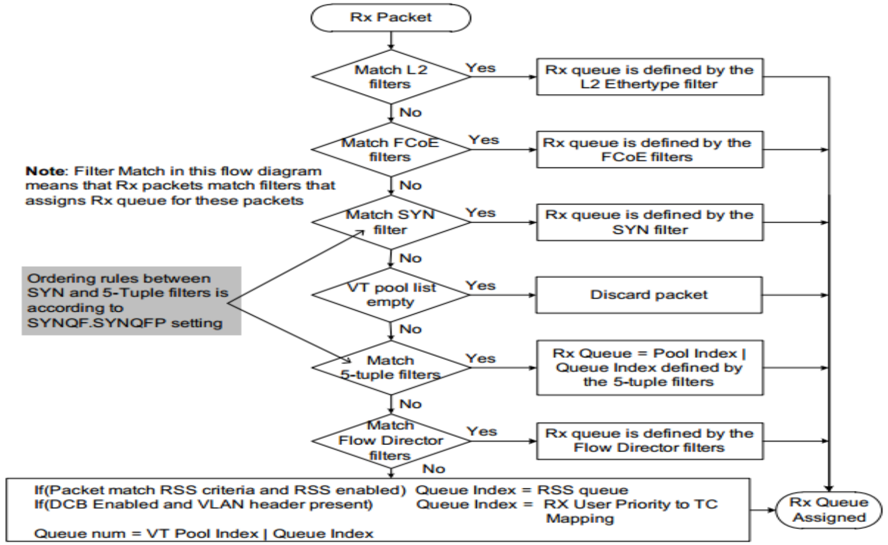
Key Hardware Features for Virtualization

Feature	**XXV710**	**82599EB**
SR-IOV support	Yes	Yes
VF to PF mailbox	Yes	Yes
Max Number of Virtual functions	128 per device (globally)	64 per port (single queue)
Max number of Queues	**1536**	128
Max number of queues per VF	**16**	8
Max number of queues per VMDq2 VSI	16	8
Max Number of VMDq2 ports	256 per device (globally)	64 per port (single queue)
MAC addresses	**1024** per device (globally)	128 per port
VLAN tags	**512** per device (global)	64 per port
Queuing to Pool/VSI method	SA, VLAN pairs or SA or VLAN	SA or VLAN or (SA and VLAN)
Cloud filter in Switch	Yes	No
RSS per VF	**Yes**	No (Single RSS used for all VFs).
Switching modes	VEB, VEPA	VEB*
Promiscuous modes per VM	VLAN, Multicast, Unicast	Multicast

