Open Infrastructure FOUNDATION

Soumaya MSALLEM

Cloud Consultant at Red Hat

OpenInfra DAYS

Mexico

Accelerating NFV Data Plane with SR-IOV and DPDK

NFV (Network Functions Virtualization)

What is NFV

Is a **concept** that aims to transform **dedicated physical network hardware** to **virtual network appliances** running on standard, resilient, scalable hardware

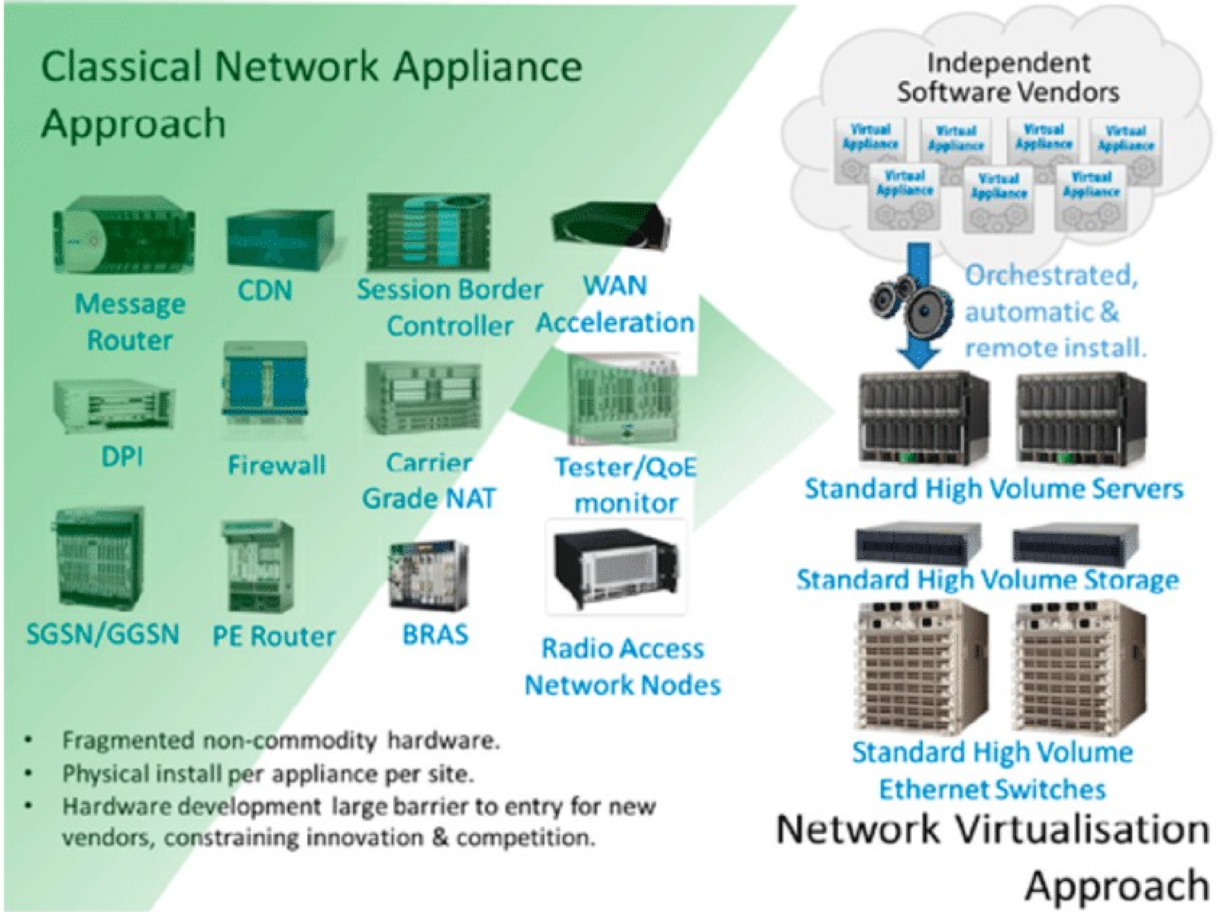


Figure 1 : Network Function Virtualiation approach

What is VNF

A **Software**, that performs a specific network function, such as routing or firewalling

NFV vs VNF

NFV refers to a global **concept** as a framework for running software-defined network functions;

VNF is the **implementation** of a network function by using a software decoupled from the underlying hardware infrastructure

Benefits of NFV

- Cost effectiveness
- Power consumption reducing
- Time saving
- Scalability, elasticity and dynamicity

Enablers for NFV

Several recent technology developments make the goals of NFV achievable.

