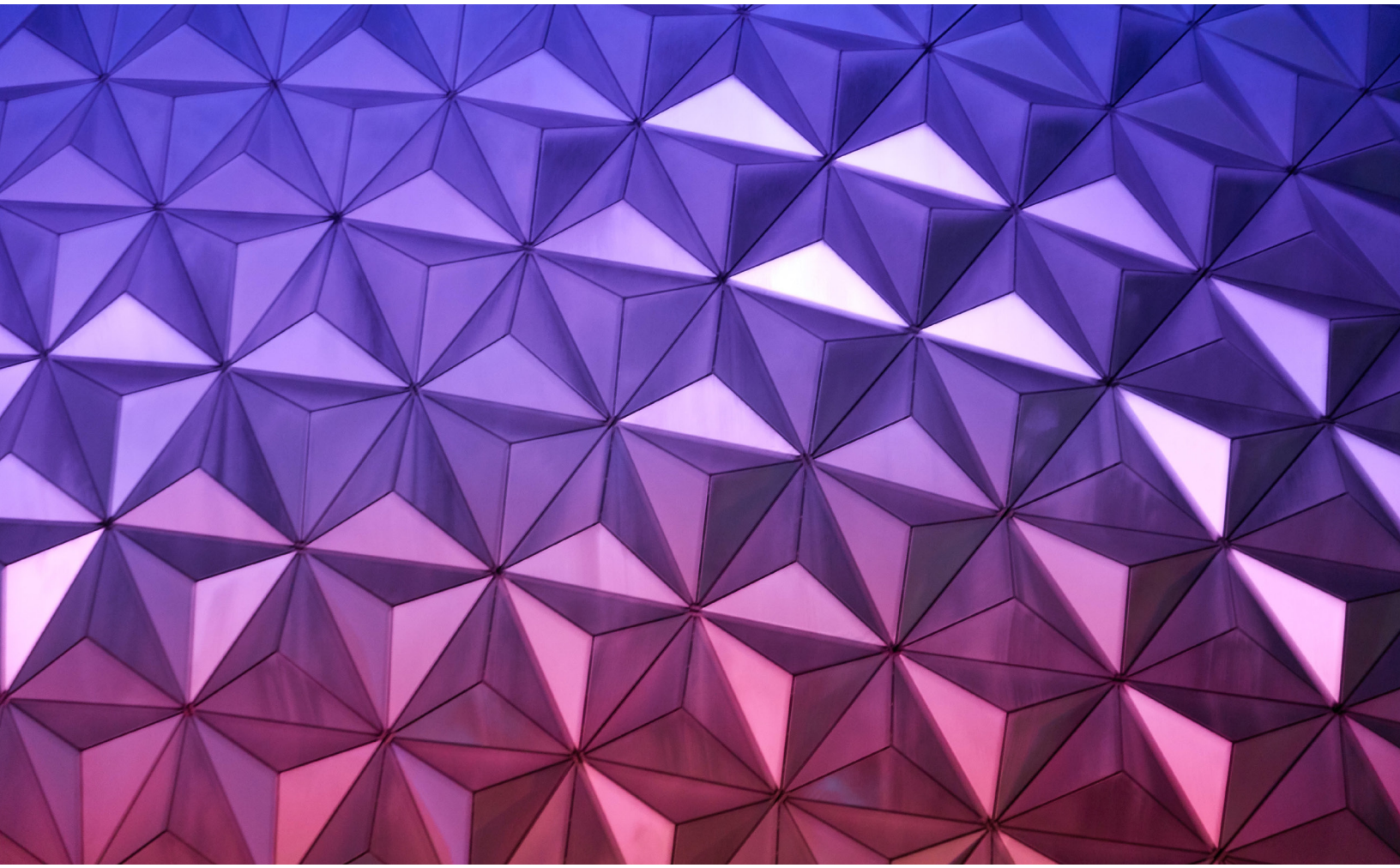
DPDK
DATA PLANE DEVELOPMENT KIT

Produced by AvidThink®



Myth-busting DPDK in 2020

Revealed: the past, present, and future of the most popular data plane development kit in the world.

THE LINUX FOUNDATION

Cover photo attribution: Robert Horvick from Unsplash

Introduction

The Data Plane Development Kit (DPDK) is a set of software libraries and drivers, running in userspace, that accelerate packet-processing workloads running on all major CPU architectures. Created by Intel about ten years ago and now housed as a project under the Linux Foundation, DPDK has been instrumental in driving the use of general-purpose CPUs in high-performance environments, from enterprise data centers to public clouds and particularly in telco networks.

This paper, produced by AvidThink, was commissioned by the Linux Foundation. It outlines the critical role played by DPDK in the evolution of networking infrastructure while dispelling several myths and misconceptions about the technology.

What is DPDK?

DPDK is an open-source project hosted by the Linux Foundation. To accelerate network I/O, DPDK allows incoming network packets to transition to userspace with no overhead for memory copying, where they are rapidly processed without the expense of context switching between user space and kernel space.

Architecturally, DPDK sits alongside the standard kernel network stack, accelerating specific networking functions where high throughput and low latency are critical, such as wireless core, wireless access, wireline infrastructure, routers, load balancers, firewalls, video streaming, voice over IP and more. Many popular Linux distributions, including those backed by Red Hat and Canonical, include DPDK support as part of their standard packaging.

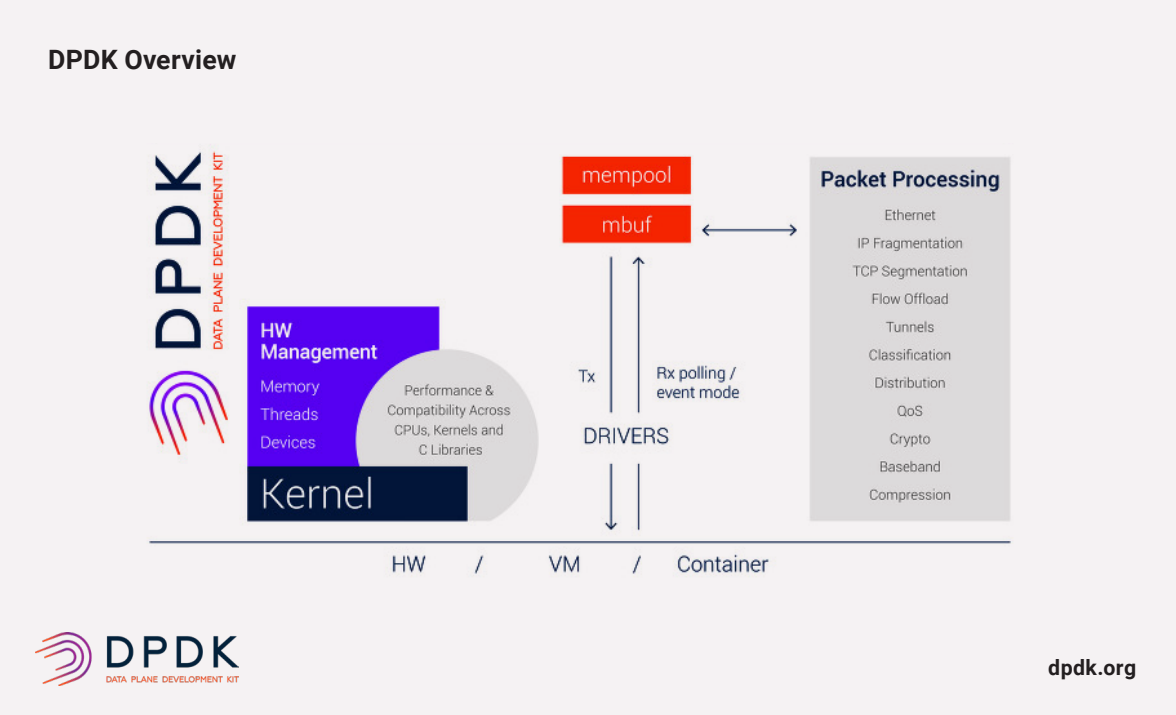


Figure 1

Why DPDK Was Needed — A Prehistory of DPDK

Traditionally networking equipment provided by vendors like Cisco, Ericsson, Huawei, Juniper, Nokia, and ZTE used Application-Specific Integrated Circuits (ASICs) to perform low-level data plane functions such as packet processing. In some cases, these ASICs were proprietary, while in others, they were standard products provided by silicon suppliers such as Broadcom or Marvell. In turn, ASICs were leveraged by proprietary software to implement a wide range of networking protocols supported by firewalls, routers, switches, base stations, and other networking devices. While these architectures delivered the throughput necessary for high-performance networking, the schedules for introducing new products were constrained by lengthy silicon development/debug cycles, and there was no opportunity for software portability between vendors.

