Accelerating packet processing in a Xen environment With OpenDataPlane

Tarek Rabia, Othmen Braham, Guy Pujolle

Accelerating packet processing in a Xen environment With OpenDataPlane

Tarek RABIA
VirtuOR
Paris, France
Tarek.Rabia@virtuor.fr

Othmen BRAHAM
VirtuOR
Paris, France
Othmen.Braham@virtuor.fr

Guy PUJOLLE
Laboratoire d'Informatique de Paris 6 (LIP6)
Paris, France
Guy.Pujolle@lip6.fr

Abstract—**Over recent years, packet processing acceleration has become a hot topic. Indeed, several software solutions have been developed to offer the best possible performance. Recently, a new open source project has started called OpenDataPlane (ODP), this framework provides a set of APIs, configuration files, and other functions to accelerate packet processing and manage effectively the networking data plane. In this paper, we present our new Xen architecture, implemented within Metamorphic Networks "MNet" platform that integrates ODP in a privileged domain, called driver domain. This architecture allows us to associate ODP with virtual CPU cores in order to accelerate packet processing and improve the performance of our platform, without adding overhead in physical processors. The results of our experiments show that our new architecture improves packet processing performance by 15%, only using the virtual CPU resources.**

Index Terms—**OpenDataPlane; Packet processing acceleration; Xen architecture;**

I. Introduction

The world of computer networks has developed considerably over the past decade. This was accompanied by the exponential growth of users number and their data which impact the behavior and the performance of these networks. Nowadays, it is no longer sufficient to expand the physical network infrastructure in order to treat such a charge. But it is necessary to add mechanisms and tools to manage a major amount of data in efficient and optimal way. To address this problem, ICT factories have developed softwares such as packet processing accelerators, dedicated to the networking data plane improvement. These accelerators use the resources offered by multi-core processor (CPU) architectures to accelerate packet processing, and thus quickly handle a large amount of data.

Over the last three years, a wide range of packet processing accelerators, and software development kits have emerged, giving developers and users a variety of functions and libraries that allows the adaptation of these accelerators to their own devices and Network Interfaces (NIC) for better performance in the data plane. Currently, two software solutions are mainly used, Intel Data Plane Development Kit (DPDK) [1] and netmap [2]. DPDK is a set of open source libraries and drivers developed by Intel and 6Wind, only compatible with Intel processors and Intel NIC's. These libraries offer an appropriate environment for programmers to develop and implement packet processing acceleration functions adapted to their architectures. Netmap is an integrated framework in FreeBSD and Linux that handle a large traffic load, without depending on a specific hardware.

Recently, a new open source project has been developed called OpenDataPlane (ODP) [3]. ODP is a set of Applications Programming Interfaces (API's), using multi-core CPU architecture and allowing to manage effectively the networking data plane. In this work, we integrate ODP to our virtual environment. Our choice was motivated by the ease of integration of ODP within the Xen virtual environment [4] and for his interesting packet processing performance. However, ODP overuses the physical CPU cores of a device (up to 89%). This overuse could negatively impact the behavior and the performance of other processes executed by the device, particularly when the number of cores is limited. To address this issue, we propose a solution to optimize the use of CPU cores, without penalizing the performance obtained by ODP. We opt for the implementation of a new Xen architecture, integrating ODP in a virtual privileged domain, called driver domain. Within the driver domain, we create and instantiate multiple virtual CPU cores that will be used by ODP to accelerate packet processing without overloading the use of physical CPU cores.

Our paper is organized as follows, we start in the section II by presenting some related works done in the packet processing field. In section III, we give an overview of OpenDataPlane framework and explain the functioning of its components. Section IV focus on our new architecture, integrating ODP within the Xen driver domain. Section V presents the implementation of our architecture in the "Metamorphic Networks"(MNet) platform [5]. Section VI contains the performance results obtained by our architecture so far. Finally, we conclude our paper with a conclusion and perspectives of our future work.

II. Related work

Many studies have been made in the packet processing acceleration and resource optimization domains. We quote the two Xen architectures presented in [6] and [7] that allow to optimize performance of network virtualization and which has also inspired us in our work. Another interesting contribution in this area is ClickOS [8] which is a Xen-based software platform, optimized for use on middleboxes. Implemented within a virtual machine, ClickOS enables a faster packet processing and achieve great performance on 10 Gbits/s Ethernet NIC's.