We will explore :

- SR-IOV
- DPDK

SR-IOV (Single ROOT I/O Virtualization)

SR-IOV

SR-IOV defines a standardized mechanism to virtualize a PCIe device

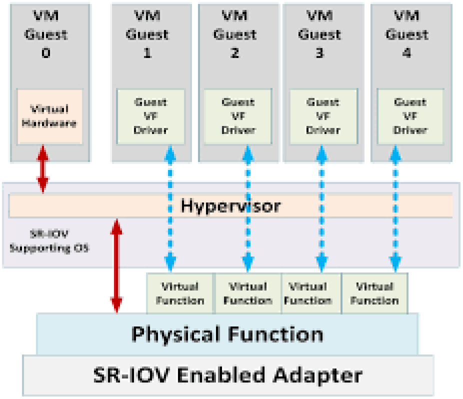


Figure 2 : SR-IOV Overview

SR-IOV

This is achieved through the introduction of two PCIe functions:

- Physical Function (PF)
- Virtual Function (VF)

SR-IOV

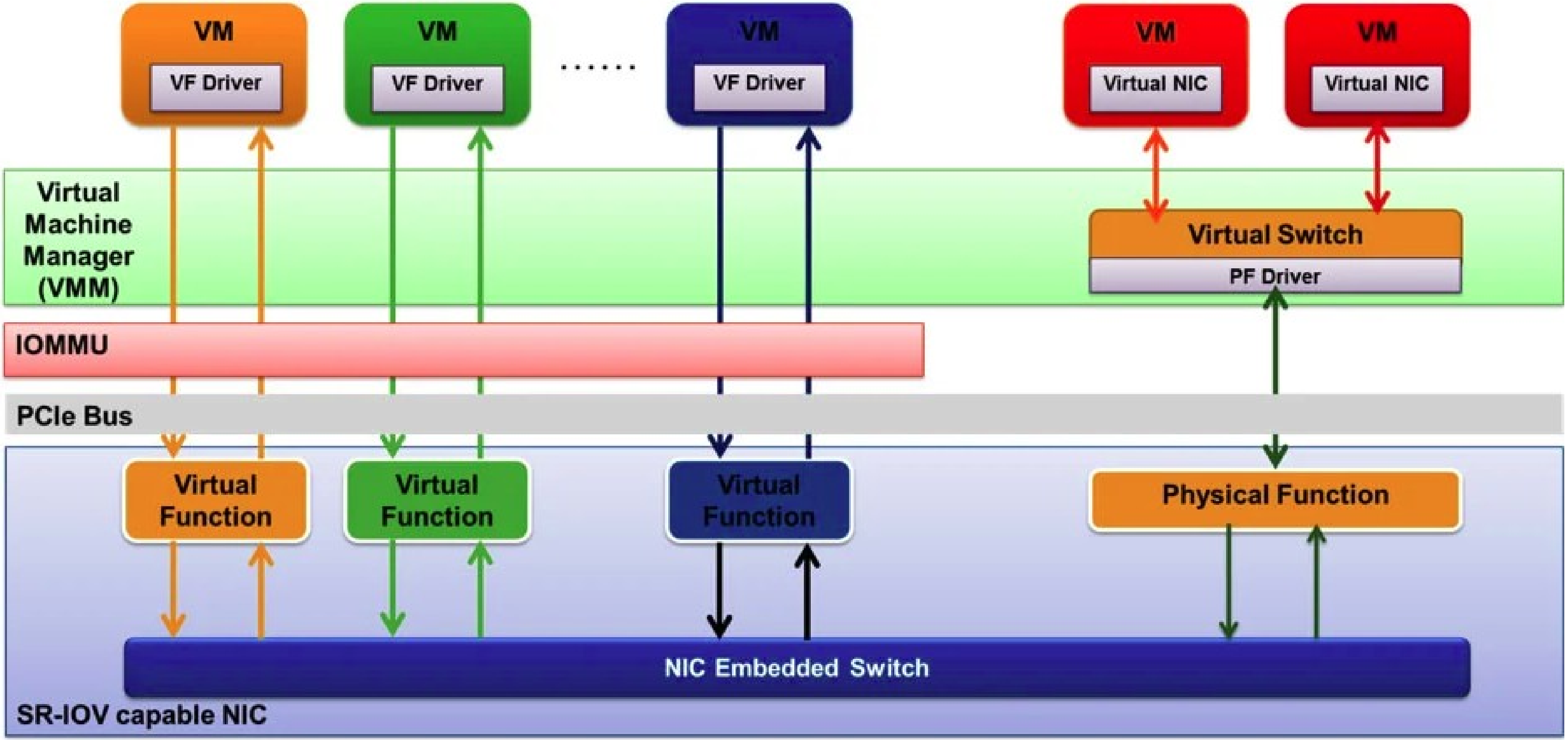