By 2007, semiconductor companies like Intel, Cavium (now part of Marvell), Freescale (now part of NXP) and NetLogic (now part of Broadcom) were introducing standard multi-core processors with sufficient processing power to perform low-level packet processing functions at the cost and performance required to compete effectively with ASIC-based networking products. The problem was the software: while Linux was the default operating system for networking equipment, the Linux kernel contained a number of bottlenecks that prevented packets from being processed at high performance.

A solution was needed that would remove these performance constraints while maintaining compatibility with Linux applications. It would be suitable for packaging as a library present in Linux distributions for use on-demand and when managing diverse networking devices.

These goals were first achieved in 2010 when Intel introduced the initial version of DPDK targeted at the generation of Xeon processors based on the Nehalem microarchitecture. DPDK bypasses the Linux kernel, performing packet processing in user space to maximize networking performance. DPDK achieves this through the use of a Poll-Mode Driver (PMD) running in user space, that continually checks incoming packet queues to see if new data has arrived, achieving both high throughput and low latency (Figure 2). At this time, DPDK was provided via a .zip file, under an open-source Berkeley Software Distribution (BSD) license.

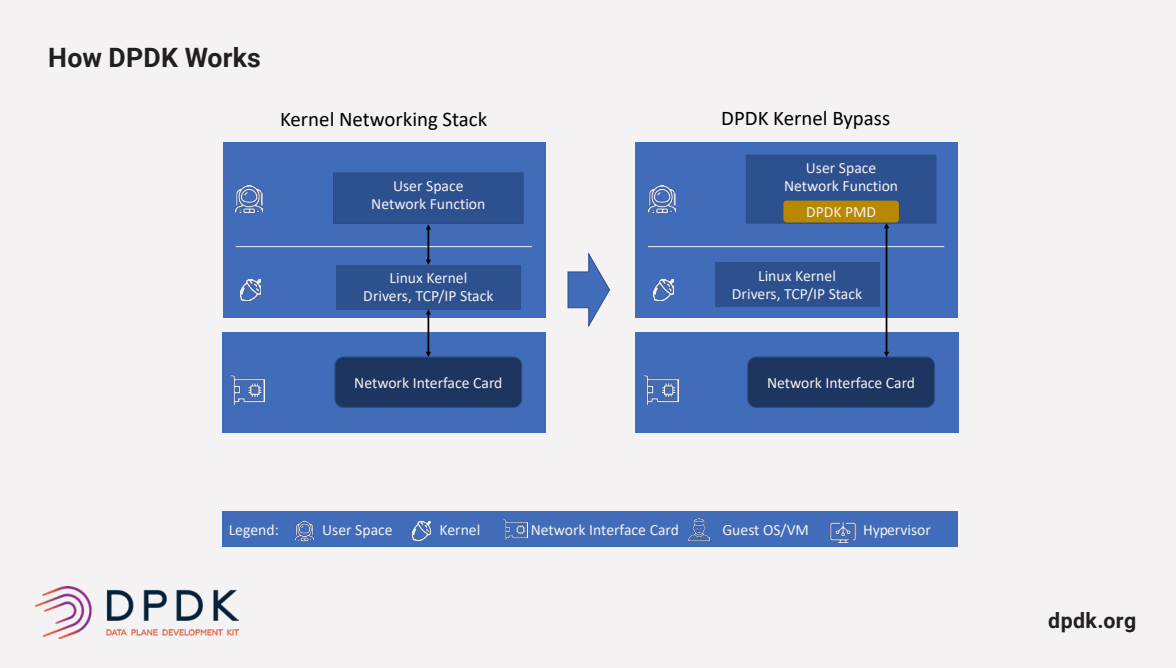


Figure 2

In 2013, networking software vendor 6WIND established the DPDK.org open-source community as a developer-focused resource, hosting libraries, drivers, documentation, a developers' mailing list, and a git repository. The emergence of DPDK.org facilitated an increase in the use of DPDK as well as significant growth in contributions to DPDK from an expanding pool of developers worldwide.

MYTH #1: Intel Controls DPDK

In the early days of DPDK, Intel did have an outsized influence on the project. However, as the project grew, so did the diversity of the community, with early contributions from device vendors like Chelsio and Mellanox. Contributions from the other members have become very significant. Intel continues to contribute efforts to improve DPDK, but other vendors representing instruction architectures such as Arm, Power, and Tilera (now part of Mellanox) have pushed DPDK outside of purely x86 support. As we've described, the governance structure under the Linux Foundation directs the future of the project and includes more participants than just Intel.

DPDK's Role in NFV and Beyond

Within the telecom industry, DPDK became a critical enabling technology as companies developed solutions to meet the goals of network functions virtualization (NFV). The NFV initiative started in 2012, when seven communications service providers (CSPs) agreed to collaborate under the auspices of ETSI: AT&T, BT, Deutsche Telekom, Orange, Telecom Italia, Telefonica, and Verizon. The list of member companies grew rapidly and now comprises over 800 organizations, including CSPs, vendors, public cloud providers, researchers, academia, and government entities.

A key objective of NFV was to enable the implementation of networking functions as virtualized software hosted on standard server platforms, as opposed to traditional networking equipment comprising proprietary software running on proprietary hardware. In order for this approach to be cost-effective, however, it was critical that the virtualized network functions (VNFs) were able to achieve adequate performance, both throughput and latency, when running on standard CPU architectures under the control of a hypervisor. The original ETSI NFV white paper, which outlined the goals for the initiative and the high-level architecture, identified DPDK as a critical enabling technology for NFV, along with multi-core processors and telco-grade virtualization platforms. The majority of telecom traffic is composed of small packets; without DPDK, the bottlenecks associated with processing these packets in the Linux kernel would have prevented VNFs from reaching the required performance level. However, by leveraging DPDK both in the VNFs themselves and in virtual switches (vSwitches), developers were able to achieve the high throughput and low latency that made the NFV concept viable and cost-effective as an alternative to fixed-function devices.

While the initial premise of NFV involved running virtualized workloads on standard servers, there is now a focus on accelerated platforms comprising Smart Network Interface Cards (SmartNICs) and Graphics Processing Units (GPUs), which enable significant reductions in host CPU resource requirements leading to major CAPEX and OPEX savings. Work is underway within the DPDK community to ensure that both SmartNICs and GPUs can be interfaced as accelerators for DPDK, enabling applications to leverage SmartNIC and GPU resources when those are available in the system.

MYTH #2: SmartNICs Will Kill DPDK

SmartNICs work collaboratively with DPDK. DPDK adds value to SmartNICs, whether they are based on ASICs, GPUs, Field-Programmable Gate Arrays (FPGAs), or Network Processing Units (NPUs). For instance, DPDK can provide an interface from the host system to SmartNICs, simplifying the interaction of various OSes with SmartNICs. There are also SmartNIC architectures with an embedded instance of OVS, that use DPDK on the embedded CPU for acceleration of the vSwitch. These SmartNICs provide flexible options for the bare metal host, VMs and hypervisors to interface with them.

