DPDK vs SR-IOV for NFV? – Why a wrong decision can impact performance!

35 Comments / NFV, SDN / By Faisal Khan

It is not easy to settle the debate for DPDK vs SR-IOV-the technologies used to optimize packet processing in NFV servers.

For one, you will find supporters on both sides with their claims and arguments.

However although both are used to increase the packet processing performance in servers, the decision on which one is better comes down to

So a wrong decision on DPDK vs SR-IOV can really impact the throughput performance as you will see towards the conclusion of the article.

To understand why design matters, it is a must to understand the technologies, starting from how Linux processes packets.

In particular, this article attempts to answer the following questions!

1. What is DPDK
2. What is SR-IOV
3. How DPDK is different than SR-IOV
4. What are the right use cases for both and how to position them properly?
5. How DPDK/SR-IOV affects throughput performance.

I recommend that you start from the beginning until the end in order to understand the conclusion in a better way.

What is DPDK?

DPDK stands for Data Plan Development Kit.

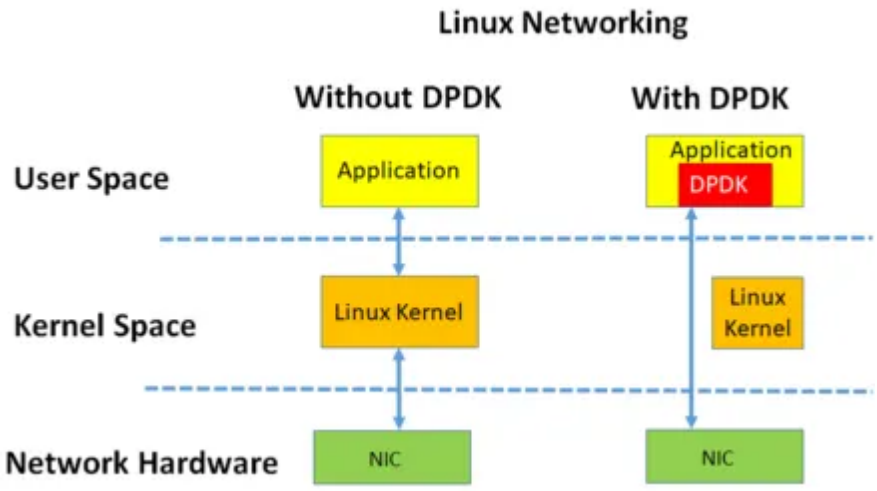
In order to understand DPDK , we should know how Linux handles the networking part

By default Linux uses kernel to process packets, this puts pressure on kernel to process packets faster as the NICs (Network Interface Card) speeds are increasing at fast.

There have been many techniques to bypass kernel to achieve packet efficiency. This involves processing packets in the userspace instead of kernel space. DPDK is one such technology.

> **User space versus kernel space in Linux?**
> **Kernel space** is where the kernel (i.e., the core of the operating system) runs and provides its *services*. It sets things up so separate user processes see and manipulate only their own memory space.
> ***User space*** is that portion of system memory in which *user processes* run . Kernel space can be accessed by user processes only through the use of *system calls*.

Let's see how Linux networking uses kernel space:



For normal packet processing, packets from NIC are pushed to Linux kernel before reaching the application.

However, the introduction of DPDK (Data Plane Developer Kit), changes the landscape, as the application can talk directly to the NIC completely bypassing the Linux kernel.

Indeed fast switching, isn't it?

Without DPDK, packet processing is through the kernel network stack which is interrupt-driven. Each time NIC receives incoming packets, there is a kernel interrupt to process the packets and a context switch from kernel space to user space. This creates delay.

With the DPDK, there is no need for interrupts, as the processing happens in user space using Poll mode drivers. These poll mode drivers can poll data directly from NIC, thus provide fast switching by completely bypassing kernel space. This improves the throughput rate of data.

DPDK with OVS

Now after we know the basics of how Linux networking stack works and what is the role of DPDK, we turn our attention on how OVS (Open vSwitch) works with and without DPDK.

What is OVS (Open vSwitch)?
Open vSwitch is a production quality, multilayer virtual switch licensed under

the open source Apache 2.0 license. This runs as software in hypervisor and enables virtual networking of Virtual Machines.
Main components include:
Forwarding path: Datapath/Forwarding path is the main packet forwarding module of OVS, implemented in kernel space for high performance
Vswitchid is the main Open vSwitch userspace program

An OVS is shown as part of the VNF implementation. OVS sits in the hypervisor. Traffic can easily transfer from one VNF to another VNF through the OVS as shown

In fact, OVS was never designed to work in the telco workloads of NFV. The traditional web applications are not throughput intensive and OVS can get away with it.

Now let's try to dig deeper into how OVS processes traffic.

OVS, no matter how good it is, faces the same problem as the Linux networking stack discussed earlier. The forwarding plane of OVS is part of the kernel as shown below, therefore a potential bottleneck as the throughput speed increases.

Open vSwitch can be combined with DPDK for better performance, resulting in a DPDK-accelerated OVS (OVS+DPDK). The goal is to replace the standard OVS kernel forwarding path with a DPDK-based forwarding path, creating a user-space vSwitch on the host, which uses DPDK internally for its packet forwarding. This increases the performance of OVS switch as it is entirely

DPDK (OVS + VNF)

It is also possible to run DPDK in VNF instead of OVS. Here the application is taking advantage of DPDK, instead of standard Linux networking stack as described in the first section.

