An Experiment of Container Networking using OpenVSwitch-DPDK

Quang-Huy Nguyen, *Younghan Kim
*Soongsil Univ.

huynq@ssu.ac.kr, *younghak@ssu.ac.kr

Abstract

Network Function Virtualization (NFV) is a prominent technology that replaces the traditional hardware network node by the virtualized network software. Besides, along with the development in the software field that divides an application into microservices, the container appears to adapt the reliability, availability, operationality of the application. To facilitate the development of the I/O intensive in the high-density container environment, the combination OVS and DPDK is a perfect that increases performance significantly. In this paper, we design and integrate OVS-DPDK in the container networking.

Ⅰ. Introduction

In recent years, the telecommunication industry has evolved towards Network Function Virtualization (NFV) where the traditional "hardware heavy" network node functions are being replaced by virtualized software that can run on generic hardware. Current de facto implementations of software virtualization utilize Virtual Machines (VMs) and is rapidly moving towards lightweight containers [1].

Based on the quick growth of container technology, network functions (like firewall, routing, etc.) are divided into many pieces that called microservices and linked with each components to create a service function chain replacing the existing monolithic applications. For instance, LinkedIn replaced their monolithic application with microservices in 2011 which resulted in less complex software and shorter development cycles. Currently, the container networking faces many challenges to adapt to this change. The high-density container causes the latency time to establish its network and time-consuming to container launch time.

To facilitate the development of the input/output intensive, the combination between OpenVSwitch (OVS) and Data Plane Development Kit (DPDK) [2] brings the significantly increased performance. OpenVSwitch is a feature-rich, multilayer, distributed virtual switch widely used as the main networking layer on virtual environments and other SDN applications [3]. DPDK is a framework that offers efficient implementations for a wide set of common functions such as NIC packet input/output, easy access to hardware features (e.g., SR-IOV, FDIR, etc.), memory allocation and queuing. OVS has traditionally been divided into a fast kernel-based datapath (fastpath) consisting of a flow table and a slower userspace datapath (slowpath) that processes packets that did not match any flow in the fastpath. By integrating OVS with DPDK, the fastpath is also in userland, minimizing kernel to userland interactions as well as leveraging the high performance offered by DPDK. The result is a ~10x performance increase of OVS with DPDK over native OVS. In this paper, we focus on the design and integrate OVS-DPDK for the container networking.

Ⅱ. OVS-DPDK overview

DPDK aims to provide a simple and complete framework for fast packet processing in data plane applications. It implements a "run to completion model for packet processing" meaning that all resources need to be allocated prior to calling the data plane application. Those dedicated resources are executed on dedicated logical processing cores.

This is opposed to a Linux kernel where we have a scheduler and interrupts for switching between processes, in the DPDK architecture the devices are accessed by constant polling. This avoids the context switching and interrupt processing overhead at the cost of dedicating 100% of part of the CPU cores to handle packet processing.

In practice DPDK offers a series of poll mode drivers (PMDs) that enable direct transfer of packets between user space and the physical interfaces which bypass the kernel network stack all together. This approach provides a significant performance boost over the kernel forwarding by eliminating interrupt handling and bypassing the kernel stack. The DPDK are a set of libraries. Thus, in order to use them, you need an application that links with these libraries and invokes the relevant APIs.

OVS is a software switch which enables the packet forwarding inside the kernel. It's composed of a userspace part and a kernel part:

- User space - including a database (ovsdb-server) and an OVS daemon for managing and controlling the switch (ovs-vswitchd).
- Kernel space - including the ovs kernel module responsible for the datapath or forwarding plane.

The OVS kernel module contains a simple flow table for forwarding packets received. Still, a small part of the packets which are called exception packets (first