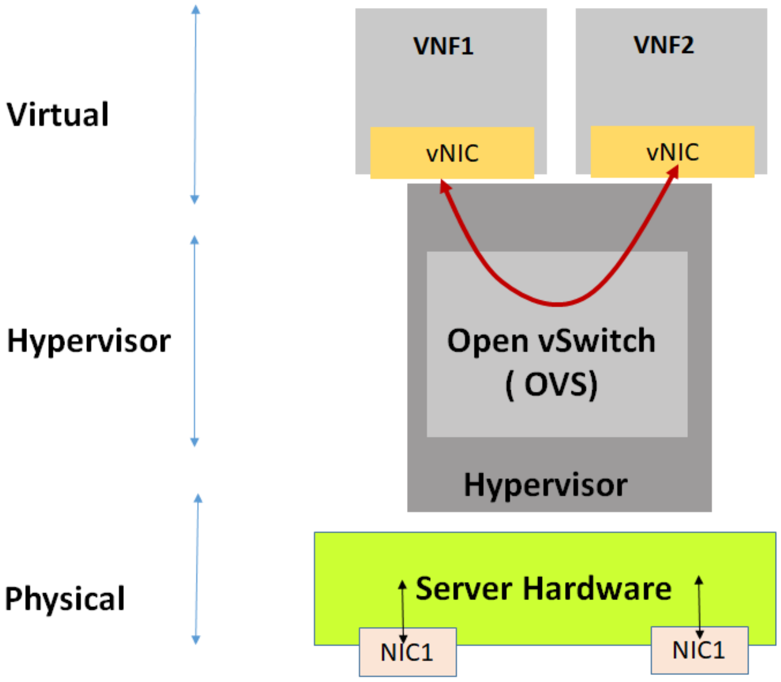
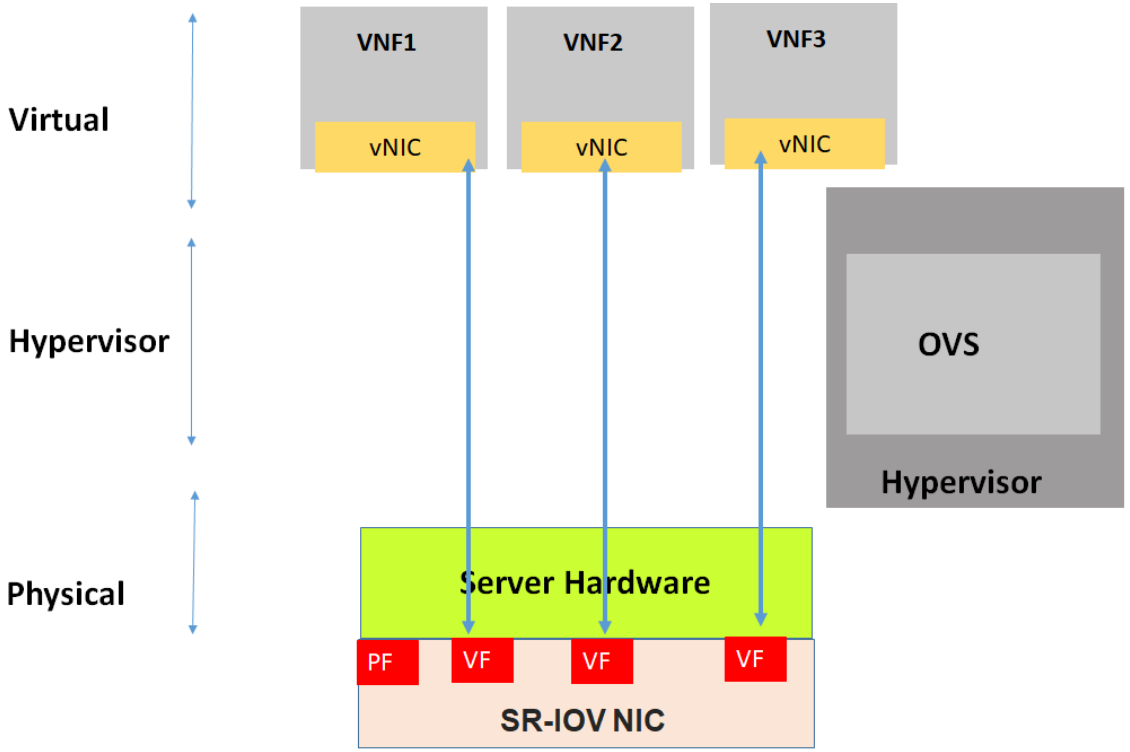


Figure 3 : SR-IOV Architecture

Virtualization vs SR-IOV



Without SR-IOV



With SR-IOV

Figure 4 : Comparison between Traditional Virtualization and SR-IOV

SR-IOV Benefits

- Low latency, increased network throughput and near-line wire speed are achieved
- extending the capacity of a device and lowering hardware costs