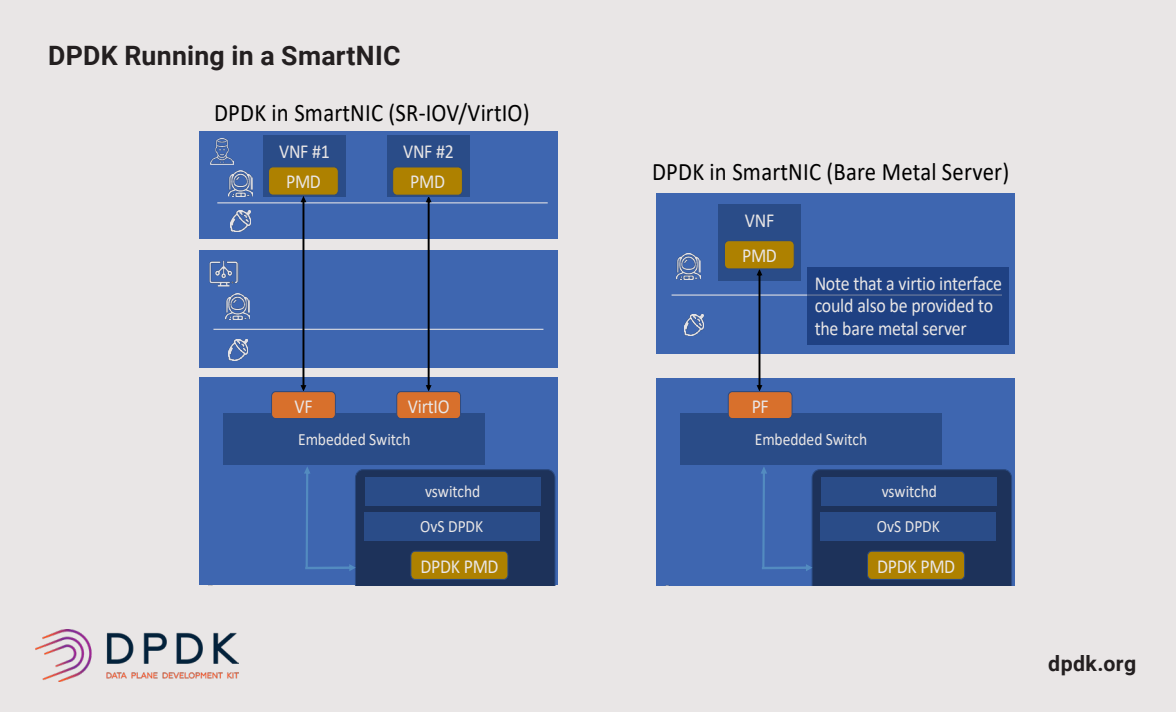


Figure 3

Linux Foundation and DPDK

In 2017, DPDK moved to the Linux Foundation, as a neutral home that promotes collaboration around a wide range of open-source technologies. The DPDK project now comprises one core software project, DPDK itself, as well as various smaller hosted projects that are closely related to DPDK. The core DPDK project code is managed through a number of git repositories, the most important of which is the “main repository” from which all DPDK releases are produced. As well as this, there are several “next” or staging repositories which are used to hold changes for specific areas before they get pulled into the master repository and a stable release repository. Each project has its maintainers and maintenance process.

The main DPDK code is available under the BSD license mentioned above, while sections related to the Linux kernel are licensed under the Gnu Public License (GPL).

Governance is driven through `two different boards, the governing board, and the technical board. The governing board deals

with budgets, marketing, lab resources, administration, legal, and licensing issues. The technical board is responsible for technical issues, including the approval of new subprojects, deprecating old subprojects, and resolution of technical disputes. These two boards are peers and work together to oversee the DPDK project.

The Linux Foundation organizes a number of DPDK-focused events around the world. DPDK Summits covering the latest developments to the DPDK framework and related projects have been held at various cities in China, Europe, India, and North America. A smaller number of DPDK Userspace events with a more technical agenda, including the roadmap, have been held in Europe.

DPDK's Community Growth

By late 2019, DPDK had received contributions from a diverse set of over 160 experts at more than 25 different organizations.

Independent of the technical contributors, the DPDK project has two categories of membership. As of late 2019, there were 10 Gold Members (Arm, AT&T, Ericsson, F5, Intel, Marvell, Mellanox, NXP, Red Hat, and ZTE) and three Silver Members (6WIND, Broadcom, and Huawei). The Gold Members have greater representation on the governing board.

A growing list of open-source projects leverages DPDK. These include ANS, BESS, Butterfly, DPVS, FD.io/VPP, FastClick, Lagopus, MoonGen, mTCP, OPNFV, OpenDataPlane, Open vSwitch, Packet-journey, Pktgen-dpdk, PcapPlusPlus, Ruru, Seastar, SPDK, TRex, WARP17, YANFF, and YAStack. As part of Open vSwitch (OvS), DPDK is widely deployed throughout cloud, enterprise, and telecom data centers worldwide.

MYTH #3: DPDK is a closed and exclusive community

It is true that the DPDK project was started by a small group of experts and gurus who were deeply involved with Linux kernel and device driver programming. However, the community has grown significantly and continues to do so. There are hundreds of different people involved in the project, from hardware developers and software developers to QA personnel, to documentation writers and end-consumers of the libraries. Growing beyond the few companies initially, there are now tens of companies involved in the evolution of the project. Anyone can become a contributor to DPDK by joining at https://www.dpdk.org/contribute/. All submissions are reviewed by expert maintainers who assimilate changes to ensure that quality standards are met, and best practices are followed.

Market Evolution

Most of the initial applications of DPDK were in telecom. As CSPs adopted network virtualization to reduce operational costs and accelerate the deployment of new services, they virtualized use cases that required high throughput and low latency, such as routers, firewalls, radio access network (RAN) and evolved packet core (EPC). Suppliers of virtualization platforms, VNFs, and applications have leveraged DPDK in their products in order to meet CSPs' performance goals in these cases.

As CSPs explore new edge-hosted applications such as video caching, surveillance, augmented reality (AR), assisted driving, retail, and industrial IoT, DPDK remains a critical technology for achieving aggressive performance targets.

Increasingly, DPDK is being applied to a range of enterprise and cloud use cases that have similar packet-processing performance requirements to the telecom applications where it was first used. In 2018, for example, VMware introduced a DPDK-based edge configuration of their NSX-T Data Center software-defined infrastructure. This version of NSX-T addresses

applications requiring high packet throughput with variable packet sizes as well as servers that support high-speed NICs with up to 100Gbps of north/south traffic. Typically, north-south flows have a wide variety of packet sizes and high packet-processing requirements, even though they constitute less than 20% of overall traffic. Analysis by Intel and VMware showed up to five times performance improvement through the use of DPDK with small packets (64 bytes) in this use case.

Several companies have adopted DPDK for financial applications where latency represents a significant competitive advantage. In high-frequency trading (HFT), for example, latency has a direct impact on the effectiveness of traders' algorithmic strategies and their ability to outperform competing traders. Information Week estimated that an advantage of one millisecond could be worth $100M a year to a major brokerage firm. DPDK is a key technology for vendors developing solutions in this market.

MYTH #4: DPDK is only for the telecommunications industry

With the arrival of NFV, DPDK became particularly important to the success of virtualizing network functions. However, DPDK is used across solutions deployed in enterprise data centers as well as in cloud service providers. In addition to the various use cases mentioned previously, other applications for DPDK include virtual switching, software-defined networking (SDN), data analytics, artificial intelligence (AI) inference, video transcoding, and online gaming. The big seven cloud providers — Alibaba Cloud, Amazon Web Services, Baidu Wangpan, Google Cloud Platform, Microsoft Azure, Tencent Cloud — all use DPDK in their offerings.

DPDK Proliferation Across Architectures

Intel Architecture, AMD, Arm, and Power

At the time Intel introduced DPDK in 2010 with support for x86 Xeon platforms, several other suppliers offered multi-core processors targeted for networking applications along with their own SDK libraries to enable high-performance packet processing. As examples, both Cavium and NetLogic provided processors based on the MIPS CPU architecture, while Freescale based theirs on the PowerPC architecture.

Over the next few years, all three of these companies introduced new multi-core processor families based on the Arm architecture. No doubt coincidentally, they were all eventually acquired as the semiconductor industry continued its never-ending evolution. Today, the vendor landscape for processors targeted at high-end networking applications has progressed to the point where the market-leading products are based either on the x86 architecture from Intel and AMD or on the Arm architecture from multiple suppliers, including Broadcom, Marvell, and NXP. The IBM Power architecture is used in processors for enterprise and cloud applications. All three of these architectures are now supported in DPDK.

As the originator of DPDK, a Gold Member of the Linux Foundation project and one of the leading contributor companies, Intel drives many of the innovations in DPDK, ensuring that highly optimized support for all relevant Intel products is available and well maintained. Besides the applicable Xeon processors, these products include network adapters such as the e1000, ixgbe, i40e, fm10k, and ipn3ke. DPDK also provides full support for Quick-Assist Technology (QAT) and software-based crypto implementations, such as those based on IPsec multi-buffer libraries.

DPDK runs on AMD's x86 processors, such as the EPYC Embedded 3000 family. The AXGBE poll mode driver (PMD) provides support for AMD's family of 10Gbps network adapters, which are integrated into EPYC SoCs.

The first open-source project to implement APIs for networking data plane processing on Arm-based SoCs was OpenDataPlane (ODP), launched by the Linaro Networking Group in 2013. Linaro is an engineering organization founded in 2010 by Arm, Freescale Semiconductor, IBM, Samsung, ST-Ericsson, and Texas Instruments to work on open-source software for Arm architecture platforms. In 2018, the Linaro Networking Group was disbanded, and governance of ODP was moved to the OpenFastPath (OFP) Foundation, which had been founded in 2015 by Arm, Enea, and Nokia. To enable compatibility with DPDK, OFP implemented DPDK support through the ODP-DPDK layer.

MYTH #5: DPDK is a pure software project that is anti-hardware

DPDK started out as a project to improve the packet-handling performance of a general-purpose OS on general-purpose CPUs. It has evolved to be much more than that. Today's DPDK is more of a framework to enable acceleration on Linux, other UNIX-type OSes, and Windows (see OS support listed at http://doc.dpdk.org/guides/). DPDK supports a wide range of hardware devices, from SmartNICs to crypto accelerators. It provides application APIs that can take advantage of hardware functions when available, but implements an all-software path when the hardware is unavailable. This ensures that applications dependent on DPDK will always run correctly, albeit with lesser performance, when unique hardware acceleration hooks aren't present.

