applied sciences

Article

The Effects of High-Performance Cloud System for Network Function Virtualization

Wu-Chun Chung * and Yun-He Wang

Information and Computer Engineering, Chung Yuan Christian University, Taoyuan 320314, Taiwan
* Correspondence: wcchung@cycu.edu.tw

Abstract: Since ETSI introduced the architectural framework of network function virtualization (NFV), telecom operators have paid more attention to the synergy of NFV and cloud computing. With the integration of the NFV cloud platform, telecom operators decouple network functions from the dedicated hardware and run virtualized network functions (VNFs) on the cloud. However, virtualization degrades the performance of VNF, resulting in violating the performance requirements of the telecom industry. Most of the existing works were not conducted in a cloud computing environment, and fewer studies focused on the usage of enhanced platform awareness (EPA) features. Furthermore, few works analyze the performance of the service function chain on a practical cloud. This paper facilitates the OpenStack cloud with different EPA features to investigate the performance effects of VNFs on the cloud. A comprehensive test framework is proposed to evaluate the verification of functionality, performance, and application testing. Empirical results show that the cloud system under test fulfills the requirements of service level agreement in Rally Sanity testcases. The throughput of OVS-DPDK is up to 8.2 times as high as that of OVS in the performance test. Meanwhile, the hardware-assisted solution, SR-IOV, achieves the throughput at near the line rate in the end-to-end scenario. For the application test, the successful call rate for the vIMS service is improved by up to 14% while applying the EPA features on the cloud.

Keywords: enhanced platform awareness; test framework; cloud computing; network function virtualization; performance evaluation

check for updates

Citation: Chung, W.-C.; Wang, Y.-H. The Effects of High-Performance Cloud System for Network Function Virtualization. *Appl. Sci.* **2022**, *12*, 10315. https://doi.org/10.3390/app122010315

Academic Editors: Pedro Valero-Lara and Antonio J. Pena

Received: 17 September 2022
Accepted: 11 October 2022
Published: 13 October 2022

1. Introduction

In 2012, the European Telecommunications Standards Institute (ETSI) launched an industry working group to comply with the Network Function Virtualization (NFV) framework and standardization [1]. The primary goal of NFV technologies is to decouple network functions from the dedicated hardware to be run on commodity servers as software. With the help of NFV, telecom operators exploit commercial off-the-shelf servers and virtualization technologies to provide a variety of network functions. Therefore, telecom operators reduce capital expenditure while improving flexibility from the benefits of NFV technologies [2]. Cloud computing also provides scalability and availability for NFV services to be more reliable in business operations [3,4]. These advantages are attractive for telecom operators towards NFV on the cloud.

OpenStack, an open-source cloud operation system, plays a critical role in network function virtualization infrastructure (NFVI) and virtualized infrastructure managers (VIM) [5] of the ETSI framework. Telecom operators build an on-premises cloud on multiple nodes and deploy virtual network functions (VNFs) on top of the NFVI. However, the network functions in virtual machines (VMs) require a hypervisor or virtual machine manager (VMM) to mediate available virtual and physical resources. This extra layer brings challenges to the performance degradation of VNFs [6] on a cloud system. The performance on computation and networking is dramatically reduced without any improvement.

Many works attempt to improve the efficiency of virtual resources on the cloud from different perspectives. The workflow scheduling problem is addressed in [7–9] to improve the workload execution on the cloud system. The authors of [10–14] focus on VM placement to improve the power consumption for the cloud system. Some works [15–19] further take the load-balancing issue into consideration for better resource utilization in VNF placement. Previous works have proposed effective mechanisms and algorithms to improve the resource management of VNF on the cloud. However, rare works emphasize performance enhancement techniques from scratch for system virtualization.

Intel Corp. devotes massive efforts to enhanced platform awareness (EPA) [20] for virtualization technologies. The bottlenecks of computation performance are enhanced by CPU pinning for allocating dedicated CPU cores to a virtual machine (VM), and hugepages for improving effective memory access. A practical cloud system with enhanced technologies for computation has been preliminarily verified in [21]. For the enhancement of network performance, Open vSwitch with DPDK (OVS-DPDK) and single root I/O virtualization (SR-IOV) are two effective technologies to overcome the communication bottleneck. However, most of the previous works mainly focus on the perspective of network devices [22–25] rather than a comprehensive verification of the overall cloud system under test (SUT).

In addition, a telecom service usually requires chaining of network functions, named the service function chain (SFC) [26], to serve the NFV application. The length of a service chain has an impact on network performance. When the path of a service chain contains only one VNF on the cloud system, the packets are sent from a traffic generator to the target VNF via a physical network interface controller (NIC). The packets are then transmitted back to the traffic generator through a physical top-of-rack (TOR) switch. This communication scenario is called a physical–virtual–physical (PVP) flow. If the SFC is compounded by two VNFs, the packets are sent from a traffic generator to one VNF and forwarded to another VNF before transmitting back to the traffic generator. This scenario is a typical physical–virtual–virtual–physical (PVVP) flow. When the service path becomes longer, the workload of components in this path is heavier. Thus, the performance is deteriorated, but less attention has been paid to this in the literature.