SR-IOV Support

SR-IOV is now supported by most Hypervisors :

- KVM
- VMware ESXI
- Microsoft Hyper-V

Hardware Considerations for Implementing SR-IOV

The **physical host** must

- Have a compatible processor
- Support I/O memory management unit (IOMMU), and have IOMMU enabled in the BIOS
- Support SR-IOV, and have SR-IOV enabled in the BIOS

Hardware Considerations for Implementing SR-IOV

The **physical NIC** must

- Be supported for use with the host and SR-IOV
- Have SR-IOV enabled in the firmware

Hardware Considerations for Implementing SR-IOV

PF driver for the physical NIC

A needed driver, compatible with Hypervisor type and release, must be installed on the physical host

Hardware Considerations for Implementing SR-IOV

Guest OS must

Be supported by the NIC on the installed Hypervisor release

Hardware Considerations for Implementing SR-IOV

VF driver in the guest OS must

- Be compatible with the NIC
- Be supported on the guest OS release
- Be installed on the Operating System

SR-IOV in OpenStack

SR-IOV was first introduced in the OpenStack Juno release

SR-IOV in OpenStack

The following manufacturers are known to work

- Intel
- Mellanox
- QLogic
- Broadcom

Enable SR-IOV in OpenStack

Compute nodes

Create Virtual Functions

```
echo '8' > /sys/class/net/nic-name/device/sriov_numvfs
```

Enable SR-IOV in OpenStack

```
# lspci | grep Ethernet
82:00.0 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
82:00.1 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
82:10.0 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:10.2 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:10.4 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:10.6 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:11.0 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:11.2 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:11.4 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01)
82:11.6 Ethernet controller: Intel Corporation 82599 Ethernet Controller Virtual Function (rev 01
```

Enable SR-IOV in OpenStack

Compute nodes

Edit the nova.conf file

```
[pci]
passthrough_whitelist = { "devname": "nic-name", "physical_network":
"network-name"}
```

Enable SR-IOV in OpenStack

Controller nodes

Edit the ml2_conf.ini file

```
[ml2]
mechanism_drivers = openvswitch,sriovnicswitch
```

Enable SR-IOV in OpenStack

Controller nodes

Configure nova-scheduler

```
[filter_scheduler]
enabled_filters = AvailabilityZoneFilter, ComputeFilter,
ComputeCapabilitiesFilter, ImagePropertiesFilter,
ServerGroupAntiAffinityFilter, ServerGroupAffinityFilter,
PciPassthroughFilter
```

Enable SR-IOV in OpenStack

Compute nodes

- Install the SR-IOV agent
- Edit the sriov_agent.ini file

```
[sriov_nic]
physical_device_mappings = net-name:nic-name
```

- Run the SR-IOV agent

DPDK (Data Plane Development Kit)

Why DPDK ?

By default, in Netwoking, Linux uses Kernel to process network packets.

This puts pressure on kernel to process packets faster as the **NICs speeds are increasing at fast**

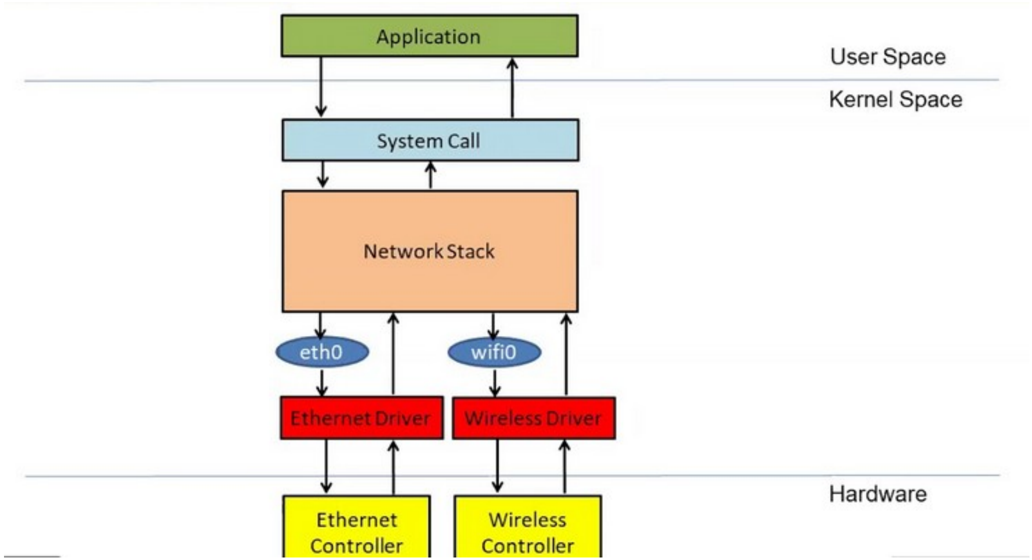


Figure 5 : Networking Stack

What is DPDK

DPDK is a set of **libraries** and user-space **drivers** for fast packet processing, enabling applications to perform their own packet processing directly to the NIC.