As a result of both the ODP-OFP initiative, the Arm architecture has strong support within DPDK. Multiple Arm licensees contribute to DPDK, focused on ensuring that DPDK leverages all the capabilities of the architecture, which has been promoted by some CSPs and equipment providers as potentially more power-efficient than an x86 platform with the same performance. Support for the IBM Power architecture is available in DPDK, with the most recent DPDK release incorporating libraries for Power9. IBM and Canonical have collaborated to create a version of Ubuntu Server, orderable directly from IBM, that includes DPDK performance optimizations.

Given its breadth of architecture support, DPDK is increasingly viewed less as an accelerator for NFV and more as a Linux userspace framework that provides a uniform abstraction layer for core functions important to high-performance I/O and packet processing, including cryptographic functions.

With support for a wide range of hardware devices, including FPGAs, ASICs, and SmartNICs, DPDK leverages all available, supported resources to maximize overall networking performance at the system level.

MYTH #6: DPDK is only relevant to Linux

The initial implementation of DPDK was on Linux due to its pervasive use within data centers. However, DPDK today supports a broader range of operating systems, such as FreeBSD. In addition, DPDK is being ported to Windows.

Open vSwitch and Open Virtual Network

Open vSwitch, a Linux Foundation Networking project, is an open-source implementation of a distributed virtual multilayer switch available under the Apache 2.0 license. It enables massive network automation through programmatic extension, while still supporting standard management interfaces and protocols, e.g., NetFlow, sFlow, IPFIX, RSPAN, CLI, LACP, and 802.1ag. The Linux kernel implementation of OvS was merged into the kernel mainline in 2012.

Intel approached the OvS community to explore an architecture enhancement that would make DPDK an additional data plane for OvS, but with the community's focus on enterprise applications, the value proposition was unclear, and the overture failed. Based on the performance requirements of NFV use cases, Intel then worked with two telecom customers to create DPDK Accelerated Open vSwitch (OvS-DPDK) as a fork of OvS which leveraged DPDK to optimize the performance of OvS on traffic dominated by small packets, as required for telecom networking.

In 2013, Intel publicly announced OVDK as an open project with participation from external contributors. In 2014, they stated that the OVDK project had achieved the goal of demonstrating the desired performance acceleration and that OVDK would be discontinued in favor of integrating DPDK into the mainline OvS code base, committing to maintain this support within OvS.

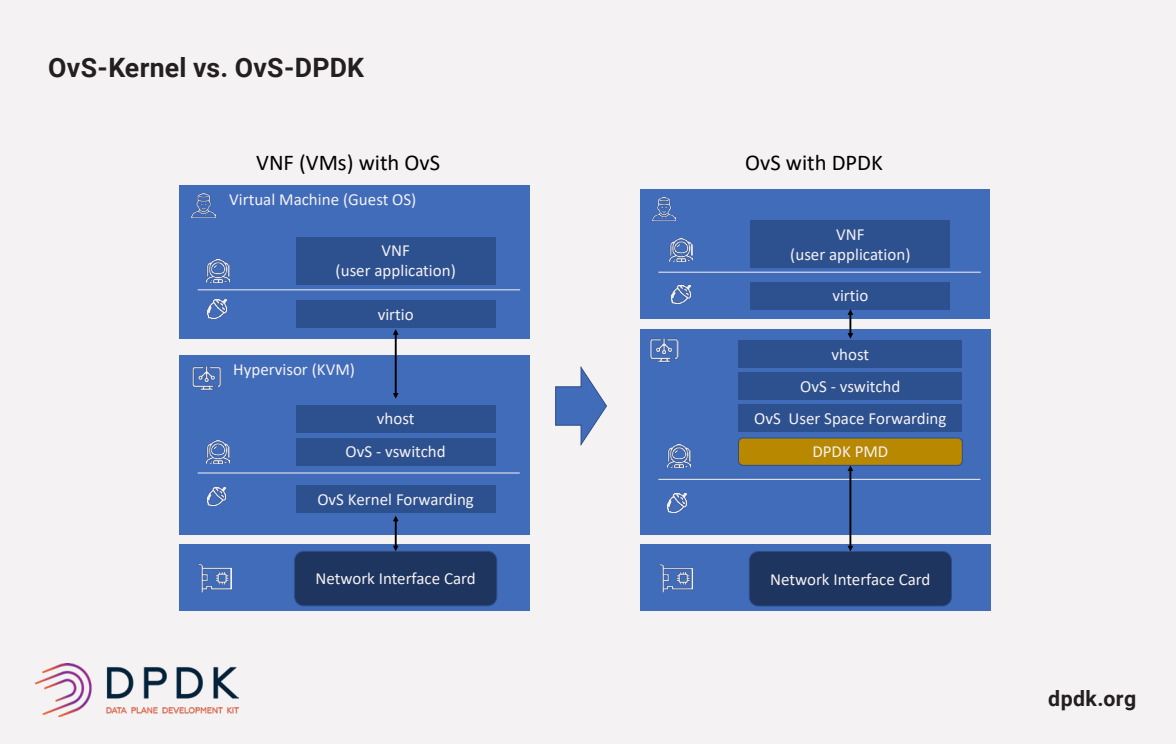


Figure 4

By 2015, OvS-DPDK had been merged into the mainline as a build option for OvS, thus providing the capability to implement a user space data plane in OvS (Figure 4). OvS-DPDK provides an increase greater than ten times in small packet throughput and much lower latencies. Several performance hot-spot areas inside OvS were also optimized using the DPDK packet processing libraries. Open Virtual Network (OVN) is an open-source project originally launched in 2015 by the OvS team at

Nicira (now part of VMware) as a system to support virtual network abstraction. OVN complements the capabilities of OvS, including the OvS-DPDK configuration, by adding native support for virtual network abstractions, such as virtual layer two and layer three overlays and security groups. Open-source bindings for OVN are available for a number of platforms, such as OpenStack and Kubernetes. OVN is the SDN platform used in a number of commercial products, including Red Hat Virtualization, Red Hat OpenStack, and Red Hat OpenShift. OvS-DPDK brings high packet-processing performance to deployments based on these products.

Today, for most NFV deployments, we see a combination of OvS-DPDK, along with DPDK enabled in the VNF itself (Figure 5). This provides strong networking performance all the way through (from the NIC to the application) while retaining the flexibility of having a vSwitch available to multiple VNFs that may communicate with each other (east-west traffic for service chaining). As we will see, DPDK can be used in numerous deployment architectures and combined with other methods for acceleration, including hardware-based approaches.

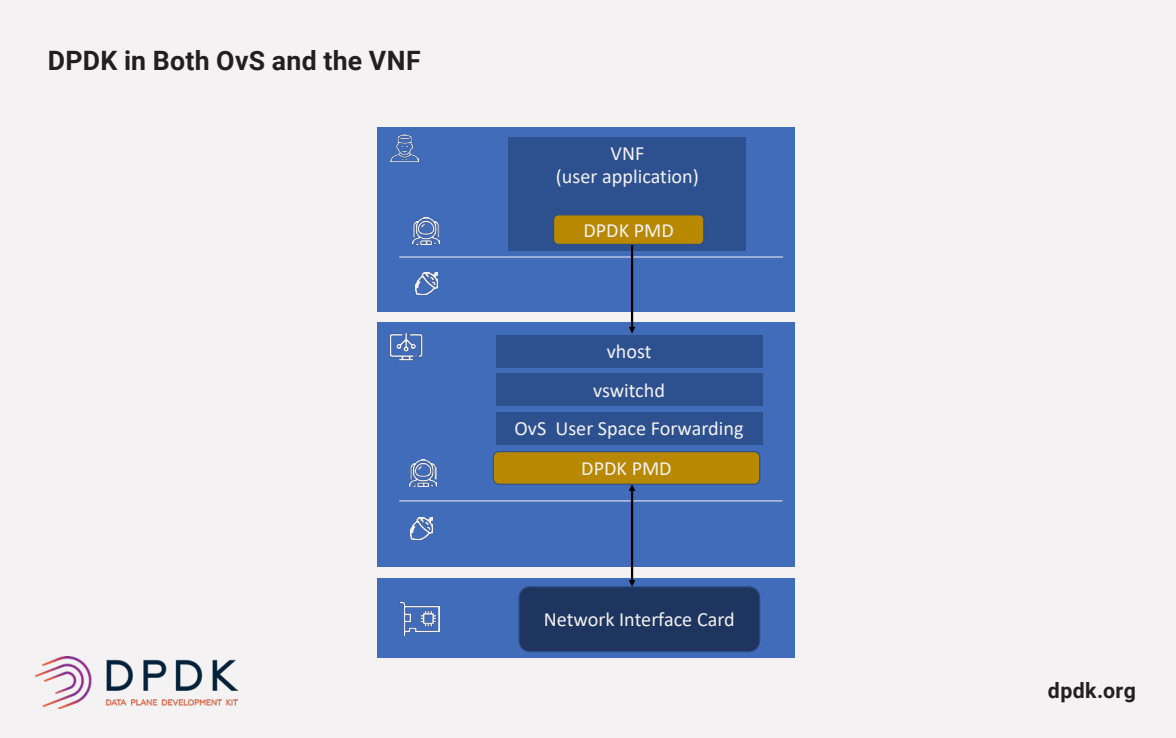


Figure 5

MYTH #7: DPDK is only interesting to hardware vendors

Yes, there are lots of hardware vendors involved in DPDK. However, there are just as many, if not more, service providers, system integrators, end consumers, and software developers involved in DPDK and other related projects.