Considering the telecom service as the use case, the telecom operators replace the proprietary hardware with commodity servers for hosting VNFs for the SFC. Before deploying telecom services on the cloud, the operator has to verify and validate if the cloud system is adequate for the carrier-grade services. According to the ETSI testing specification [27], both functionality and performance tests are the basis of the verification process on the NFV cloud. The functional tests aim to check if the system works correctly. After verifying that the system is operational, evaluating the performance of the system is also necessary. In many use cases, network performance is a critical concern in deploying and orchestrating cloud services. However, the performance is degraded due to the virtualization overhead. Telecom operators need a systematic approach to verify the network performance with different enhanced technologies of the cloud system.

In this regard, this paper investigates the enhancement of network techniques to mitigate the deterioration in a practical cloud system. A comprehensive testing framework is proposed to evaluate the cloud system with different EPA technologies in three areas, which are functionality, performance, and application. The functionality tests focus on instantiating VMs, building a tenant network, and the connectivity of VMs. In performance tests, both PVP and PVVP scenarios are considered to estimate the network throughputs. As for the application test, this paper not only deploys a virtual IP multimedia subsystem (vIMS) on the cloud, but also evaluates the performance of the vIMS via a session initiation protocol (SIP) stress benchmark. The performance metric of successful call rate is measured for the vIMS test. Empirical results show that the throughput of the PVP test performs near the line rate when applying the SR-IOV acceleration, while OVS-DPDK approaches 90% of the line rate when packet size is 1518. Increasing the number of lcores and pmd-cpus further enhances the performance by about 50% in OVS-DPDK. In addition, the successful call rates in vIMS validations with different techniques are similar in lower call rates. When the call rate exceeds 100, the improvement brought by OVS-DPDK is up to 14%. To sum up, the contributions of this paper are:

- Proposing a comprehensive testing framework to investigate a practical cloud system on different areas of performance effects.
- Conducting performance comparisons between the generic and the enhanced cloud system for NFV.
- Considering both PVP and PVVP packet paths for different SFC scenarios.
- Deploying the vIMS application on the enhanced cloud system to show effective performance.

The rest of this paper is organized as follows. Section 2 presents the background and discusses related works to support our research. The comprehensive testing framework and the test methodologies are proposed in Section 3. Section 4 introduces the experiments and results. Finally, Section 5 concludes this paper with summarizing remarks.

2. Background and Related Works

This section first introduces the background of enhancement technologies and then discusses the most relevant works to improve the performance of virtualization on the cloud.

2.1. Background

Regarding the virtualization of network functions, VM and container are two common technologies for running applications on a shared resource. The VM shares the underlying hardware via the VMM. Each VM is running with a guest operating system and dependencies of the application. On the other hand, the container simply packages the user program and corresponding dependencies to run the application. Multiple containers on the same host share the same application runtime environment of an operating system. Therefore, the container consumes less warmup time and fewer system resources [28]. However, running a container relies on sharing the host operating system, which may lead the system to a security issue. Most of the cloud service providers tend to provide the container service based on a VM for better isolation and easy management in the data center. Tackling the performance degradation of a VM is a pressing matter. Accordingly, this paper takes the VNF as an example to verify the proposed framework.

Regarding the enhanced technologies for the computation, EPA features involve typical mechanisms in the operating system such as CPU isolation, CPU pinning, and hugepages. The resource allocation scheme with a nonuniform memory access (NUMA) awarded topology is also considered as the performance impact in cloud computing. As for network improvement, many technologies are proposed for providing high-speed networking in the literature. This paper takes Open vSwitch (OVS), OVS-DPDK, and SR-IOV as examples for accelerating the packet processing in the cloud system. The basic concept of each enhanced technology adopted in this paper is introduced as follows.

- CPU Isolation

When the system administrator sets the isolated CPU mask, the system cannot assign regular jobs to the isolated CPU cores. In other words, each isolated CPU core remains idle if the system does not dispatch any specific process to the isolated CPU core. Therefore, the system administrator must set the isolated CPU properly; otherwise, the setting wastes computing resources for the system.

- CPU Pinning

CPU pinning enables the system to assign a job to specific CPU cores. In general, CPU isolation and CPU pinning are complementary. CPU isolation reserves a set of CPU resources, while CPU pinning allows the specific process to acquire the dedicated resources. If the system administrator enables these two settings, the VM process does not have to compete for CPU resources with others [29]. Accordingly, eliminating the CPU resource competition keeps the VM process away from the context switch overhead.

- Nonuniform Memory Access

NUMA is a memory architecture in a multi-CPU computer system. Figure 1 depicts a host consisting of two NUMA nodes. Each NUMA node has its CPU cores and memory resources. Memory access is faster if CPU cores access the memory on the same NUMA node. However, if CPU cores have to access the memory from another NUMA node, the crossing Quick Path Interconnect (QPI) access results in increasing access latency.