DPDK

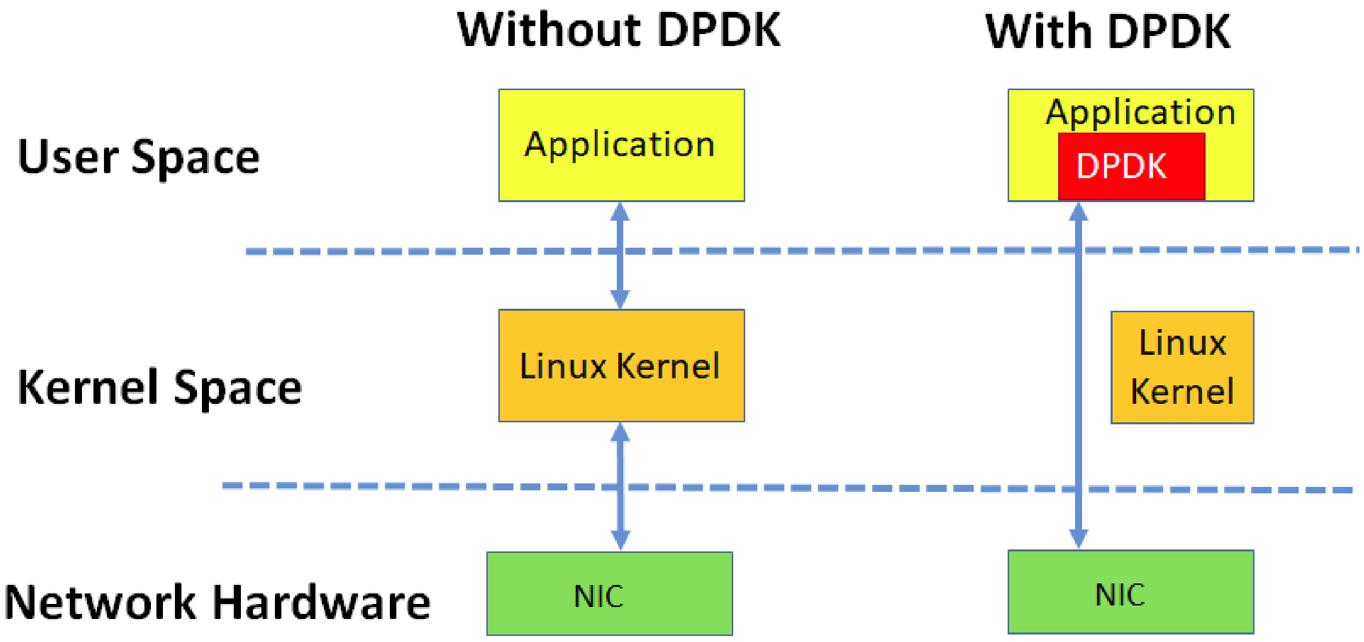


Figure 6 : Linux Networking Without and with DPDK

DPDK with OVS

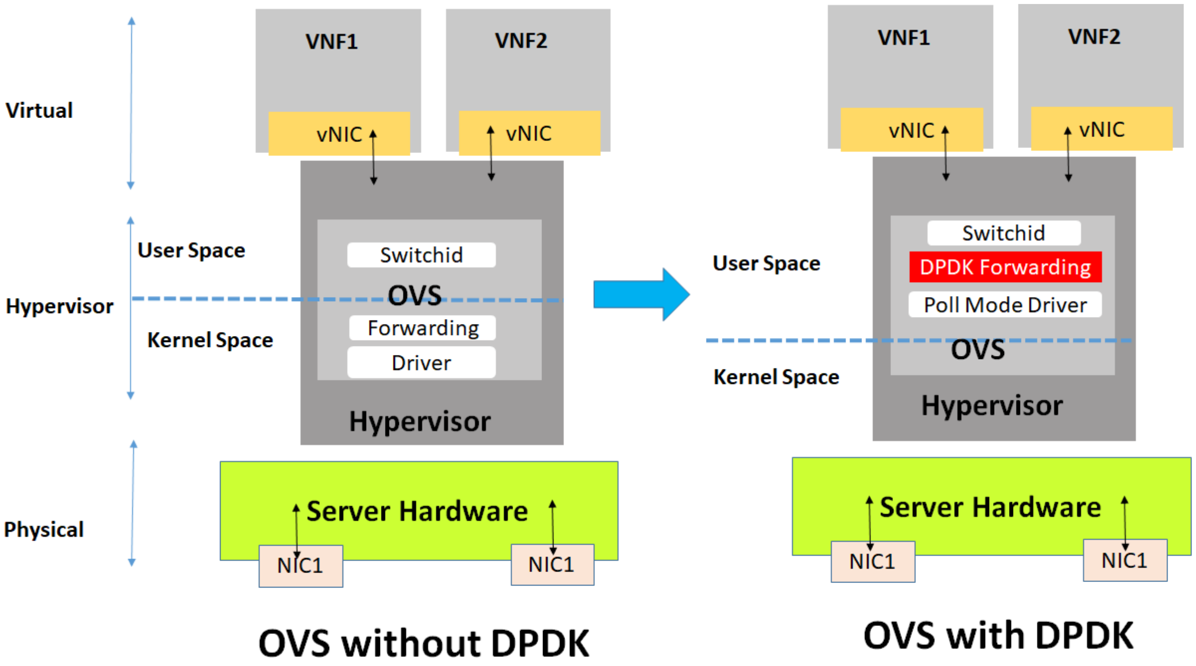


Figure 7 : Comparison between OVS without and with DPDK

DPDK (OVS + VNF)

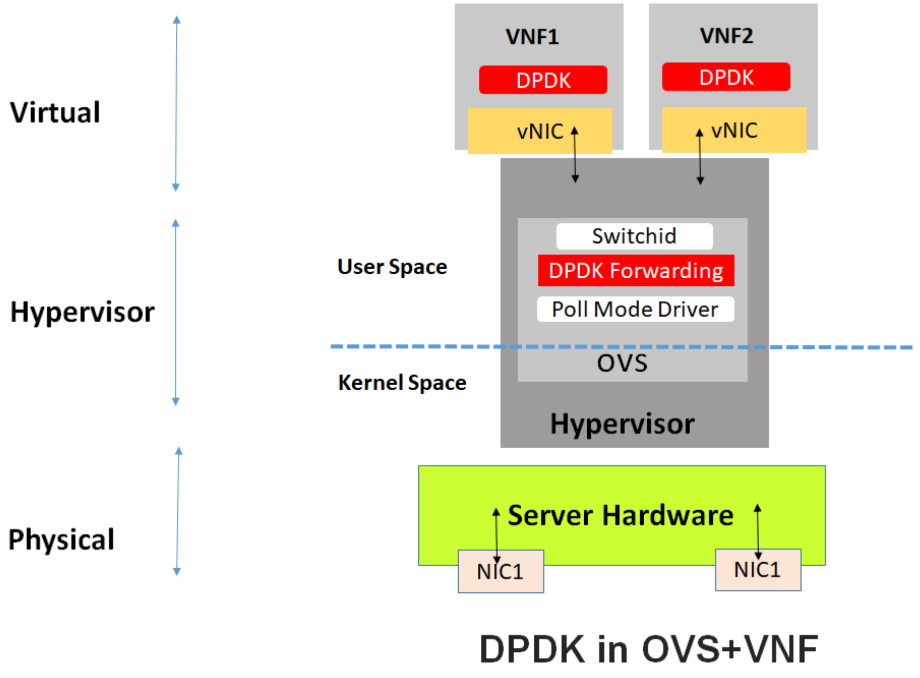


Figure 8 : DPDK integrated in OVS and in VNF

DPDK Support

Supports all major CPU architectures and NICs

DPDK in OpenStack

OVS-DPDK, DPDK bundled with OVS, can be used to provide high-performance networking between instances on OpenStack compute nodes

DPDK Benefits

Move performance-sensitive applications like the backbone for mobile networks and voice to the cloud.

Key enabler technology for NFV

4G and Vitualization

Telecom **4G** architectures use **VNFs,** typically running on Virtual Machines
Those Virtual Machines are deployed, in most of the cases, in a **Cloud** platform, like **OpenStack**