Some works were carried out in Network Functions virtualization(NFV) [9], based on primitives offered by Intel DPDK to build an architecture that can simultaneously handle throughput of multiple network virtualized functions [10]. Likewise, NetVM [11] is a new virtualization-based platform built on top of DPDK libraries and KVM environment [12]. NetVM uses high throughput packet processing achieved by DPDK to support high speed communication between virtual machines. Hwang et al. indicate that NetVM improves throughput of more than 250% compared to SR-IOV architecture [13].

Other authors have proposed their own software solutions for accelerating packet processing. Luigi Rizzo and Giuseppe Lettrier presented an architecture to handle large traffic loads. Called VALE [14], this architecture was the precursor of Netmap. A performance comparison has also been made between different packet processing accelerator frameworks in [15] and [16].

Recently, Garzella et al. presented pnetmap [17], a virtual passthrough network device that allows Virtual Machines to connect to any netmap port by removing the constraints of hardware passthrough. HAN et al. designed a new hybrid software/hardware architecture called SoftNIC [18]. SoftNIC provides a programmable platform that extends or augment the NIC features for better flexibility and performance, comparing to the simple hardware NIC's. To accelerate packet processing and increase throughput performance, SoftNIC architecture uses Intel-DPDK and can reach 40 Gbits/s with 1 or more cpu cores.

III. OpenDataPlane overview

OpenDataPlane (ODP) is a new open source project that provides an environment for easy programming of data plane applications. This environment consists of a set of API's, configuration files, and other functions, operating on linux, forming a packet processing scheme that can adapt to different underlying platforms. ODP aims to separate the data plane design of the underlying hardware design, by abstracting network characteristics (reception, transmission, etc.) associated with heterogeneous SoC technologies, also by abstracting flow management for a better benefit of scheduling services and flow classification provided by ODP API's.

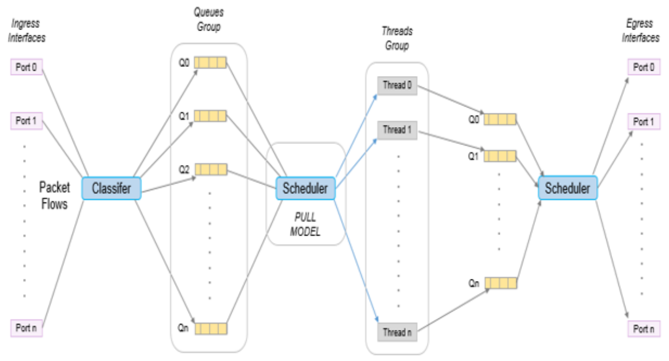

Fig. 1. ODP packet processing scheme using Pull Model

The other objective of ODP is to accelerate the packet processing. For this purpose, ODP incorporates within its API's a functions (buffers, queues, schedulers, classifiers, etc.) allowing an optimal use of available physical resources in order to accelerate packet processing. Developed to run as a process in the Linux user space, ODP implements two architectures which are the Push Model and the Pull Model (Fig. 1). We focus in this paper to the second model which is explained below:

- at the arrival of packets at the ingress ports managed by ODP, a classifier splits the traffic to flows according to the packet type. The different flows are then directed to the corresponding queues (depending of the flow type). Each queue (or a group of queues) handles only a specific flow type,
- a scheduler selects a packet from a queue and sends it to a thread for processing. The scheduler is a specific function to pull model. In the push model, threads are directly connected to the queues,
- once the packet processed by the thread, it is sent to the egress queue and then a scheduler redirects the packet to the egress port in order to be forwarded to their destination.

As mentioned above, the difference between the Pull model and the Push model is the non-existence of the scheduler in the second model, as shown in Fig. 2. Therefore, the packets queues are directly linked to the threads. The advantage of using a scheduler is to have the possibility of prioritizing some defined packets flows comparing to others, for a faster processing. In our work, we used the Pull model.

The number of threads launched by ODP depends on the number of CPU cores allocated to the application. Thus, each thread will use all the resources of the corresponding core in order to accelerate the processing of the packets going through this thread. The processing speed will depend on the number of threads (number of allocated cores) launched by the application.