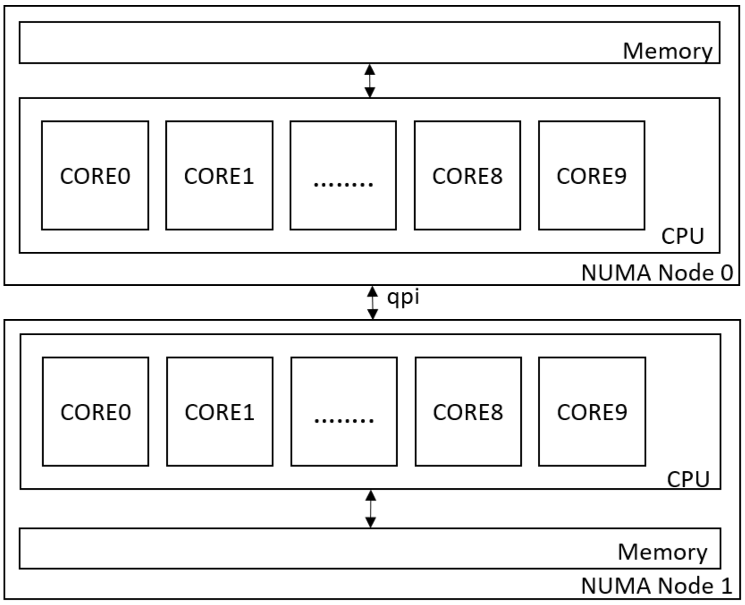

Figure 1. NUMA architecture.

- Hugepage

The page table in the memory stores the translation between logical and physical addresses. The system locates the memory address according to the page number and corresponding frame number in the page table. Accessing a particular memory address requires several mappings to find the accurate translation. A translation lookaside buffer (TLB) is responsible for speeding up the effective memory access via caching the recent address translation. The more the cache hits, the less access time there is for the memory addressing. Hugepage technology enables a large page size to provide more memory space for data access [30]. Therefore, the number of TLB misses and page faults is decreased, so as to accelerate the effective memory access.

- OVS

In the past, VMs did not share a virtio ring with the user space process in a host [31], i.e., vswitch. The packet processing had to be handled in the kernel space. Figure 2 illustrates the network architecture of OVS. ovs-vswitchd is used to set up the OVS data path. The configuration of ovs-vswitchd is stored in ovsdb. Therefore, ovs-vswitchd connects to the ovsdb-server to retrieve the configuration. When a cloud system adopts the OVS network, the OVS kernel module looks up the flow table when a packet is received. If the flow matches a network rule, the packet is processed according to the actions associated with the flow. When the packet does not match any rule in the flow table, the OVS kernel module queues the packet to the user space. This procedure introduces context switches and results in slowing down the packet processing.

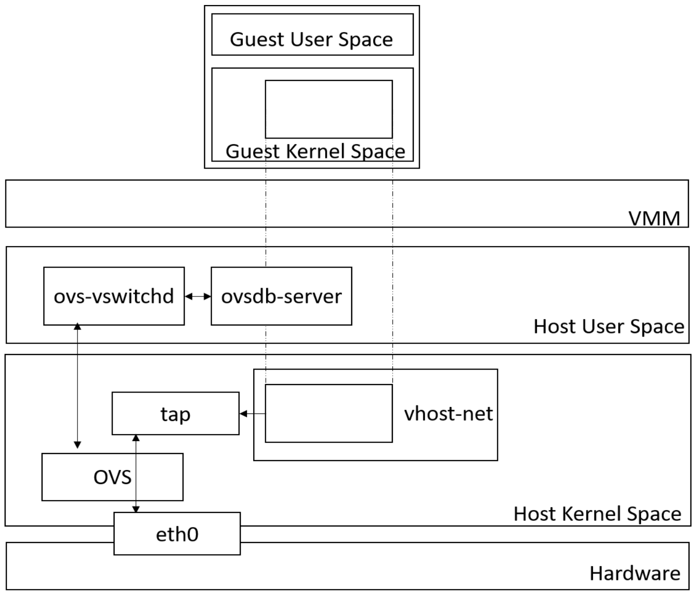

Figure 2. OVS architecture.

- OVS-DPDK

The data plane development kit (DPDK) aims to accelerate the packet processing. The network architecture of OVS-DPDK is shown in Figure 3. With the help of vhost-user, VMs share the memory space with user space processes in the host. Sharing memory between two processes is implemented by a file descriptor. The file descriptor can be accessed by the vhost process (i.e., virtual switch). The NIC driver, such as UIO or VFIO [32], is responsible for OVS-DPDK to handle the user space interrupt. Therefore, OVS-DPDK can process the packets in user space and pass them through the kernel. The kernel bypass eliminates the overhead of system calls and context switches. OVS-DPDK also provides a poll mode driver (PMD) [33] to poll the network queue. The dedicated CPU cores assigned to PMD are helpful to alleviate the interrupt overhead.

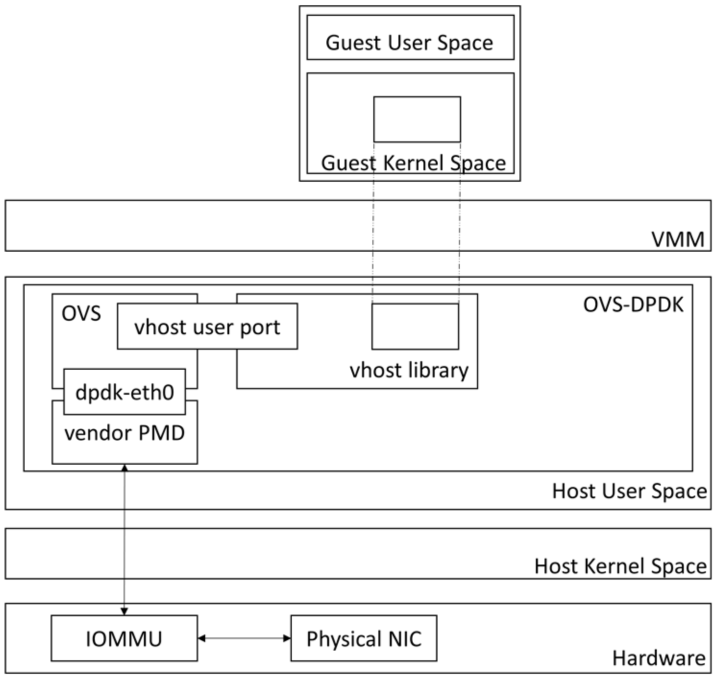

Figure 3. Network architecture of OVS-DPDK.