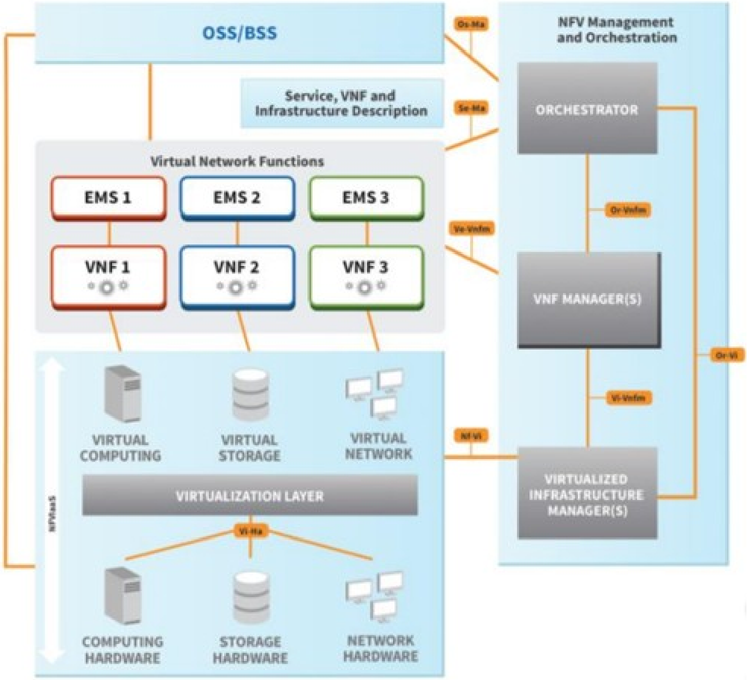


Figure 9 : VNF in 4G Context

VMs OUT, Containers IN

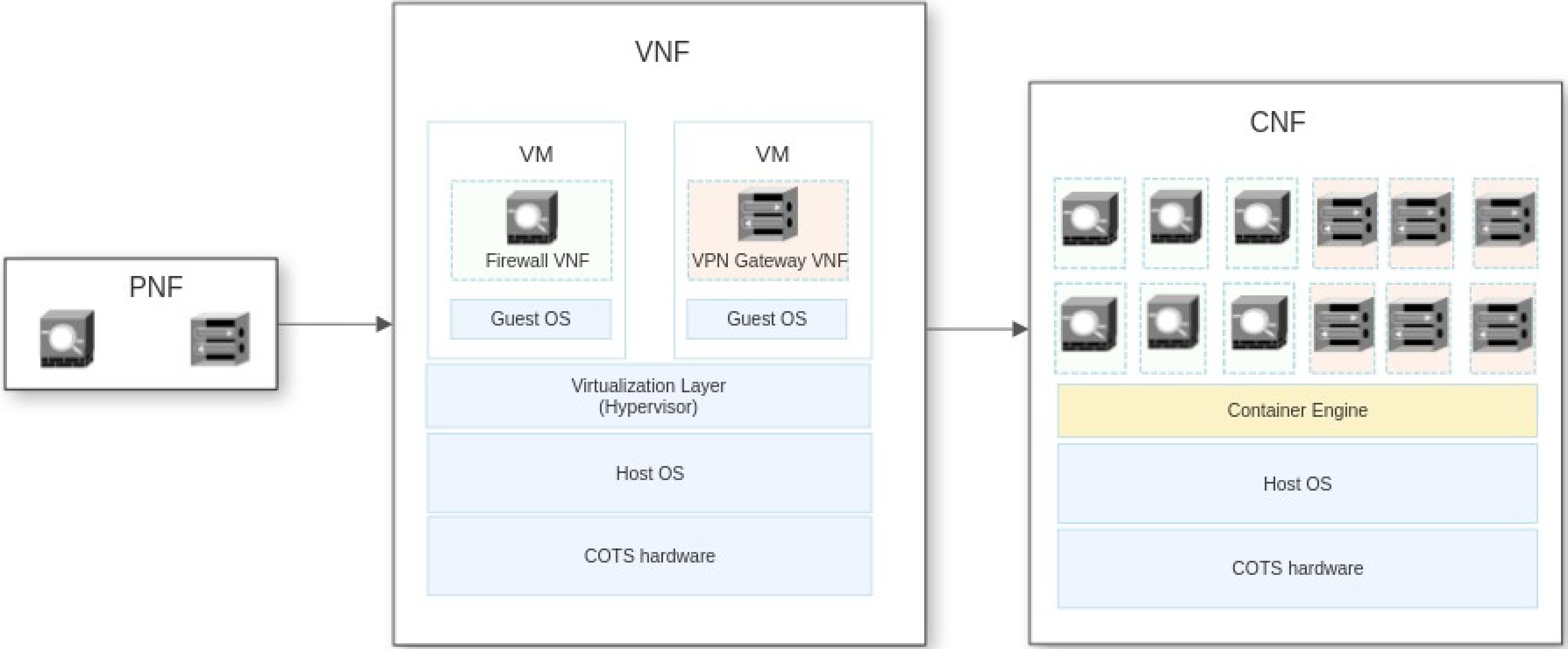


Figure 10 : VM-based VNFs transition to Container-based Network Functions CNFs

5G and Kubernetes

The 5G architecture, in most cases, is based on a **Cloud Native** design that leverages **Kubernetes** as the orchestrator that provides automated deployment and lifecycle management of **CNFs** deployed on **pods**

Benefits of transition to CNFs

CNFs offer

- Autoscaling
- Support for DevOps
- Incredible fault tolerance and fast restart
- Monitoring and reporting