Internal Packet Processing

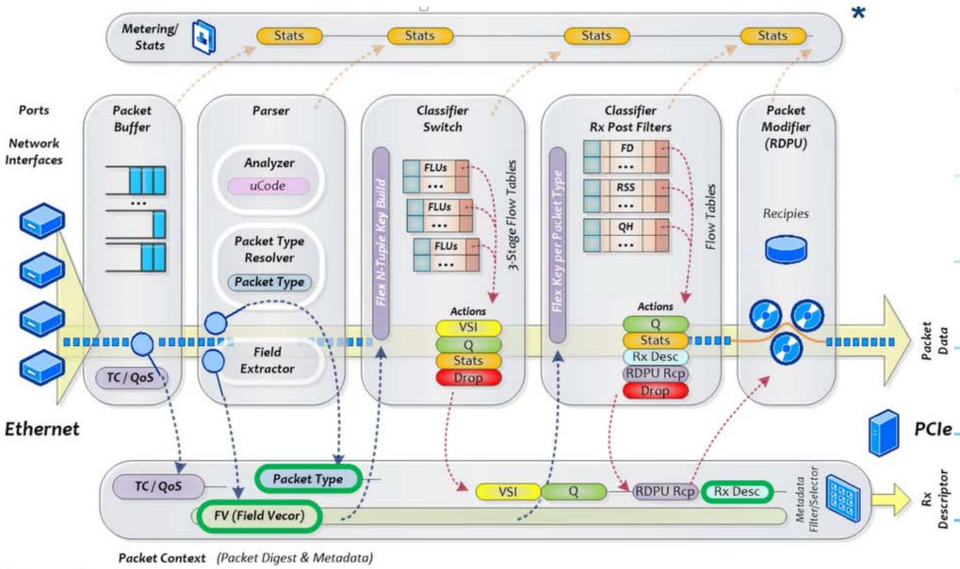
82599EB

- Fixed packets Parse graphic.
- Input set of filtering/steering is fixed.


Rx Packet
Match L2 filters
Yes
Rx queue is defined by the L2 Ethertype filter
No
Match FCoE filters
Yes
Rx queue is defined by the FCoE filters
Note: Filter Match in this flow diagram means that Rx packets match filters that assigns Rx queue for these packets
No
Match SYN filter
Yes
Rx queue is defined by the SYN filter
No
Ordering rules between SYN and 5-Tuple filters is according to SYNQF.SYNQFP setting
VT pool list empty
Yes
Discard packet
No
Match 5-tuple filters
Yes
Rx Queue = Pool Index | Queue Index defined by the 5-tuple filters
No
Match Flow Director filters
Yes
Rx queue is defined by the Flow Director filters
No
If(Packet match RSS criteria and RSS enabled) Queue Index = RSS queue
If(DCB Enabled and VLAN header present) Queue Index = RX User Priority to TC Mapping
Queue num = VT Pool Index | Queue Index
Rx Queue Assigned

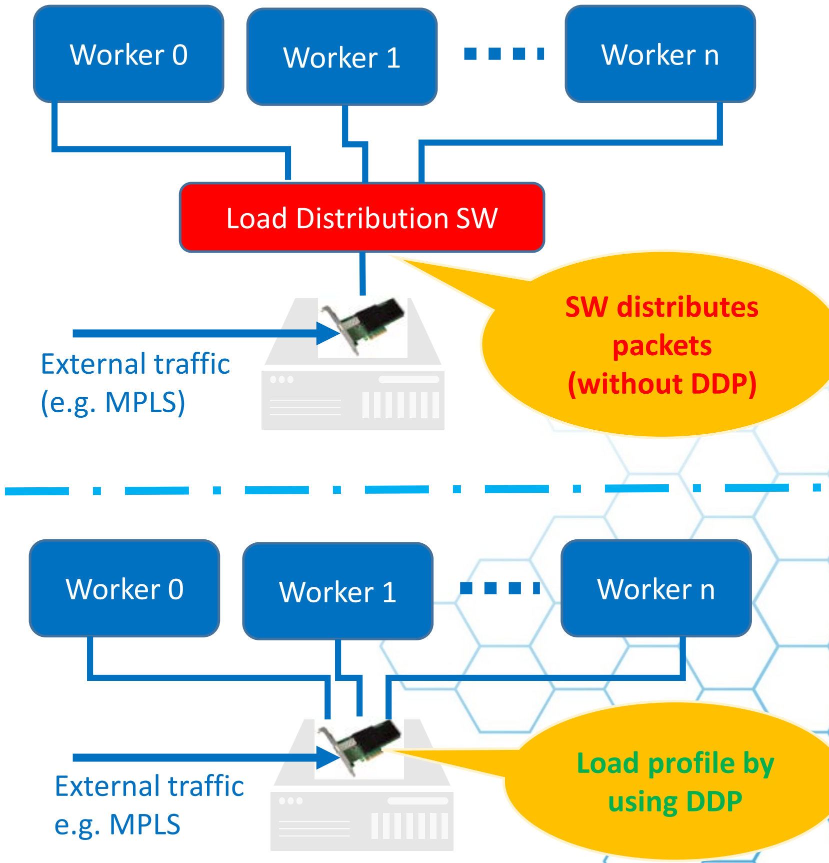
XXV710

- Configurable input set for RSS and FD.
- DDP to support more protocol steering.