- SR-IOV

With the help of the input–output memory management unit (IOMMU), the VM can access Peripheral Component Interconnect (PCI) devices directly, called PCI passthrough. Direct memory access (DMA) remapping [34] allows the VM access to physical memory via the virtual address. When a VM accesses the network device, the interrupt remapping is triggered to send an interrupt to the VM instead of the host. With DMA and interrupt remapping, the network interface is assigned directly to a VM so that the VMM will not affect the performance.

In this regard, accessing the NIC directly to a VM can significantly improve the network performance. However, allocating a dedicated physical NIC for each VM is too expensive. The SR-IOV is proposed to tackle this problem. As shown in Figure 4, a NIC with SR-IOV support can provide physical function (PF) and virtual function (VF) [35]. The PF has all the functions of the NIC and is responsible for managing the VFs. The VF is a simplified PCIe function and allows the VM to access the physical NIC directly. Accordingly, the network performance of a VM can be improved in the cloud system. However, SR-IOV is a hardware-assisted technique, and it is not easy to live-migrate a VM with an SR-IOV VF. Procuring the specific NIC is also necessary while adopting the SR-IOV, so as to increase the cost of capital expenditure and management efforts for the cloud.

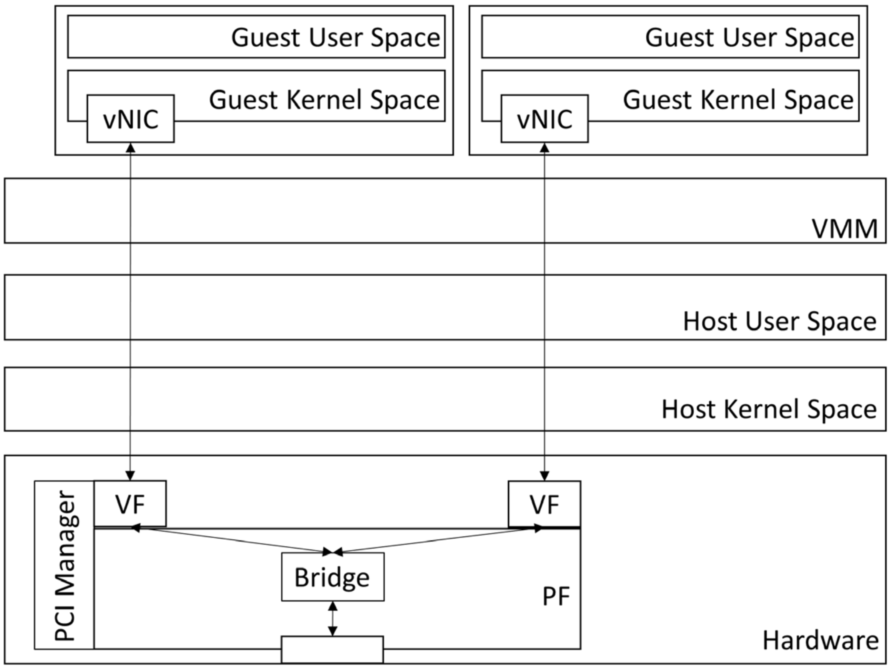

Figure 4. Network architecture of SR-IOV.

2.2. Related Works

Although NFV realizes flexible deployment, virtualization technology introduces the performance overhead. Rojas-Cessa et al. [36] estimated the performance of virtualized and nonvirtualized routers to show the degradation caused by virtualization. Their work revealed the performance impact of the number of cores and different packet sizes. The experimental results also demonstrate the benefit of using multicores. On the other hand, the performance degradation of traditional Linux I/O for high-speed networking has been discussed in previous works [37–39]. The main reasons are summarized as the long paths, the frequent interrupts, and the repeated data replications. Therefore, researchers try to reduce the duplication of packets between the physical NIC and the user space process. Previous works focus on reducing or avoiding hardware interruptions after the physical NIC receives packets. This paper attempts to investigate the enhanced technologies on packet acceleration for a cloud system.

In the past, the virtual switch ran as a user space process and could not share the memory with VMs. Owing to this limitation, the virtual switch had to receive packets from the kernel space. However, moving the packets between kernel space and user space introduces too many context switches to deteriorate the performance of VMs. To relieve the bottleneck, Virtual Open System Corp. designed the vhost-user for VM to share its memory with a virtual switch. Paolino et al. [31] then proposed a virtual switch, named Snabbswitch, and evaluated the performance of the virtual switch, such as OVS, OVS-DPDK, VFIO, and SR-IOV. In that work, the authors considered the scenarios of one VM and two VMs, in which each VM is enhanced with a 1G hugepage. However, only the process of Snabbswitch is allocated with the dedicated CPU cores. In addition, the experiments in their work were not conducted in a cloud system.