Related Open-Source Projects and Standards

While DPDK is the most prominent of software acceleration toolkits, there are also other open-source projects and frameworks that overlap with or are adjacent to DPDK. These include Linux Foundation's Fast data – Input/Output (FD.io) Vector Processing Plane (VPP) project, as well as Express Data Path (XDP) from the Linux Foundation ioVisor project, along with Address Family XDP (AF_XDP) and virtual data path acceleration (vDPA). There is also the Snabb packet toolkit and the netmap project, which both take a similar approach of shuttling packets into user space as quickly as possible. While Snabb is still an active project, DPDK has superseded netmap because of its superior performance.

There are many ways to improve networking performance in Linux and other UNIX-style operating systems. These range from offloading functions like payload checksumming into NICs to full-blown TCP offload engines (TOEs). With the recent wave of NFV workloads, there's been renewed interest in the different methods, DPDK being just one. In the interest of brevity, we will take just a brief pass across the different techniques and provide readers with context as to where DPDK fits today and information on other popular and promising initiatives for network acceleration.

We'll start with hardware-assisted mechanisms, many of which are complementary to DPDK:

PCI Passthrough

PCI passthrough involves binding the VM guest to a specific PCI card as if the VM were a bare-metal system with full access and ownership of the NIC. Similar techniques can be used to bind a specific NIC to a container. This means there is no contention by other applications for the NIC, and the networking application has exclusive access. After packets get into the VM or container, they still have to be processed in either user or kernel space, but that's a separate consideration.

Single Root I/O Virtualization

Single root I/O virtualization (SR-IOV) is an extension to the PCI standard that involves creating virtual functions (VF) that can be treated as separate virtual PCI devices. Each of these VFs can be assigned to a VM or a container, and each VF has dedicated queues for incoming packets. Again, how packets are handled after they arrive is a separate decision. SR-IOV is often combined with DPDK in real-world deployments, the goal being to bypass the kernel for both the host and guest OSs. It can be used to provide direct access from user space in a VNF (VM) to the incoming packet queue on the virtual interface on the NIC, reducing latency and copies along the way (Figure 6). The same technique can be used with containers. The main issue with SR-IOV is that it requires a NIC that supports the capability, and each VF takes up physical resources on the NIC, so while theoretically, the VF limit is high, there may be practical memory limits on how many VFs can reasonably be supported.

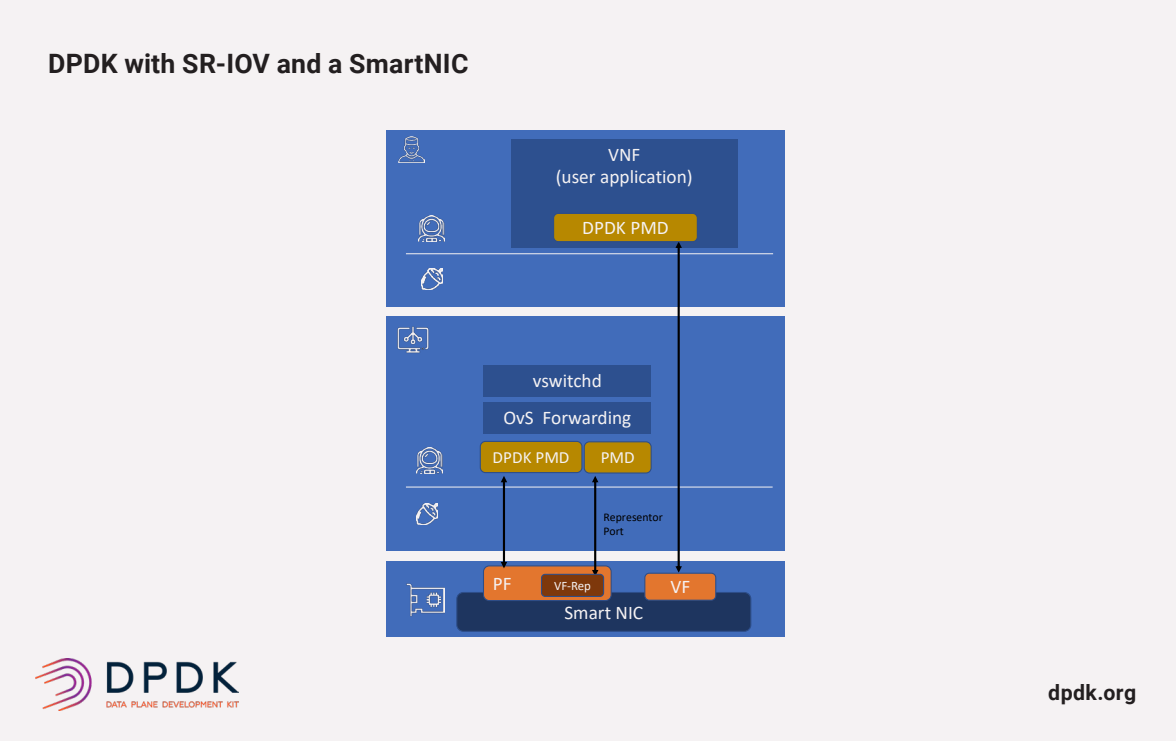


Figure 6

In addition to hardware approaches like PCI-passthrough and SR-IOV, there are a host of open-source software projects that complement and interact with DPDK. Many of these software initiatives, like VPP, XDP/ioVisor are also hosted within the Linux Foundation. And many, like DPDK, have multi-vendor support. All the following software approaches are in active development today, a testament to the richness and vitality of the open-source ecosystem.

VPP

Vector Packet Processor (VPP) is part of the Fast Data – Input/Output (FD.io) project under the Linux Foundation. VPP was originally contributed by Cisco as an open-source project. The goal of VPP is to provide a fast layer 2-4 userspace network stack that runs on common architectures like x86, Arm, and Power. Most implementations of VPP today leverage DPDK as a plug-in, to accelerate getting packets into userspace via DPDK PMDs.

VPP focuses on upper-layer networking protocols (Figure 7). VPP gets much of its performance by performing functions on batches or vectors of packets instead of on single packets. VPP provides a virtual switch and virtual router for layer two and layer three packet handling. It also incorporates an optimized TCP/IP stack that uses vectorized packet processing for increased performance. This makes building higher-layer (L4-7) network functions on top of VPP much easier than utilizing DPDK directly.

VPP is focused on recruiting developers of upper-layer network functions, such as virtual broadband gateways and security gateways. It is also moving to ensure that cloud-native platforms using containers can effectively use VPP.