Dynamical Device Personalization (DDP)

- By default, it supported limited protocols, due to hardware resources
  - e.g. VXLAN
- Loadable profiles for packet classification for extra protocols
  - e.g. MPLSoGRE
- Configurable tunnel filters for traffic steering
  - Steering packets to VM queues on QinQ/tunnel ID


Worker 0
Worker 1
Worker n
Load Distribution SW
External traffic (e.g. MPLS)
SW distributes packets (without DDP)
Worker 0
Worker 1
Worker n
External traffic e.g. MPLS
Load profile by using DDP

Generic Flow API Support

- A generic way to configure the hardware
  - Don’t need to know the HW specific filters
- Flow rule
  - Attributes
  - Matching pattern
  - Actions
- Rule management
  - rte_flow_validate()
  - rte_flow_create()
  - rte_flow_destroy()
  - rte_flow_flush()

Example

- Direct the VXLAN packet with specific inner MAC and VNI to queue #2.

Friendly, and consistent to applications!

Legacy filter control API

```
struct rte_eth_tunnel_filter_conf tunnel_filter_conf = {
        .outer_mac = {0x11, 0x22, 0x33, 0x44, 0x55, 0x66};
        .inner_mac = {0x00, 0x11, 0x22, 0x33, 0x44, 0x55};
        .inner_vlan = 0;
        .ip_type = RTE_TUNNEL_IPTYPE_IPV4;
        .ip_addr.ipv4_addr = 1;
        .filter_type = RTE_TUNNEL_FILTER_IMAC_TENID;
        .tunnel_type = RTE_TUNNEL_TYPE_VXLAN;
        .tenant_id = 1;
        .queue_id = 2;
};
int ret;

ret = rte_eth_dev_filter_ctrl(port_id, RTE_ETH_FILTER_TUNNEL,
RTE_ETH_FILTER_ADD, &tunnel_filter_conf);
```

Generic flow API

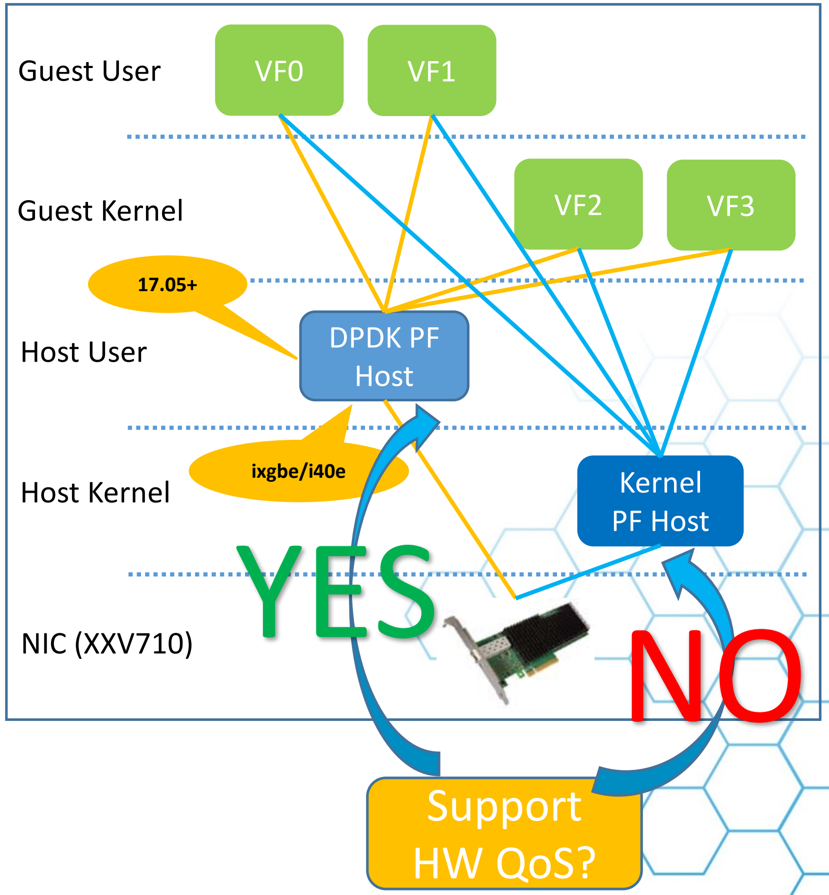
```
const struct rte_flow_item pattern[] = {
        { RTE_FLOW_ITEM_TYPE_ETH, NULL, NULL, NULL},
        { RTE_FLOW_ITEM_TYPE_IPV4, NULL, NULL, NULL},
        { RTE_FLOW_ITEM_TYPE_UDP, NULL, NULL, NULL},
        { RTE_FLOW_ITEM_TYPE_VXLAN, {.vni = 1}, NULL, {.vni = "\xff\xff\xff"}},
        { RTE_FLOW_ITEM_TYPE_ETH,
                {.dst = { 0x00, 0x11, 0x22, 0x33, 0x44, 0x55}}, NULL,
                {.dst = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}}},
        { RTE_FLOW_ITEM_TYPE_END, NULL, NULL, NULL},
};

const struct rte_flow_action actions[] = {
        { RTE_FLOW_ACTION_TYPE_PF, NULL},
        { RTE_FLOW_ACTION_TYPE_QUEUE, { .index = 2}},
        { RTE_FLOW_ACTION_TYPE_END, NULL},
};
struct rte_flow_error flow_err;

flow_err = rte_flow_create(port_id, NULL, pattern, actions, &flow_err);
```