Pitaev et al. [40] compared the performance of three different I/O architectures: OVS-DPDK, FD.io VPP, and SR-IOV. The experimental environment contains a host and a packet generator. The target VM is deployed directly on the KVM hypervisor rather than a cloud system. The results show that SR-IOV is better than OVS-DPDK or FD.io VPP in IPv4 forwarding and the SFC scenario. In addition, the limitation of OVS-DPDK or FD.io VPP is that enough cores must be bound for PMD to maintain high throughputs. This constraint also restricts the maximum number of available VMs. The container technique is becoming more popular. G. Ara et al. [41] evaluated the network performance of various vswitches such as Linux-bridge, OVS-DPDK, VPP, and SR-IOV in containers. The results demonstrate that SR-IOV outperforms OVS-DPDK in terms of throughput.

R. Bonfiglia et al. [23] also tested the effects of OVS-DPDK and OVS on VMs and containers. The throughputs of a VM and a container are similar when both are applied to the OVS. In addition, the OVS-DPDK performs higher throughputs when compared to the OVS. Kourtis, M. A. et al. [22] went further to compare the network performance of a VNF with/without OVS-DPDK and SR-IOV. The authors used DPI as the VNF for testing. The results demonstrate that the throughput of a VM with OVS-DPDK or SR-IOV is higher than that of a VM without the OVS-DPDK or SR-IOV technique. Nevertheless, the previous works were not conducted on a cloud system. This paper tends to conduct experiments on a practical cloud system and verify the empirical results, which are rarely given attention.

The authors in [6,42] indicated that the VM in cloud computing has an issue of performance degradation. F. Callegati [43] et al. compared the performance of the Linux bridge and OVS in OpenStack. The authors found that the performance of the Linux bridge is worse than that of OVS. Tsai, M. H. et al. [21] discussed the effect of enabling EPA on a computing-intensive cloud system. They exploited CPU pinning and hugepages to improve the computing capability of VMs in NFVI. The results show that the computing performance of VMs could be enhanced by up to 7%. However, the performance of network enhancement techniques was not further discussed in the previous work.

To sum up, this paper synthesizes the relevant works on experimental conditions as shown in Table 1. The conditions include whether the enhanced techniques are applied or not, and whether the experiments are conducted in a cloud system. This table helps us find out where the discussion has been inadequate over the years. Based on the table, hardware accelerator architectures such as SR-IOV are rarely introduced for the verification on a cloud system. Even if the SR-IOV is adopted, the proposed experimental environment is not designed for ETSI NFV framework. Furthermore, EPA technologies enhance the performance of VNFs in either computing or networking. Previous work rarely considers the corresponding technologies at the same time. Therefore, this paper considers the OpenStack cloud as the system under test and attempts to assess the effect of EPA features for the NFV on a practical cloud system.

Table 1. Summary of related works.

Works	Hugepage	CPU Pinning	OVS	OVS-DPDK	SR-IOV	In Cloud
Tsai et al. [21]	v	v				v
Kourtis et al. [22]				v		
Gallenmuller et al. [24]				v		
Kawashima et al. [25]			v	v	v	
Huang et al. [28]			v		v	
Paolino et al. [31]	v		v	v	v	
Shanmugalingam et al. [38]			v	v		
Pitaev et al. [40]				v	v	
Ara et al. [41]				v	v	
Callegati et al. [43]			v			v
Ours	v	v	v	v	v	v

3. Methodology

This paper designs a comprehensive testing framework for the NFV cloud. The goal is to verify the performance effects of a practical cloud system with different enhanced technologies. This section first introduces the proposed framework and then discusses the key factors of the network performance.

3.1. Overview of Testing Framework

Figure 5 illustrates an implementation of the proposed test framework in this paper. This framework is aligned with the ETSI NFV framework and mainly contains NFVI, VIM, VNFM, NFVO, and test host. NFVI is composed of physical servers in the cloud system. The virtualization layer pools and extracts the physical resources into virtualized resources. The virtualized resources are allocated and managed by VIM. In this paper, OpenStack plays the role of VIM. The testing tools build the testing environment via OpenStack services. Most testing tools are installed in the test host, which is independent of the nodes of a cloud system. Therefore, the testing is triggered on the host and connected to OpenStack services via the external network and a TOR switch. Only if the tools are authorized with a credential file are the test processes then activated and verified on the OpenStack system. According to the ETSI TST specification, the proposed framework considers a complete cloud system under test. The verifications of the cloud SUT emphasize three areas: functionality, performance, and application testing.