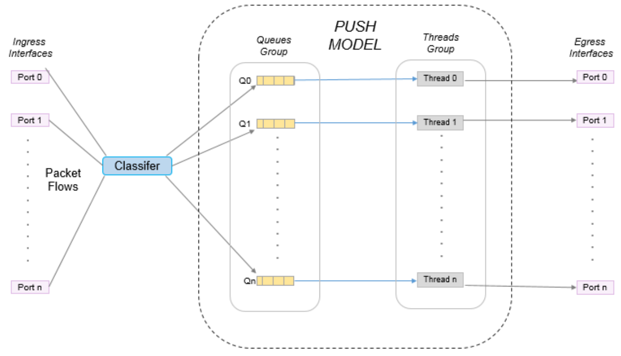

Fig. 2. ODP packet processing scheme using Push Model

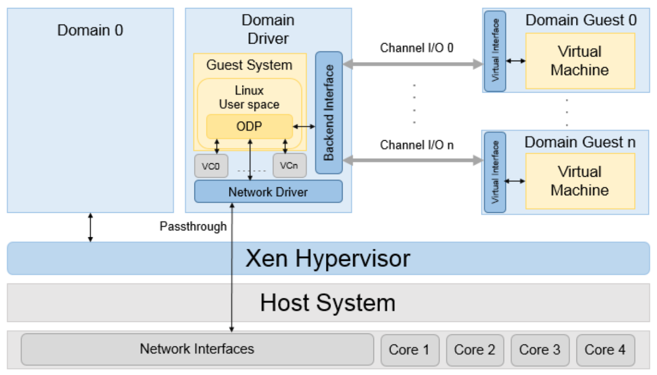

Fig. 3. Our new Xen architecture

IV. Our Xen architecture

To design our architecture (Fig. 3), we based on Xen environment and the I/O architecture presented in [6][7] which is composed by a driver domain, using its own NIC drivers to direct access (passthrough) to the I/O ports. This privileged domain acts as an intermediate interface between the physical I/O ports and the less privileged domains, called guest domains. The driver domain is connected to the guest domains through a virtualized channel, connected to a backend interface (driver domain side) and virtual interfaces (guest domains side).

To accelerate the processing of packets ranging towards the guest domains and passing through the driver domain, we deploy ODP within a linux virtual machine located in the driver domain. The interest of this implementation is to control the allocated virtual CPU resources through the driver domain. Indeed, the driver domain allows to determine (by adding or deleting) the number of virtual CPU cores (VCi) that will be used by ODP in order to accelerate packet processing. This solution has two advantages, the first advantage is to not be limited by the number of CPU cores that can be allocated for ODP. The second advantage is that adding virtual CPU cores has no influence on the use of the underlying physical CPU resources. Therefore, this solution offers an efficient and optimal way to manage CPU resources used by ODP for accelerating packet processing.

The principle of the solution implemented in our architecture is to replace the physical CPU cores by virtual cores located in the driver domain. Thus, ODP launches a number of threads corresponding to the number of virtual cores of the driver domain. The threads then accelerate the processing of packets that pass through the driver domain, without creating an additional load at the underlying physical processor.

The use of ODP in our architecture brings multiple advantages, the main advantage is the compatibility of ODP with the majority of network cards and drivers available in the market. This is due to the abstraction level brought by ODP API's. A further provided advantage is the possibility to classify the different packet flows with the functions (classifier) provided by ODP, this allows us to prioritize packet flows for a better monitoring. In addition, it is possible to associate ODP with other packet processing accelerators, such as Intel DPDK or netmap, to provide a better performance.

V. Implementation details

We implement the architecture presented above within the "Metamorphic Networks" platform (M-Net) [5]. This platform is based on the urbanization of virtual machines, thus it offers the possibility of creating, moving, or removing dynamically VM's within a Xen environment. The MNet platform (Fig. 4) is composed by a set of physical nodes (devices) called MNet-boxes and developed by VirtuOR [19]. These nodes are connected to each other through a wired network using the layer 2 TRILL protocol [20]. We chose to implement TRILL within our platform because it offers the benefits of layer 3 protocol, in terms of calculating the shortest path (by using IS-IS protocol [21] and Djikstra algorithm), and offers also the benefits of a layer 2 protocol, in terms of forwarding simplicity and speed. In each MNet-box, a Xen environment is implemented to monitor the different virtual machines instantiated within the box.