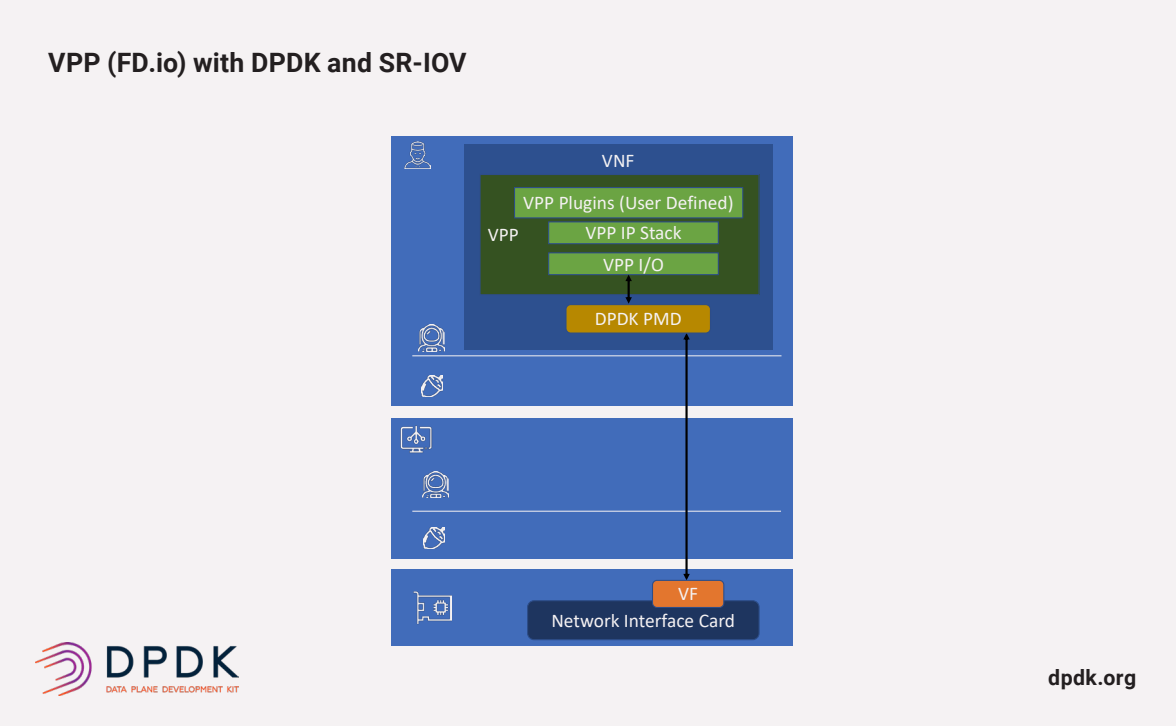


Figure 7

XDP and eBPF/BPF

The eXpress Data Path is an in-kernel packet processing framework under the ioVisor project, also hosted by the Linux Foundation. It was designed to provide high-performance packet processing in kernel space. XDP achieves this by hooking into the kernel to run an optimized code on incoming packets. These programs are executed under the extended Berkeley Packet Filter (eBPF) framework. eBPF is an improved version of the original BPF in-kernel solution created in 1992, which utilized a limited machine instruction set to run user-created programs, essentially in a lightweight virtual machine, in kernel space. Before being loaded into the kernel, these user programs were verified to ensure they had no malicious intent and terminate within a finite or short amount of time. The original uses of BPF were for network troubleshooting and analytics via filtering, soon to be extended for security use cases. Created in 2013, eBPF was an improvement to the original BPF and introduced a more sophisticated virtual machine. The eBPF user programs are compiled via a low-level virtual machine (LLVM) into eBPF instructions and then loaded into kernel to execute. Today, we use the term BPF to refer to eBPF, relegating the old BPF to classic BPF.

The XDP approach, therefore, achieves performance by running custom logic to handle packets in the kernel (Figure 8). XDP and BPF have been used to create sophisticated networking solutions, including security projects like Cilium. Taking an in-kernel approach as opposed to a kernel bypass approach, XDP is able to process packets while taking advantage of the kernel networking stack. It is under active development, improving its performance while adding sophisticated actions beyond dropping, switching, or passing into the kernel stack.

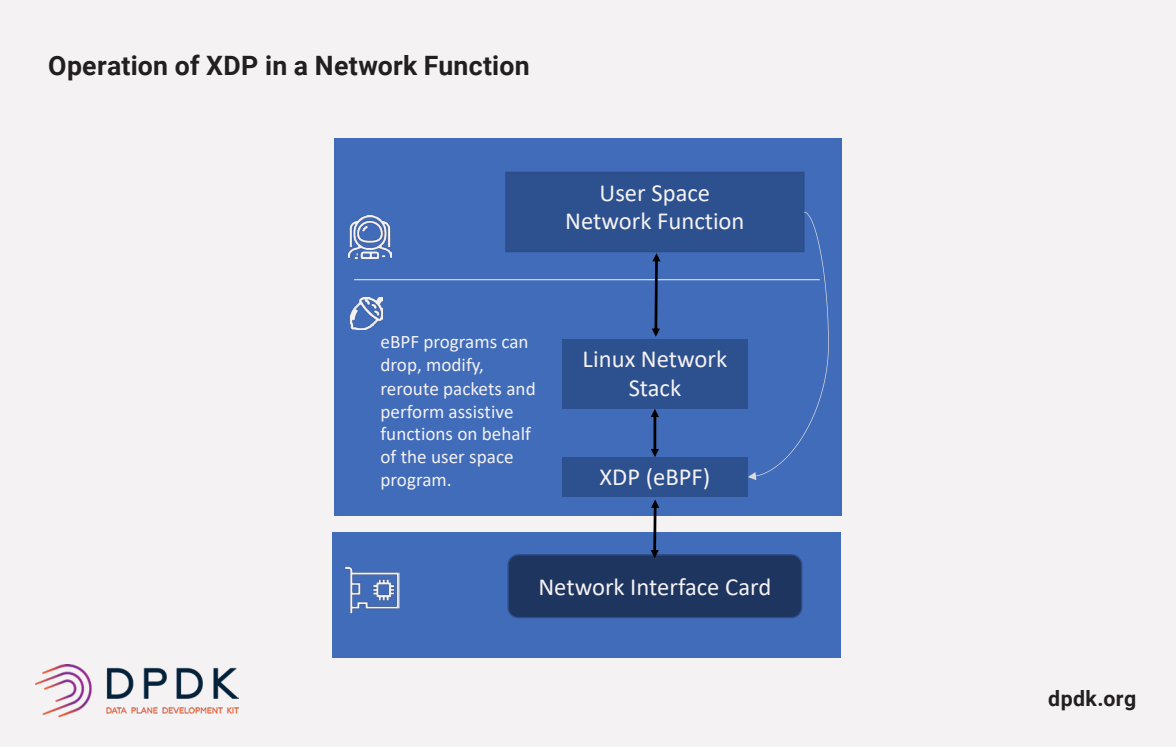


Figure 8

AF_XDP

AF_XDP stands for Address Family XDP, a new socket address family type in Linux. AF_XDP is related to XDP in that it uses the eBPF mechanism as well as the VXP driver layer. It steers packets matching certain criteria directly into userspace, in a way similar to DPDK's kernel bypass. When used in combination with XDP, this approach combines the best of in-kernel logic with DPDK-style kernel bypass. AF_XDP sockets allow in-kernel XDP programs to redirect frames to buffers in userspace for processing, or continue to shunt some traffic through the kernel's existing networking stack, TCP/IP, etc. (Figure 9).

AF_XDP so far does not achieve raw I/O performance that matches DPDK, lacking support for hardware offload. Nevertheless, the project has support from Intel, Red Hat, and Mellanox, among other vendors. One of the potential key advantages of the AF_XDP approach is reducing the need for vendor-specific PMDs and instead allowing for portable network function applications that do not concern themselves with the underlying NICs. In fact, the DPDK project has a PMD which uses the AF_XDP driver framework to simplify the large collection of vendor-specific PMDs, providing a bifurcated model for devices that do not have a native bifurcated model. However, this simplification comes at the cost of performance.