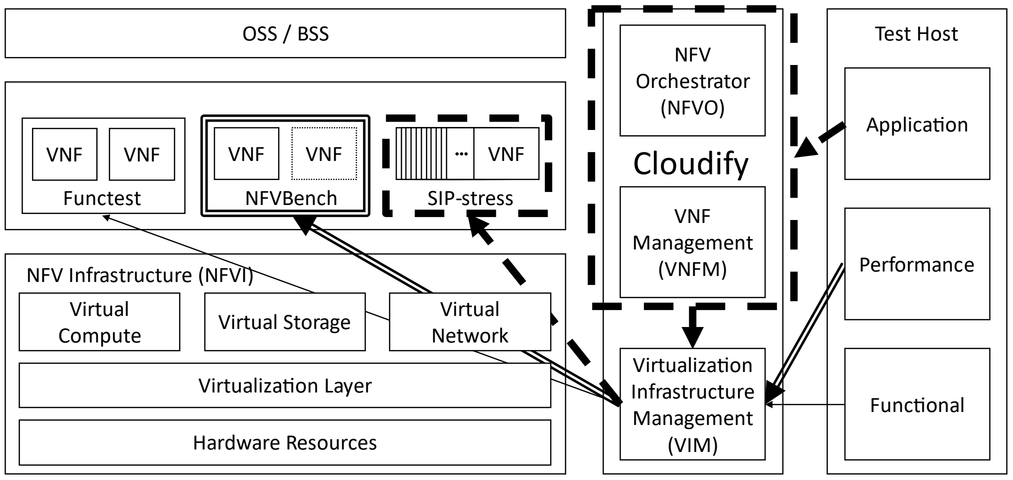

Figure 5. Proposed testing framework for comprehensive verifications on the NFV cloud.

3.1.1. Functionality Verification

In the functionality test, the verification focuses on testing the basic operations of a cloud system. The main goal is to verify the core service operations in a cloud system via a set of test tools. For example, the system under test in this paper is an OpenStack cloud. The Functest [44] provides a set of test cases to verify core services for operating the OpenStack cloud. If the services of OpenStack are active properly, the testing process further evaluates the operating performance of each service, e.g., the turnaround time of each test case. This performance evaluation checks if OpenStack services finish the specific operations in time. If the operations break the time limits, the relevant services work incorrectly. This paper applies Rally Sanity to check if the services fulfill the predefined SLA.

For the test cases in Functest, this paper adopts Healthcheck and Rally_Sanity to test the primary functionalities of an OpenStack cloud system. In Healthcheck test suite, Connection_check is used to check the connectivity of OpenStack Endpoints. Vping_ssh tests whether the VM can be accessed via SSH from the external network. Vping_userdata is used to verify if the VMs can communicate with each other through the internal network. Rally_Sanity evaluates the operating performance of OpenStack services. The primary services include Nova, Neutron, Cinder, Keystone, and so on. Nova is used to provide computing instances. Neutron is responsible for the networking as a service. Cinder provides the block storage service. Keystone is used to provide the authentication service. These test cases of Rally_Sanity measure the time taken by each scenario to make sure that all tests fulfill the SLA.

3.1.2. Performance Verification

In the performance test of a SUT, the verification focuses on testing the network performance of VNFs. The main goal is to benchmark the target VNF and evaluate the performance metric. This paper adopts NFVBench [45] to evaluate the throughput of VNFs on the SUT. In NFVBench, TRex is applied as a traffic generator and a TestPMD as a target VNF to forward the received packets. When launching the performance test, TRex sends packets from the test host to the target VNF. Before reaching the VNF, the packets are transmitted throughout the TOR switch and the OpenStack cloud system. VNF then uses L2 Forward to send the packets back to the traffic generator.

In practice, a telecom application usually deploys multiple VNFs and connects these VNFs to the SFC. When the number of VNFs in the SFC increases, the packet path of this SFC is longer. The length of a packet path has an impact on the network performance. This paper considers different traffic flows for performance test scenarios: PVP and PVVP tests. The experiment of the PVP scenario is used to estimate the performance of end-to-end traffic. As shown in Figure 6, the traffic generator installed on the test host sends out the packets to the target VM deployed on a cloud system. When the VM receives packets, the VM sends the packets back to the traffic generator through the TOR switch.

The PVVP scenario is conducted for the multiple VNFs of a simple SFC scenario as shown in Figure 7. The traffic generator sends out the packets to the first VM. When the first VM receives packets, the first VM forwards the packets to the other VM via the east–west traffic in OpenStack. The VM then sends the packets back to the traffic generator through the TOR switch. Lengthening the packet path increases network communications in the data plane. If a component in the packet path cannot handle the packets, the throughput worsens. Therefore, this experiment involves both evaluations of the north–south traffic and the east–west traffic in the SUT.

3.1.3. Application Verification

Regarding the application test, the verification focuses on deploying the NFV application on the cloud system. The main goal is to validate the feasibility and effectiveness of SUT. This paper applies the Clearwater vIMS to act as the telecom service. The architecture of Clearwater vIMS is shown in Figure 8.

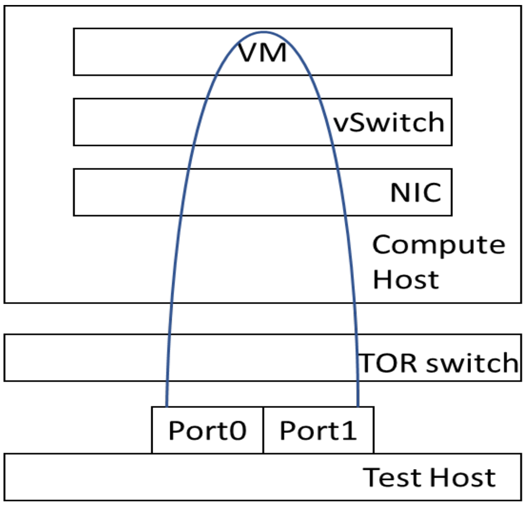

Figure 6. Packet path in PVP scenario.