As discussed in Section IV, the Xen environment of each node contains the main domain Dom0, a driver domain containing a Linux virtual machine wherein ODP is installed, and guest domains that represent virtual machines containing users applications. The driver domain directly manages the physical I/O ports of the MNet-box by passthrough. Therefore, all traffic going to different guest domains is managed by the driver domain and ODP.

VI. Performance evaluation

To evaluate the performance of our solution, we rely on the implementation presented in Section V. We

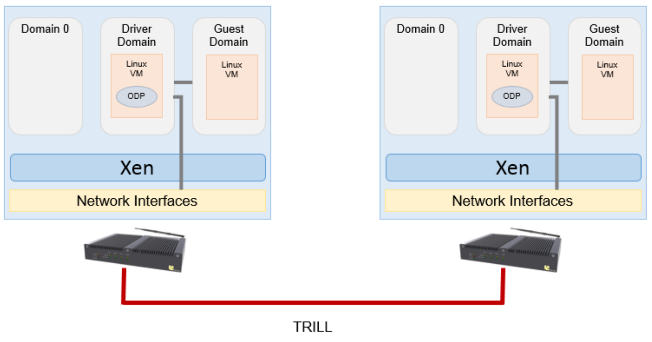

Fig. 4. Implementation scheme

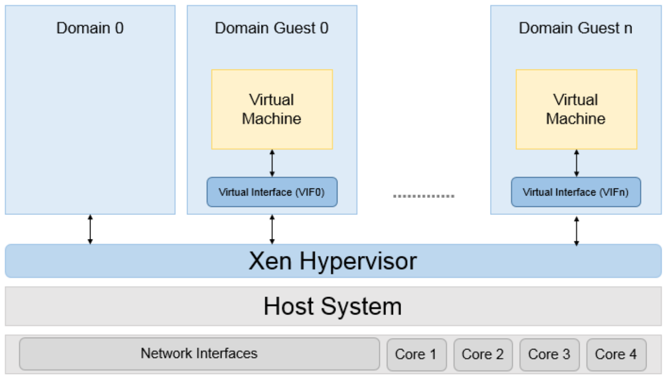

Fig. 5. Our old Xen architecture

perform our measurements on a topology composed by a couple of connected MNet-boxes. MNet-box is equipped with an 2.5 GHz Intel core 2 duo processor and four Intel 82571EB Gigabit Ethernet cards. Each box includes a proprietary Linux distribution developed by VirtuOR and containing the Xen environment with the architecture that we presented in the section IV.

We present different measurements carried from a comparison between our new Xen architecture (using ODP and driver domain) and our old Xen architecture (Fig. 5) that we call native architecture. In contrast to the new architecture, in the native architecture, the guest domain is directly connected to the host (no intermediate driver domain). Moreover, ODP is not implemented.

We evaluate several parameters such as maximum reached throughput, number of processed packets, bandwidth use percentage and use percentage of the virtual and physical CPU resources for both architectures. For the native architecture, we evaluate the performance between two guest domains (virtual machines) located on two separate physical nodes (MNet boxes). To do this, we use Iperf tool [22] to generate UDP traffic during a period of 100 seconds. Each generated packet has a fixed size to 1512 bytes. For the new architecture, we evaluate the performance of the implementation illustrated in Fig. 4. We use the ODP traffic generator to generate