While this implementation can be combined with DPDK in OVS but this is another level of optimization. However, both are not dependent on one another and one can be implemented without the other.

SR-IOV

SR-IOV stands for “Single Root I/O Virtualization”. This takes the performance of the compute hardware to the next level.

The trick here is to avoid hypervisor altogether and have VNF access NIC directly, thus enabling almost line throughput.

But to understand this concept properly, let’s introduce an intermediate step, where hypervisor pass- through is possible even without using SR-IOV.

This is called PCI pass through. It is possible to present a complete NIC to the guest OS without using a hypervisor. The VM thinks that it is directly connected to NIC. As shown here there are two NIC cards and two of the VNFs, each has exclusive access to one of the NIC cards.

However the downside: As the two NICs below are occupied exclusively by the VNF1 and VNF3. And there is no third dedicated NIC, the VNF2 below is left without any access.

SR-IOV solves exactly this issue:

The SR-IOV specification defines a standardized mechanism to virtualize PCIe devices. This mechanism can virtualize a single PCIe Ethernet controller to appear as multiple PCIe devices.

By creating virtual slices of PCIe devices, each virtual slice can be assigned to a single VM/VNF thereby eliminating the issue that happened because of limited NICs

Multiple Virtual Functions (VFs) are created on a shared NIC. These virtual slices are created and presented to the VNFs.

(The PF stands for Physical function, This is the physical function that supports SR-IOV)

This can be further coupled with DPDK as part of VNF, thus taking combined advantage of DPDK and SR-IOV.

When to use DPDK and/or SR-IOV

The earlier discussion shows two clear cases. One using a pure DPDK solution without SR-IOV and the other based on SR-IOV. (while there could be a mix of two in which SR-IOV can be combined with DPDK) The earlier uses OVS and the later does not need OVS. For understanding the positioning of DPDK vs SR-IOV, we will use just these two cases.

On the face of it, it may appear that SR-IOV is a better solution as it uses hardware-based switching and not constrained by the OVS that is a purely software-based solution. However, this is not as simple as that.

To understand there positioning, we should understand what is East-West vs North-South traffic in Datacenters.

> **What is East-West and North-South Traffic in Data Center?**
> "East-West" traffic refers to traffic within a data center — i.e. server to server traffic. "North-south" is between the server and outside the data center. Over the past few years, the East-West traffic has increased versus North-South traffic owing to increase of virtualization and the concentration of services inside a data center thus requiring the traffic to stay inside the data center (aka service chaining) instead of going out.

There is a good study done by intel on DPDK vs SR-IOV; they found out two different scenarios where one is better than the other.

if Traffic is East-West, DPDK wins against SR-IOV

In a situation where the traffic is East-West within the same server (and I repeat same server), DPDK wins against SR-IOV. The situation is shown in the diagram below.

This is clear from this test report of Intel study as shown below the throughput comparison

It is very simple to understand this: If traffic is routed/switched within the server and not going to the NIC. There is NO advantage of bringing SR-IOV. Rather SR-IOV can become a bottle neck (Traffic path can become long and NIC resources utilized) so better to route the traffic within the server using DPDK.

If traffic is North-South, SR-IOV wins against DPDK

In a scenario where traffic in North-South (also including traffic that is East-West but from one server to another server), SR-IOV wins against DPDK. The correct label for this scenario would be the traffic going from one server to another server.

The following report from the Intel test report clearly shows that SR-IOV throughput wins in such case

It is also easy to interpret this as the traffic has to pass through the NIC anyway so why involve DPDK based OVS and create more bottlenecks. SR-IOV is a much better solution here

Conclusion with an Example

So lets summarize DPDK vs SR-IOV discussion

I will make it very easy. If traffic is switched within a server (VNFs are within the server), DPDK is better. If traffic is switched from one server to another server, SR-IOV performs better.

It is apparent thus that you should know your design and traffic flow. Making a wrong decision would definitely impact the performance in terms of low throughput as the graphs above show.

So let say you have a service chaining application for microservices within one server, DPDK is the solution for you. On the other hand, if you have a service chaining service, where applications reside on different servers, SR-IOV should be your selection. But don't forget that you can always combine SR-IOV with DPDK in VNF (not the DPDK in OVS case as explained above) to further optimize the SR-IOV based design.

What's your opinion here. Leave a comment below?

← Previous Post

35 thoughts on "DPDK vs SR-IOV for NFV? – Why a wrong decision can impact performance!"

BUDIHARTO
15 SEPTEMBER, 2019 AT 7:36 AM

thanks for the article, very informative. for real NE in telco, it most likely will need to use SR-IOV for high throughput reason and surely it will sit on top of multeple hosts/phy servers (for redundancies and capacity), but it comes with the cost: it not easy to do "live" migration without interrupting the service and less flexible as it require special mapping to phy NIC (compared to OVS). the logical choice will be limit SRIOV for telco "NE" and OVS for the rest (EMS, management etc.)

Reply

FAISAL KHAN
15 SEPTEMBER, 2019 AT 8:24 AM