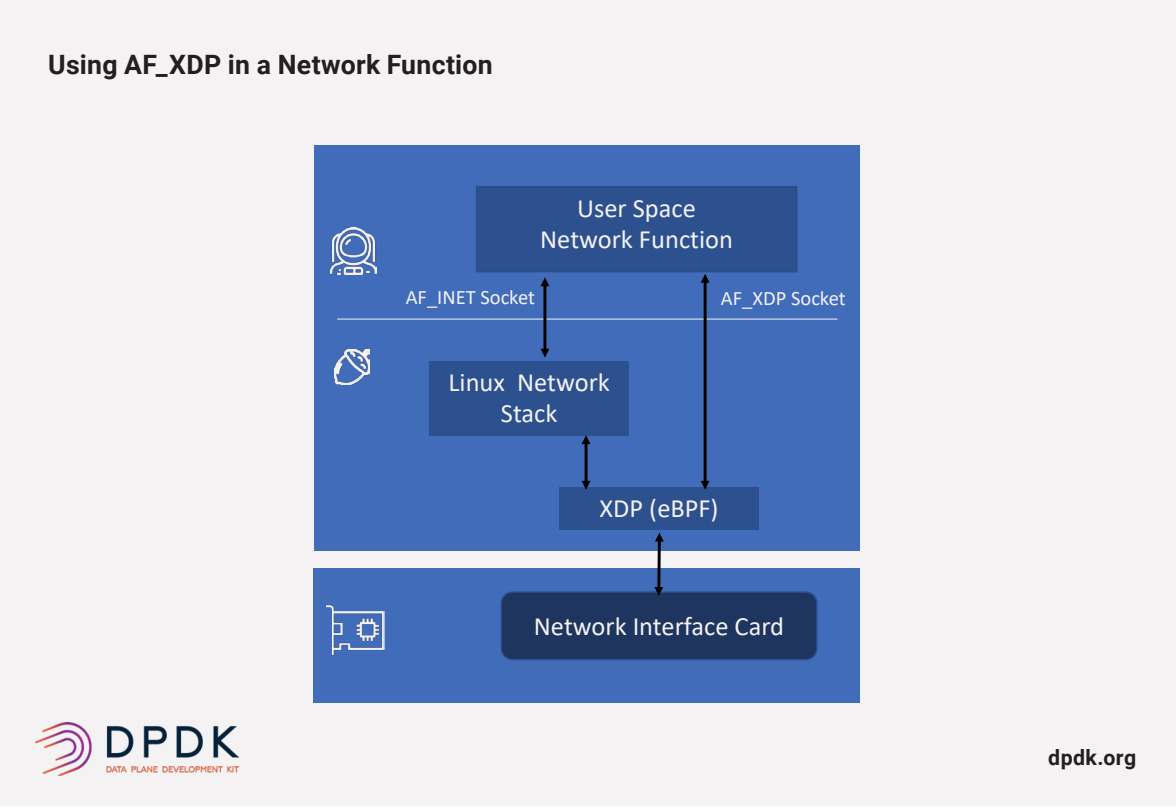


Figure 9

vDPA

The virtio data path acceleration (vDPA) project (sometimes referred to either as “virtual” or “vhost” data path acceleration) aims to standardize the NIC SR-IOV data plane. It uses the well-understood virtio framework, utilizing either a standardized virtio driver in guest VMs (Figure 10) or DPDK’s virtio-user PMD in containers, both of which are vendor-agnostic. Under the vDPA framework, each NIC vendor would conform to the virtio ring layout for data traffic. They would also provide a vendor vDPA driver that helps DPDK to translate control plane instructions into vendor-specific control plane commands. The goal is to reduce the large number of vendor-specific drivers (such as PMDs) that have to be maintained under DPDK. Like AF_XDP, it should also increase the portability of network functions by reducing vendor driver dependency.

vDPA is still in its early stages and under active development. It has support from major vendors such as Intel, Mellanox, and Red Hat.

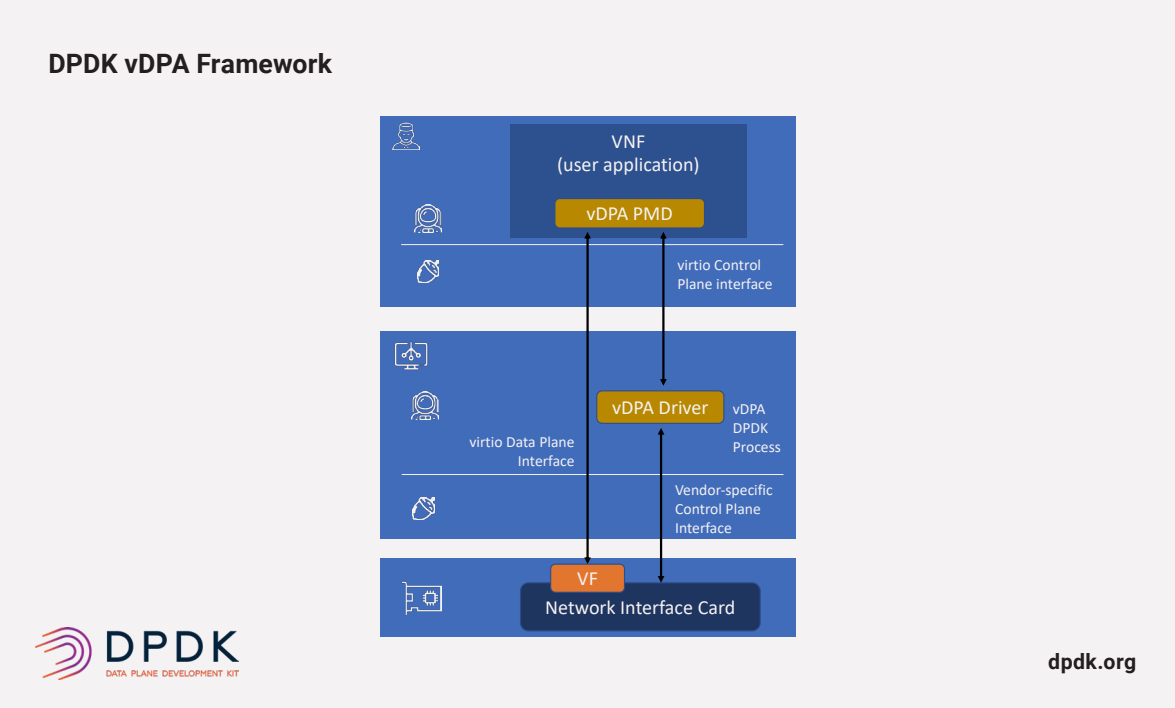


Figure 10

Switchdev and the Linux Kernel

The Ethernet switch device driver model (switchdev) is a framework for an in-kernel device model that offloads data plane traffic from the kernel into a switching ASIC.

The goal of switchdev is to allow users to access existing ASIC capabilities using well-known APIs on top of the Linux kernel. These users can then use the same type of commands and tools to configure either servers or switches. Under the switchdev model, instead of moving packets into user space or processing them in kernel space, packet handling could be offloaded into either an ASIC on a NIC card or a top-of-rack external switch, performing high-speed switching functions in hardware.

While there are new approaches to improving packet handling performance being worked on, DPDK remains the most popular and widely adopted library. The DPDK community is also actively collaborating with many of these other initiatives, including VPP, AF_XDP, and vDPA, to continue to evolve DPDK and improve both its performance and ease of use.

Infrastructure Acceleration Trends and DPDK's Continued Market Evolution

The continued generation and consumption of media at ever-higher resolutions, increased amount of data gathering for IoT devices, and growing dependence on big data and AI analytics for businesses drive network bandwidth consumption within data centers that are themselves both growing in number and scale.

As we described earlier, both the cloud and telco markets are seeing new requirements driven by market initiatives around 5G, IoT, and edge computing. New opportunities to virtualize the radio access network and the telco central office (CO) are pushing commodity servers into new locations. Disaggregated RANs running on x86 or Arm systems, edge workloads that handle video analytics, and edge systems acting as content delivery networks all demand higher network speeds and I/O processing capability. And with the continued focus on privacy and the need to prevent malicious attacks on communication streams, encryption has become critical.

When you add these requirements to the increased constraints on power, cooling, and space in these distributed locations, the need for DPDK and other methods to improve packet processing efficiency becomes paramount.

MYTH #8: DPDK is not green — it keeps the CPU busy all the time

Some believe that DPDK isn't green because PMDs are constantly polling to see if packets have arrived, regardless of whether packets are there or not. However, DPDK has an event-driven programming model available via the eventdev API that provides an alternate non-polling mode. Applications can choose which model or combination of models that works best for them. Additionally, the DPDK teams are working on methods to scale core frequencies to reduce power consumption during low traffic periods. In the end, DPDK allows telecom and cloud operators to increase the efficiency of existing platforms, reducing the need for additional CPUs while improving packet handling capacity.

FPGAs, ASICS, GPUs and their relationship with DPDK

For wireless gateways and security appliances at the edge running mobile workloads — like virtual evolved packet core (vEPC) — most of the gateway functions involve ingesting packets, examining headers and payloads, and performing some set of operations. DPDK is an ideal fit for these use cases, especially given its flexibility in running on different CPU types (x86, Arm).

When we examine these edge locations in greater detail, we see increased diversity of device types and new hardware capabilities — including FPGAs, GPUs, NPUs, and custom ASICs. DPDK will have to evolve to accommodate specialized NICs as well as converged infrastructure systems that may look different from the standard servers that live in today's data centers. Even within today's cloud data centers, there's increased adoption of SmartNICs to offload I/O processing.

Driven by cost and power consumption, we expect that lower-layer handling of packets could be offloaded to hardware counterparts like FPGA, ASICs, or NPUs. Simple switching, firewalling, basic load balancing, or routing functions might not require a general-purpose CPU's programmatic flexibility. However, for more complex L4-7 functions, DPDK will likely play a critical role in efficiently getting the packets to userspace for applications to perform deeper analysis. For example, any type

of Layer 7 proxies that involve HTTP header inspection or payload processing will likely lean towards using DPDK.

There are also efforts underway for collaborative security solutions between FPGAs and ASICs, where packets are shunted via DPDK into user space for analysis. Once a flow is determined to be safe, the FPGA or ASIC is able to direct flows without involving CPU cycles.