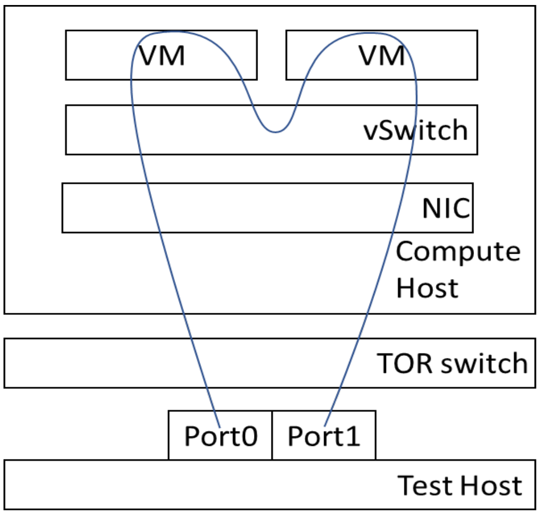

Figure 7. Packet path in PVVP scenario.

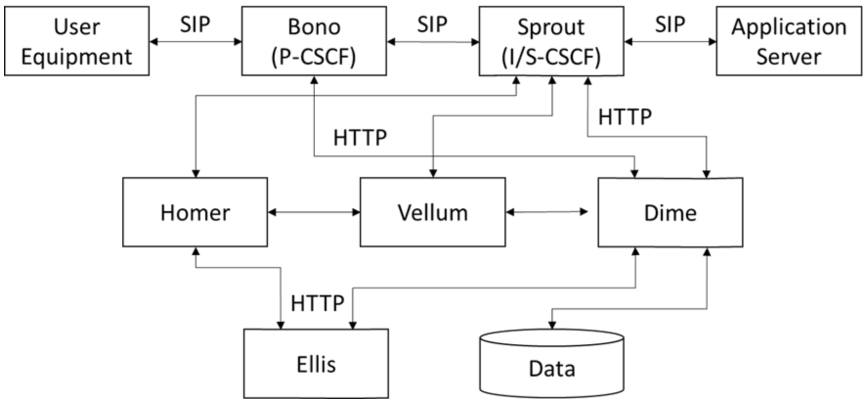

Figure 8. Architecture of Clearwater vIMS.

The main function of Bono is Proxy Call Session Control Function (P-CSCF). P-CSCF acts as an edge proxy and is responsible for the network address translation and the connection between the user equipment and the Clearwater system. Sprout is the SIP route in Clearwater. Sprout is responsible for user authentication and SIP registration. The major functions of Sprout are the Interrogating Call Session Control Function (I-CSCF) and the Serving Call Session Control Function (S-CSCF). S-CSCF manages and controls the communications, while I-CSCF sends SIP requests to the appropriate S-CSCF according to the information stored in Homer. Application servers provide advanced features. The features include call forwarding, call recording, etc. Homer is used to store the settings files of the application service. Dime acts as a diameter gateway and is composed of Homestead and Ralf. Ralf is used to trigger functions, while Homestead is an HSS cache for storing user information. Vellum is responsible for the state storing and exploits Cassandra to store all long-lived data. Lastly, Ellis is responsible for user sign-up and password management.

In this paper, the SIP stress is used to benchmark the vIMS application. SIP stress simulates the phone calls among multiple user equipments (UEs) and records the numbers of outgoing calls, successful calls, failed calls, etc. The successful call rate is measured as the performance metric of vIMS. A higher successful call rate indicates better performance. To deploy the Clearwater vIMS on the cloud system, this paper adopts Cloudify Manager [46] to manage and orchestrate VNF resources in OpenStack. Deploying vIMS requires customized configurations according to the SUT environment, e.g., authentication endpoint, credential file of the cloud, VM flavor, connection keypair, etc. After setting the corresponding information of the SUT, Cloudify Manager deploys the vIMS template according to the configurations. After the deployment is successful, the SIP stress is activated.

3.2. Verification Procedures and Factors

The testing framework is made up of open-source tools and is easy to reproduce. First, the testing tools are containerized and can be run as docker instances on the test host. Second, each verification has to set the configuration file according to the SUT environment before running the test case. Finally, the testing tools are launched to perform the SUT test. In some cases, the SUT system requires some adjustments according to a specific deployment of the cloud system. For example, the authentication endpoint varies among different cloud systems. The testing should modify the credential file according to the cloud system. As another example, enabling DPDK requires a dedicated network interface. The SUT system has to attach network interface to the provider network with OVS-DPDK.

Figure 9 depicts an overview of the verification procedure in the proposed test framework. Before deploying the SUT system, the first step is to plan and determine which kinds of EPA features should be enabled on an OpenStack cloud for verification. The second step is to deploy the OpenStack cloud with corresponding configurations to enable the accelerating technologies for the SUT test. The third step is verifying and evaluating the cloud systems with different benchmarking tools. The last step is collecting and validating the test results. The verification process will be ended until all the enhanced features are verified and accomplished.

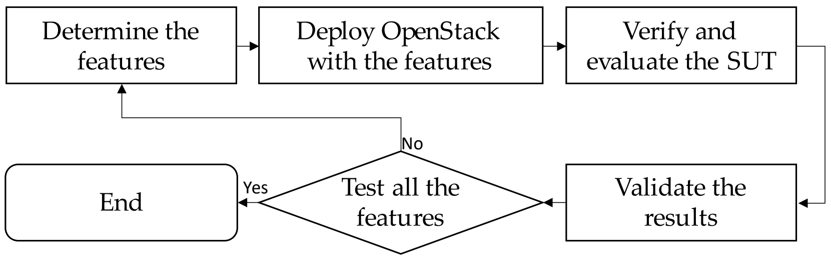

Figure 9. Verification procedures.