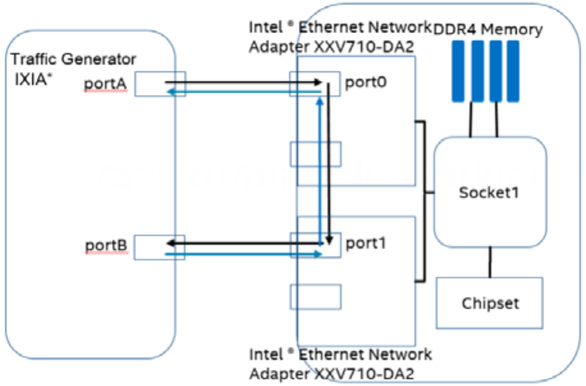
Virtual Function Daemon (VFD) Support

- DPDK as the host driver to support both DPDK and kernel VF
- Lots of VF management features are added
- Mailbox messages management are added
  - VF requests can be accepted/rejected by VFD
- Kernel driver does not support those features
- Only Intel® Ethernet 500 (ixgbe) and 700 (i40e) series are enabled
- Refer to https://github.com/att/vfd



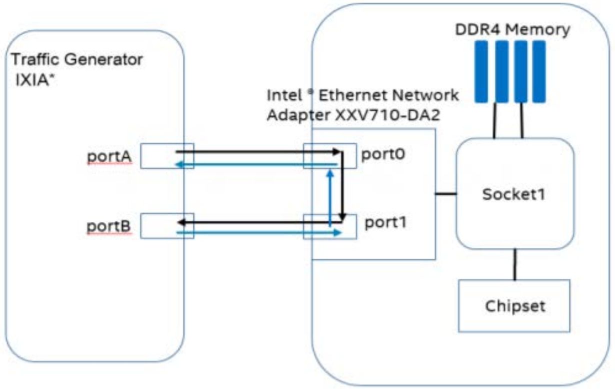
Performance

❑ 2 Cards


Intel ® Ethernet Network Adapter XXV710-DA2
DDR4 Memory
Traffic Generator
IXIA*
portA
port0
Socket1
portB
port1
Chipset
Intel ® Ethernet Network Adapter XXV710-DA2