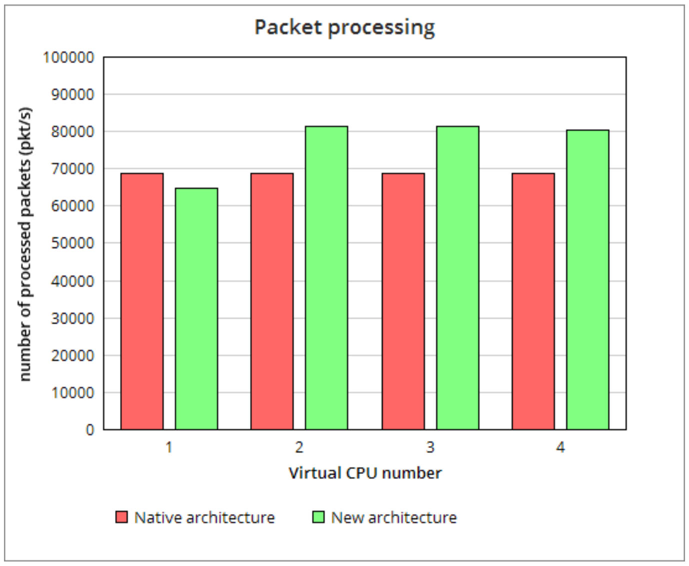

Fig. 6. UDP packets processed by each architecture

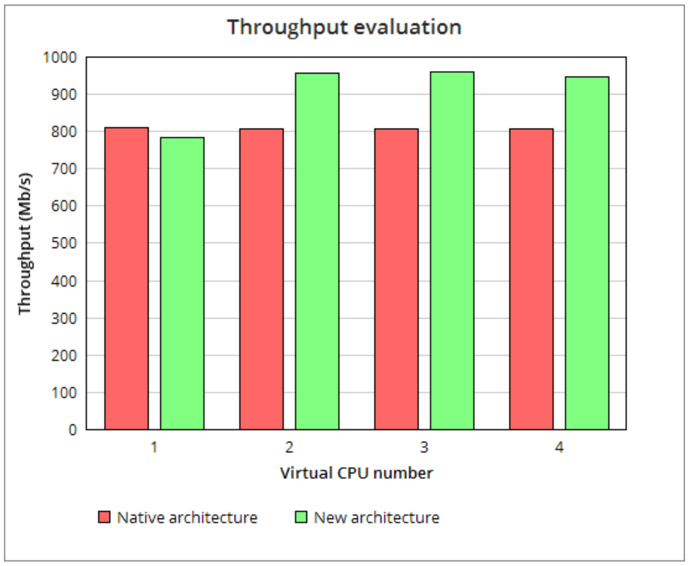

Fig. 7. Obtained throughput comparison

UDP packets from the guest domain. These packets are the same size (1512 bytes) and the same rules as a UDP packet generated by Iperf. Traffic generated by the guest domain will be reached through the driver domain which is connected to it. We use a generator for this time because ODP is not compatible with Iperf. We measure the obtained average for each performance parameter during this period of 100s. To make a comparison, we vary the number of virtual CPU cores used by the guest domain (native architecture) and the driver domain (new architecture). The obtained results are shown in the above figures.

Fig. 6 shows the number of packets processed per seconds, depending on the variation of VCPU cores number