Accelerating the 5G Dataplane with SR-IOV and DPDK

5G use cases need direct connectivity between CNFs, running on Kubernetes pods, and physical NICs

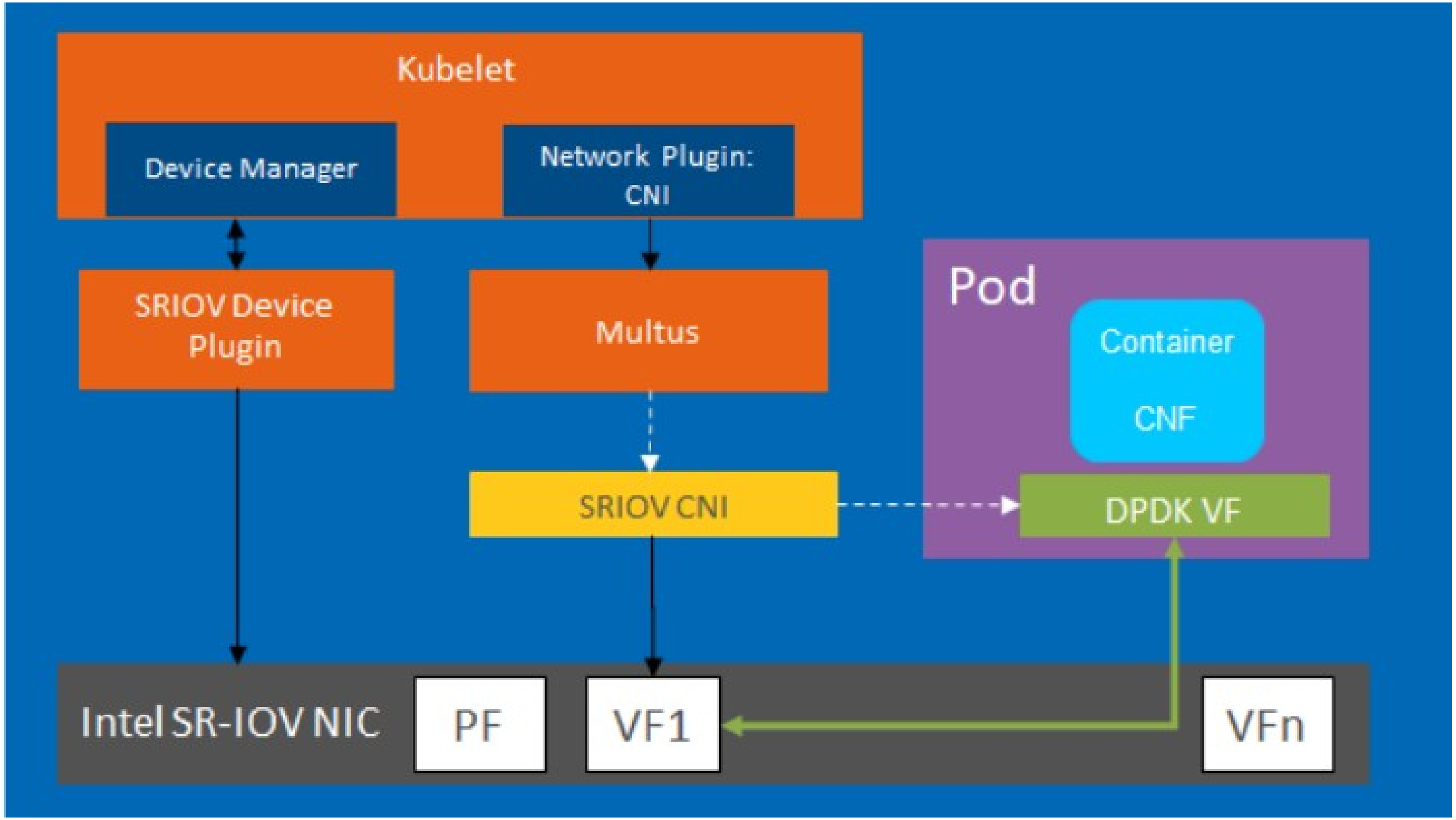
→ **SR-IOV**

Accelerating the 5G Dataplane with SR-IOV and DPDK

To bypasses the Linux Kernel so that data IO is sent from the user space in the pod to NIC

→ **DPDK CNF**

Accelerating the 5G Dataplane with SR-IOV and DPDK



Conclusion

4G and 5G applications are all be about low latency and fast data processing at high throughput.

These include not just the wireless core, radio applications but also custom workloads such as video streaming, VoIP, etc.

Along with SR-IOV, DPDK is a high-performance option that enables the industry to move latency-sensitive applications to the cloud.

Thank you so much for being here with us :)

OpenInfra DAYS

Mexico