Packet Size (Bytes)	Wire Speed (Mpps)	Packet Rate (Mpps)	%Wire Speed
64	37.2	35.63	95.78%
128	21.1	21.1	100%
256	11.3	11.3	100%

❑ 1 Cards



Packet Size (Bytes)	Wire Speed (Mpps)	Packet Rate (Mpps)	%Wire Speed
64	37.2	18.1	48.67%
128	21.1	17.26	81.74%
256	11.3	10.5	92.98%
512	5.87	5.71	97.27%
1024	3.03	2.91	97.32%

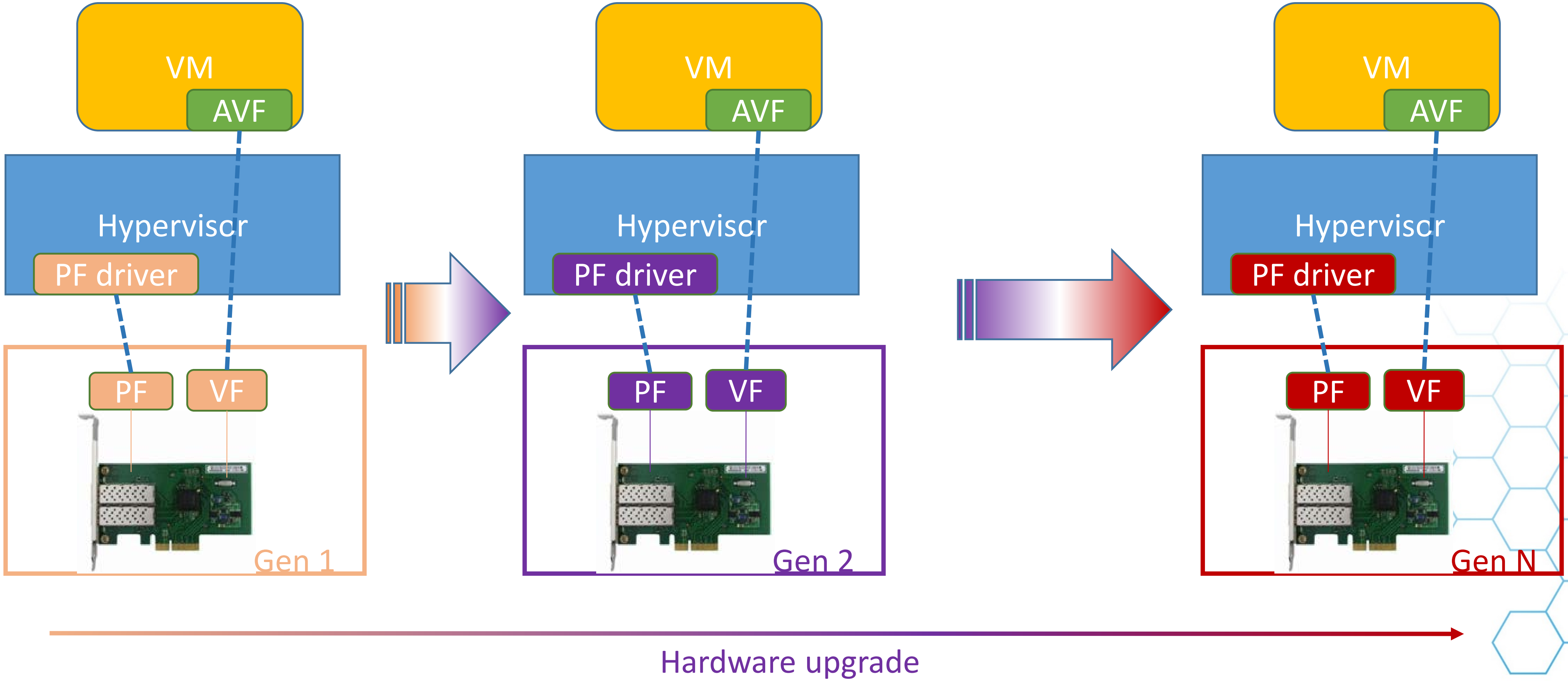
http://fast.dpdk.org/doc/perf/DPDK_17_05_Intel_NIC_performance_report.pdf