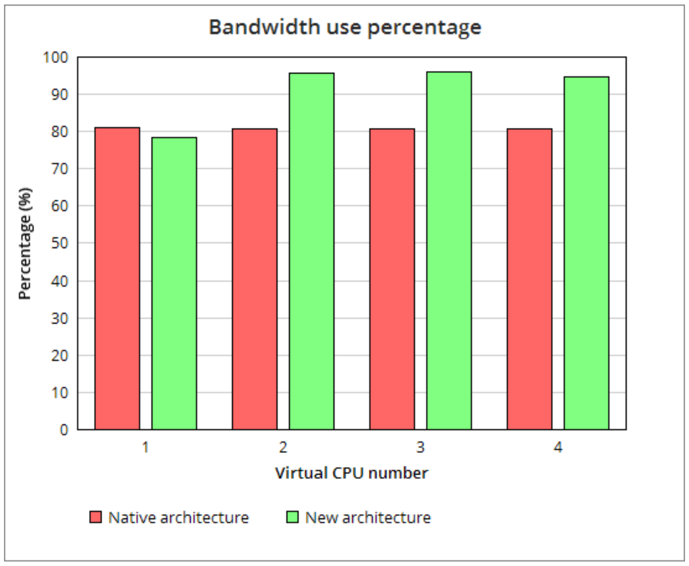

Fig. 8. Bandwidth use percentage comparison

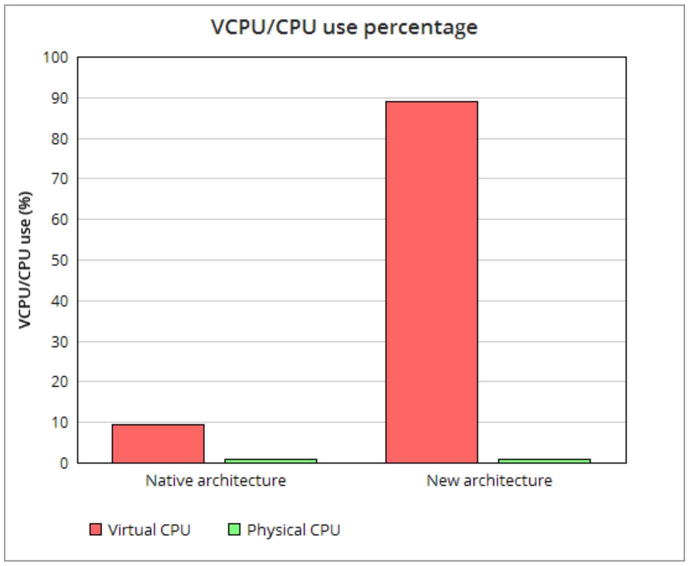

Fig. 9. used VCPU/physical CPU resources

of virtual machines. We note that the addition of ODP in our new architecture enables us to get an interesting gain (+ 15%) when the number of VCPU cores is more than 1. Thus, the new architecture allows us to achieve 958 Mbits/s (Fig. 7) with Intel 1 Gbit/s NIC cards. We also note in Fig. 8 that we can achieve the maximum use of bandwidth (95%) when using 2 VCPU cores in the new architecture. However, adding a higher number of VCPU cores slightly reduces performance obtained by ODP ($<$ 80000 pkts/s). This is due to the inability of the NIC to handle a larger number of packets. We also see through Fig. 6, Fig. 7 and Fig. 8 that the number of VCPU cores has no influence on the performance of the old architecture that does not use ODP. Indeed, the use of bandwidth percentage remains constant at 80% despite the addition of VCPU cores.

We observe in Fig. 9 that the only CPU resources used for packet processing are virtual resources (89% for the new architecture and 9.4% for the native architecture). Therefore, there is a low overhead in physical resources of the host domain (Dom0) (+1%) for both architectures. We also note that ODP consumes a lot of CPU resources, regardless of the number of used VCPU cores by VM.

VII. Conclusion

To accelerate the packet processing within our Metamorphic Networks platform, we deployed the new open source software framework OpenDataPlane(ODP) which provides a set of APIs to accelerate packet processing and improve data plane. Our choice was motivated by the fact that ODP ignores the underlying physical layers, and therefore it can handle a large number of devices and NIC drivers. In addition, ODP integrates functions tailored to our platform. However, ODP uses a lot of CPU resources and thus can be implemented only in devices that have mult-icore processors (8 cores or more). Therefore, we incorporated ODP within the Xen Driver domain in order to create and manage a large number of virtual CPU cores that ODP will use to accelerate packet processing. We showed through our experiments that the solution we proposed, accelerates packet processing and improves the performance of our architecture. In addition, the use of virtual CPU resources avoids an additional overhead in the physical processor, and provides an increased flexibility in the management of CPU resources that can be assigned to ODP.

In this paper, we presented our new architecture integrated in a Xen environment, using the privileged domain called driver domain as an intermediary between Guest domains and physical underlying platforms, on which the driver domain is connected in passthrough. We then integrated ODP within a Linux VM located in driver domain to improve the performance of this architecture.

Currently, we are still in performance measurement phase. Our objective is to test this solution on 10Gbits/s Ethernet NIC's and see the contribution of ODP in term of performance. For our future work, it would be interesting to make a comparison with other packet processing accelerators (Netmap, DPDK, etc.) that can be coupled with the ODP tools. In addition, we keep working on ODP libraries and functions in order to develop a new API's that manage flow types in virtual networks

Acknowledgment

We would like to thank our colleagues and members of VirtuOR and the PHARE team of Paris 6 Laboratory for their assistance in our work and our research and for their advices on the working methodology. We would like to recognize and thank all the researchers working on the packet processing acceleration area for their contributions that have improved the IT networks and have helped us greatly to move forward.

References

[1] INTEL, D. P. D. K. Data Plane Development Kit. URL http://dpdk.org.