packet in an Openflow flow) don't match an existing

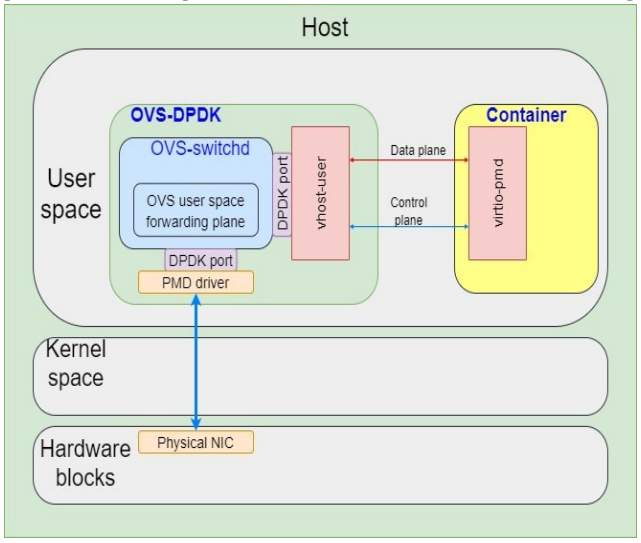

Figure 1. OVS-DPDK architecture

entry in the kernel space and are sent to the userspace OVS daemon (ovs-vswitchd) to be handled. The daemon will then analyze the packet and update the OVS kernel flow table so additional packets on this flow will go directly via the OVS kernel model forwarding table. This approach eliminates the need for context switches between user space and kernel space for most of the traffic however we are still limited by the Linux network stack that is not well suited for use cases with high packet rate demands.

Integrating DPDK into OVS can leverage the PMD drivers and move the previous OVS kernel module forwarding table to the user space. Otherwise, to boost up the networking container, the vhost-user/ virtio-pmd architecture is implemented to the OVS-DPDK. Vhost-user (backend) runs on the host userspace as part of the OVS-DPDK userspace application. As mentioned DPDK is a library and the vhost-user module are additional APIs inside that library. The OVS-DPDK is the actual application linked with this library and invoking the APIs. Virtio-pmd (frontend) runs on the container and is a poll mode driver consuming dedicated cores and performing polling with no interrupts. For an application running on the user space to consume the virtio-pmd it needs to be linked with the DPDK library as well. The figure 1 illustrates how these components come together.

III. Design and implementation

The figure 2 shows the components in the testbed. There are 2 containers that are created by the Docker [4]. Both containers use DPDK application to boost up the network performance. One container is deployed Pktgen which is a high-performance networking testing tool [5]. Other one is deployed testpmd application that receives the traffic. The traffic is generated from virtio-pmd1 on pktgen to the virtio-pm4 and the egress of the traffic from virtio-pmd3 on testpmd to virtio-pmd2.

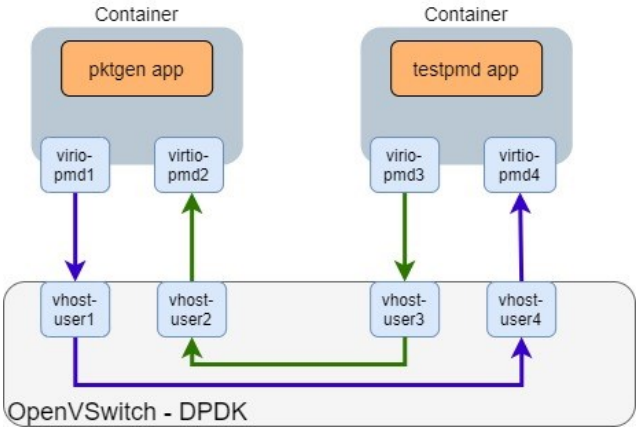

Figure 2. Test bed

IV. Evaluation

The DPDK port processes an average of about 82432 packets per second in case of the 64-byte packet, and this number decreases to about 71192 as the packet size increases (Table 1). As for the throughput, it shows an average of about 36 Mbps at the 64-byte packet size, and it increases in multiples up to about 550 Mbps as the packet size increases. The results show that the smaller packet size is, the more packets CPU has to process.

Table 1. Performance measurement of the testbed

Packet Size (bytes)	Packet per Second (pps)	Throughput (Mbps)
64	82432	36
128	83420	59
256	78912	102
512	73696	216
1518	71192	550

IV. Conclusion

In this paper, we implemented the OpenVSwitch-DPDK and designed the testbed for container network. In the future, we will conduct service migration between containers combining the OVS-DPDK to reduce latency and increase the pace of migration.

ACKNOWLEDGMENT

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2020-2017-0-01633) supervised by the IITP(Institute of Information & Communications Technology Planning & Evaluation)

REFERENCES

[1] Bernstein, David. "Containers and cloud: From lxc to docker to kubernetes," IEEE Cloud Computing 1.3, pp 81-84, 2014.

[2] http://docs.openvswitch.org/en/latest/intro/install/dpdk/

[3] http://www.openvswitch.org/

[4] https://www.docker.com/

[5] Olsson, Robert. "Pktgen the linux packet generator," Proceedings of the Linux Symposium, Ottawa, Canada. Vol. 2. 2005.