Adaptive Virtual Function (AVF)

AVF -- Adaptive Virtual Function

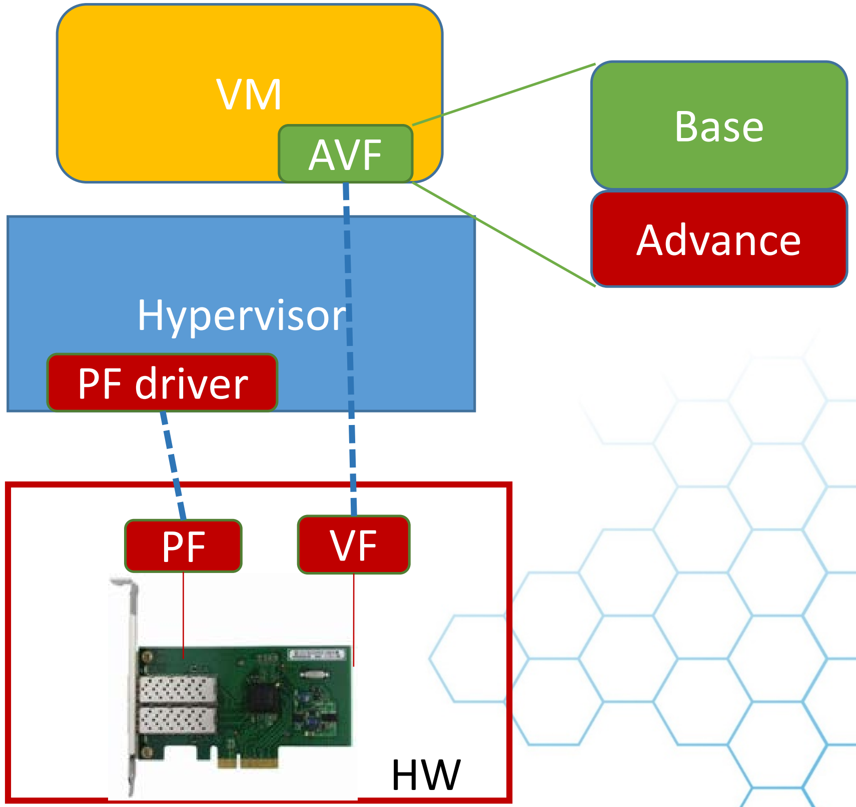
- **Needs:**
  - A **single** VF driver for all generations of Devices.
- **Solution:**
  - Adaptive Virtual Function
    - Base features
    - Negotiated Advanced Features
- **Benefits:**
  - Existing VM Images will run on the new hardware with **no change**.
- **From:**
  - Intel® 700 series Ethernet Controller