On the GPU front, we see some initiatives, like NVIDIA's parallel computing platform, CUDA, combined with virtual network functions (cuVNF), to provide direct payload copying straight into GPU graphic memory for signal processing in vRAN solutions. These are still early, and the relationship with DPDK will evolve over the next few years.

MYTH #9: DPDK is code-complete

As described in an earlier section in our paper, it's an ongoing active project with four releases per year. DPDK continues to adapt to the new needs of its users, the introduction of new accelerator devices, improvements to support container deployments, and extensions to other devices such as security and storage.

DPDK Accelerator Capabilities — RegEx, Compress Dev, SPDK, etc.

Many in the industry view DPDK as just a kernel bypass mechanism to get packets quickly into userspace. The DPDK framework is much richer than that. DPDK has evolved to essentially become a universal API that taps into acceleration capabilities of the underlying platform, including general-purpose CPUs and other capabilities such as cryptographic functions.

DPDK includes other libraries for compression and storage with work in progress on a regular expression matching (RegEx) library. More extensions are being considered, but here's a quick description of the emerging libraries:

- **RegEx** — a DPDK subsystem to provide accelerated matching of regular expressions, commonly used in header and payload matching for security and L7 load-balancing functions.
- **CryptoDev** — a library that provides an interface for accelerating operations like encryption/decryption, hashing, public/private key management, etc. based on a device's capabilities for generic cryptographic algorithms. It can also support a chain of cryptographic operations like enciphering and then hashing.
- **CompressDev** — a library that provides convenient access to common compressions algorithms, supporting both stateless and stateful compression to help in reducing payload sizes while offloading the CPU.
- **Security library** — an accelerator feature that uses a NIC or crypto device's hardware capabilities for offloading security protocols like IPsec and PDCP.
- **bbdev** — the Wireless Baseband library provides a common programming framework that abstracts hardware accelerators based on FPGA and fixed-function accelerators that assist with 3GPP physical layer processing, while decoupling the application from the compute-intensive wireless functions by abstracting their optimized libraries to appear as virtual bbdev devices.
- **eventdev** — the DPDK Event device library is an abstraction that provides an application with features to schedule events while helping to avoid constant interrupt polling.

In addition to the DPDK family, there's also the Storage Performance Development Kit (SPDK), which uses some DPDK code. While not a DPDK project, they are taking a similar approach to DPDK for storage protocols, attempting to accelerate storage I/O functions through the use of similar PMDs. By taking storage protocols straight into user space, they achieve similar performance gains in storage to those gained by DPDK in networking. SPDK uses a non-volatile memory express (NVMe) standard PMD with NvME over Fiber (NVMe-oF) and storage protocol iSCSI stacks in user space to implement high-performance storage systems.

DPDK and Linux Containers

DPDK was initially targeted at environments with VMs and worked well in both bare-metal deployments and with VMs. DPDK can be used for containers today with few issues, and support is improving rapidly.

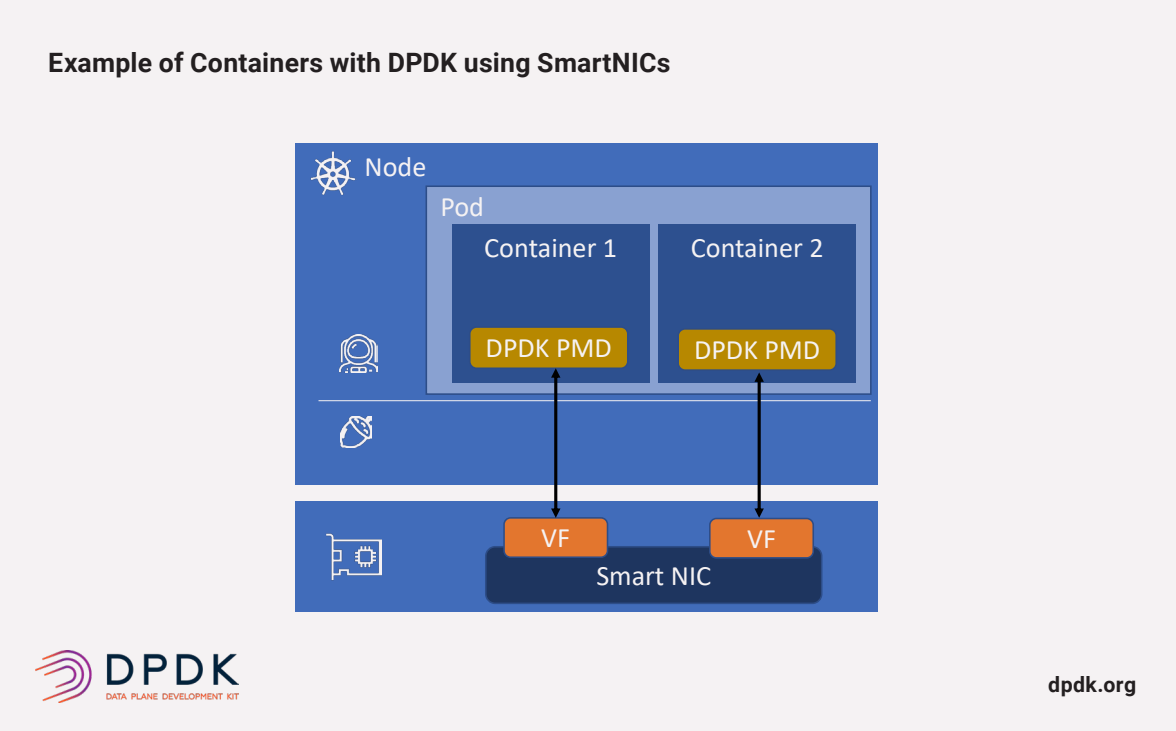


Figure 11

For example, recent releases of DPDK have made improvements for memory management that are more container friendly, such as not attempting to grab huge blocks of unused memory. Some developers have also expressed security concerns about the need to enable privileged context for containers using DPDK. This could increase the attack surface for the underlying OS, and DPDK might only be used in platforms where trusted workloads run. Nevertheless, developers are working on these issues, and we expect DPDK's container support will improve over time, given the prominence of containers in modern cloud-native architectures.

A Bright Future for DPDK

With the recent advent of SmartNICs, some in the industry had predicted the demise of DPDK. However, if anything, DPDK has evolved, and ongoing contributions to it have accelerated. As we look forward, we expect the following for DPDK:

- **Maturity and stability** — one of the unique attributes of DPDK today is the stability of the API. Unlike some other acceleration initiatives, DPDK is relatively mature, and the project maintainers understand the need to keep dependable APIs to prevent code breakage or constant patches and recompilation.
- **Ongoing performance improvements** — DPDK will continue to find ways to incorporate and integrate with SmartNICs and other hardware acceleration available. Likewise, we can expect ongoing software performance improvements.
- **Reduced complexity** — DPDK isn't the easiest to deploy and use and has multiple components that have to be configured and deployed correctly. There is one driver per hardware target as well as drivers for generic interfaces like AF_XDP and virtio (including vDPA). Additionally, there are myriad vendor-specific PMDs that have to be maintained. Vendors must maintain multiple drivers, and whether the AF_XDP unified PMD pans out or not, DPDK users would benefit from a more standardized way of integrating with the hardware.
- **Better cloud and container support** — container support today exists but can be enhanced. Ongoing improvements can be expected in DPDK memory management for container deployments, to align better with container and microservice architectures. Likewise, the security model needs to be tightened up so as not to require escalation of privileges for containers that need to use DPDK.
- **Better efficiency** — on Arm architectures, an event-triggered mode for pulling packets is supported, similar to the interrupt mode for PMDs on x86, and the eventdev subsystem helps to avoid constant polling. Approaches like this, as well as CPU clock scaling, will continue to help DPDK improve its non-green reputation.
- **Multi-vendor** — expect continued support across a wide range of instruction architectures beyond x86 (Intel and AMD), Arm, and PowerPC. Likewise, cryptographic and other acceleration capabilities from vendors like Intel, Mellanox, and Marvell will be furthered, along with support for emerging hardware vendors.
- **Rationalization and normalization of best practices** — upcoming releases will see different use cases and architectures emerge. The evolution of best practices for combining DPDK with SmartNICs (FPGA, ASIC, NPU) will continue, whether using DPDK on the SmartNIC itself or a collaborative and API-driven model where DPDK invokes SmartNIC acceleration.

Regardless, DPDK will continue to be a dominant framework in infrastructure acceleration in 2020 and beyond. As with all open-source projects, success is only possible through the participation of a diverse set of community members. Whether you are a software or hardware developer, QA, dev-test engineer, documentation expert, or marketing guru, the DPDK project is looking for new contributors and members.

To learn more, visit the Linux Foundation DPDK project today.