[2] RIZZO, Luigi. netmap: A Novel Framework for Fast Packet I/O. In : USENIX Annual Technical Conference. 2012. p. 101-112.

[3] OpenDataPlane. URL http://www.opendataplane.org.

[4] BARHAM, Paul, DRAGOVIC, Boris, FRASER, Keir, et al. Xen and the art of virtualization. ACM SIGOPS Operating Systems Review, 2003, vol. 37, no 5, p. 164-177.

[5] BRAHAM, Othmen et PUJOLLE, Guy. The metamorphosing network (M-Net). In : Global Information Infrastructure and Networking Symposium (GIIS), 2012. IEEE, 2012. p. 1-4.

[6] MENON, Aravind, COX, Alan L., et ZWAENEPOEL, Willy. Optimizing network virtualization in Xen. In : Proceedings of the annual conference on USENIX. 2006. p. 2-2.

[7] FRASER, Keir, HAND, Steven, NEUGEBAUER, Rolf, et al. Safe hardware access with the Xen virtual machine monitor. In : 1st Workshop on Operating System and Architectural Support for the on demand IT InfraStructure (OASIS). 2004. p. 1-1.

[8] MARTINS, Joao, AHMED, Mohamed, RAICIU, Costin, & al. ClickOS and the art of network function virtualization. In : 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14). USENIX Association, 2014. p. 459-473.

[9] CHIOSI, Margaret, CLARKE, Don, WILLIS, P., et al. Network functions virtualisation introductory white paper. In : SDN and OpenFlow World Congress. 2012.

[10] CERRATO, Ivano, ANNARUMMA, Mauro, et RISSO, Fulvio. Supporting fine-grained network functions through Intel DPDK. In : Software Defined Networks (EWSDN), 2014 Third European Workshop on. IEEE, 2014. p. 1-6.

[11] HWANG, Jinho, RAMAKRISHNAN, K. K., et WOOD, Timothy. NetVM: high performance and flexible networking using virtualization on commodity platforms. Network and Service Management, IEEE Transactions on, 2015, vol. 12, no 1, p. 34-47.

[12] KIVITY, Avi, KAMAY, Yaniv, LAOR, Dor, et al. kvm: the Linux virtual machine monitor. In : Proceedings of the Linux Symposium. 2007. p. 225-230.

[13] DONG, Yaozu, YANG, Xiaowei, LI, Jianhui, et al. High performance network virtualization with SR-IOV. Journal of Parallel and Distributed Computing, 2012, vol. 72, no 11, p. 1471-1480.

[14] RIZZO, Luigi et LETTIERI, Giuseppe. Vale, a switched ethernet for virtual machines. In : Proceedings of the 8th international conference on Emerging networking experiments and technologies. ACM, 2012. p. 61-72.

[15] EMMERICH, Paul, WOHLFART, Florian, RAUMER, Daniel, et al. Comparison of frameworks for high-performance packet IO. In : Architectures for Networking and Communications Systems (ANCS), 2015 ACM/IEEE Symposium on. IEEE, 2015. p. 29-38.

[16] BARBETTE, Tom, SOLDANI, Cyril, et MATHY, Laurent. Fast Userspace Packet Processing. In : Proceedings of the Eleventh ACM/IEEE Symposium on Architectures for networking and communications systems. IEEE Computer Society, 2015. p. 5-16.

[17] GARZARELLA, Stefano, LETTIERI, Giuseppe, et RIZZO, Luigi. Virtual device passthrough for high speed VM networking. In : Architectures for Networking and Communications Systems (ANCS), 2015 ACM/IEEE Symposium on. IEEE, 2015. p. 99-110.

[18] HAN, Sangjin, JANG, Keon, PANDA, Aurojit, et al. SoftNIC: A Software NIC to Augment Hardware. UCB Technical Report No. UCB/EECS-2015, 2015.

[19] VirtuOR. http://www.virtuor.fr/

[20] PERLMAN, Radia. Rbridges: transparent routing. In : INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies. IEEE, 2004. p. 1211-1218.

[21] ORAN, David. OSI IS-IS intra-domain routing protocol. 1990.

[22] TIRUMALA, Ajay, QIN, Feng, DUGAN, Jon, et al. Iperf: The TCP/UDP bandwidth measurement tool. htt p://dast. nlanr. net/Projects, 2005.