AVF – HW upgrade



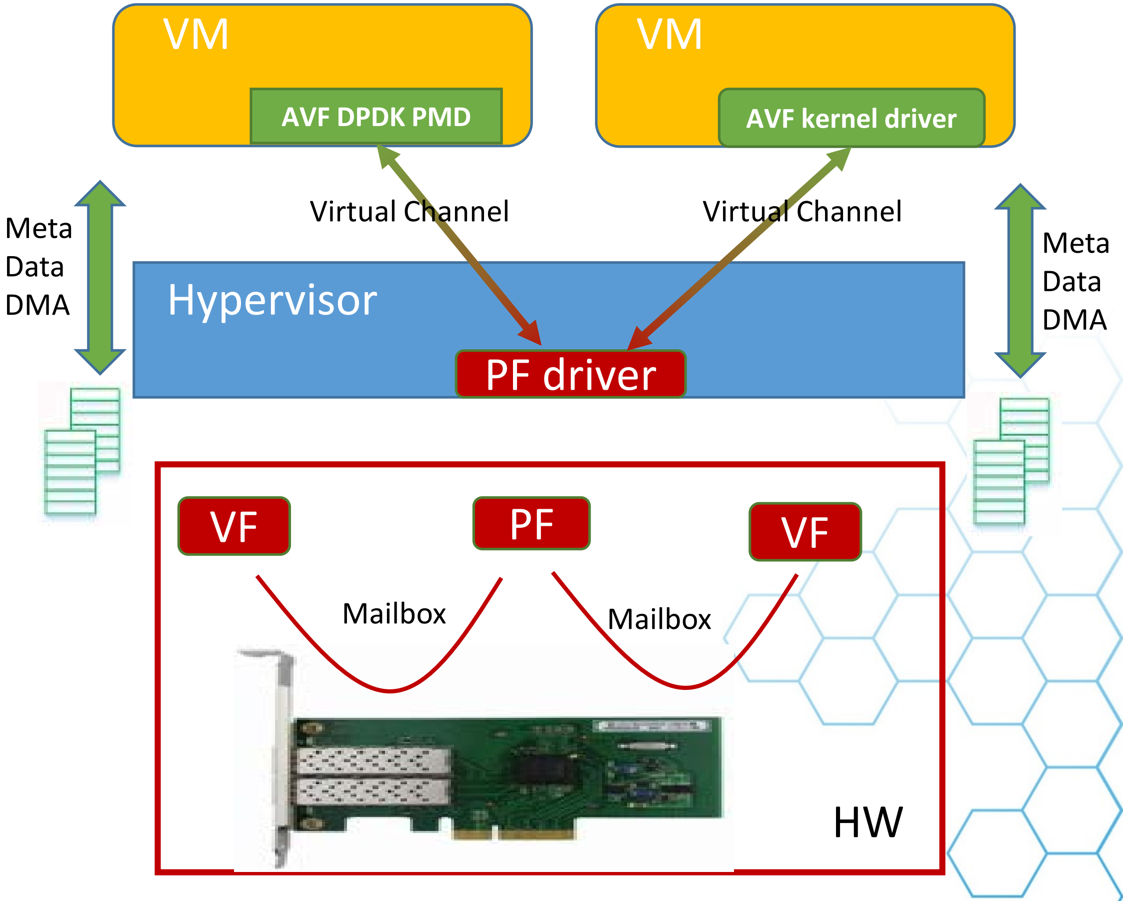
AVF -- Adaptive Virtual Function

- **Base mode supported**
  - Single device ID
  - Support for single level checksum and TSO offload
  - Multi-queue support
  - RSS
- **Advanced features**
  - Advanced feature introduced by new generation HW.
  - Negotiate with PF driver to expose.



Preserving in Hardware and Software

- **Preserving Base mode**
  - Fixed Minimum Register definition
  - A fixed Meta data format for DMA
  - A Hardware generic mailbox to talk to the PF
  - A Software defined Virtual channel layered on top of Hardware mailbox for expansion
- **Room for expansion**
  - Uncompromising on the base functionality.
  - A large range for hot path registers (Queue and Interrupt)
  - Expandable Virtual channel capability negotiation over the agreed upon communication channel between PF and VF.
  - More advanced features would be added with new drops of AVF driver if the underlying HW device supports.
  - Intel is working on the AVF specification.



Key Takeaways

- **25GbE speed, and better hardware capability**
- **Generic, flexible and configurable flow classification**
- **NFV enabled with VFD**
- **Good performance**
- **Adaptive VF driver for all Intel® NICs from 700 series**

End

- Helin Zhang, helin.zhang@intel.com
- Jingjing Wu, jingjing.wu@intel.com