Once the deployment of a customized OpenStack is complete, the verification and evaluation process tests the cloud system in terms of functionality, performance, and VNF application. The functionality tests verify the general operations of a cloud system and the connectivity of VMs. If the basic functions operate correctly, the next step is to evaluate the performance test of VNF networks on the cloud. Afterward, the application-level test is optional to deploy the SFC service and trigger the corresponding tools for a stress test. To evaluate the impacts on the SUT verification, this paper conducts the verification factors from different perspectives: packet size, VM size, and call rate.

- Packet Size

Different applications have different requirements for performance characteristics. The difference in these characteristics requires different packet sizes in various applications [47]. When the header of each packet is the same, the larger payload results in a larger packet. The application of interactive video often requires low latency. Therefore, the large packet size is not suitable for interactive video applications. However, the application using small packets tends to send out huge amounts of packets. If the virtual switch cannot afford the massive packets, some packets are dropped and the throughput is decreased. In this paper, the test framework adopts the NFVBench tool to generate PVP and PVVP traffic to investigate the throughput impacts on different packet sizes.

- VM Size

In general, packet processing requires CPU resources. For example, a VM running the VNF uses two cores to forward the packets. The other jobs also consume CPU resources on the same VM. If the target VNF only has two vCPU cores, the CPU resources are not sufficient for packet processing. When the number of vCPU cores is too small, the VM is not able to handle all the packets. As a result, the dropped packets deteriorate the network throughputs. Therefore, this paper compares the throughputs of 2-core VM and 4-core VM with OVS-DPDK. This evaluation tool in this experiment is also NFVBench. This experiment focuses on the importance of resource allocation. In this paper, CPU pinning and CPU isolation techniques are applied to limit the number of dedicated cores.

- Call Rate

In the application test, this paper applies the vIMS as a VNF application on the cloud. The test not only deploys cloud vIMS with different enhanced features, but also adopts the SIP stress test to evaluate the performance of the application. Figure 10 depicts the sequence of a phone call in this experiment. First of all, UE1 sends out the invite request to initiate a call session. The 100 trying prevents the retransmissions of invite requests. When the UE2 receives the invite request, UE2 sends the 180 ringing back and alerts the user. If the user answers the phone call, UE2 sends a 200 OK. When UE1 receives the 200 OK from UE2, UE1 replies to UE2 with an ACK. After these steps are accomplished, the call session is established. When the conversation is finished, both UE1 and UE2 hang up the phone call by sending the BYE message. When a UE receives a BYE message, the UE replies the 200 OK to the BYE. Afterward, the session is terminated. Throughout the experiment, the successful call is measured to indicate how many packets are processed without a loss. The growth of the call rate increases the workload of the application. The successful call rate is declined when the cloud vIMS cannot afford the workload.

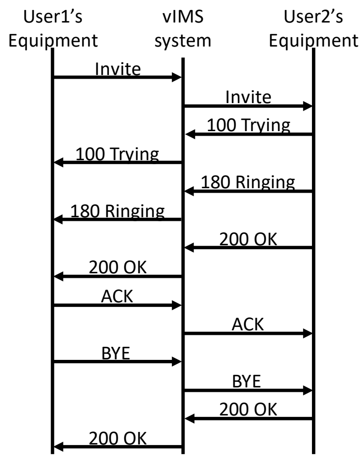

Figure 10. SIP call flow.

4. Experiments and Evaluations

In NFV applications, network performance is a critical concern, and enhanced technologies are proposed to improve packet processing. These technologies bring different effects of performance improvement to a cloud system. This section firstly verifies the functional operations of OpenStack cloud and then evaluates the performance of the clouds with different network enhancement techniques. Lastly, the vIMS application is deployed to validate effective performance. The cloud system applying the OVS networking is a generic case to act as the baseline in the experiment. To further improve the performance, the cloud system enables CPU pinning and hugepages while applying network-accelerating technologies such as OVS-DPDK or SR-IOV.

4.1. Testbed Environment

Table 2 summarizes the host configurations of the testbed. The test host is equipped with Intel i7-3770 and 16 G memory. Both control and compute hosts have two Xeon Silver CPUs and 160 G memory. In addition, all of these hosts have one Intel XXV710 as the network interface and each host is interconnected via a 10 Gbps TOR switch. The testing tools are installed in the test host. The OpenStack cloud system is composed of two nodes in the testbed. The first node acts as the control host for providing the core OpenStack services, such as Keystone, Nova API, Nova conductor, Neutron, and Glance. The other node is a compute host for launching VM resources via Nova Hypervisor.

Table 2. Host configurations of the testbed.

	Test Host	**Control Host**	**Compute Host**
CPU	Intel i7-3770	Xeon(R) Silver 4210 × 2	Xeon(R) Silver 4210 × 2
Cores	4	10 × 2	10 × 2
Memory	4 G × 4	16 G × 10	16 G × 10
Host OS	Ubuntu 18.04 Server	Ubuntu 18.04 Server	Ubuntu 18.04 Server
Physical NIC	XXV710	XXV710	XXV710

Table 3 summarizes the settings in the performance experiments, such as network enhancement techniques, scenarios, packet sizes, and enhanced configurations. In performance testing, this paper focuses on four experiments. The first test compares the network performance of the cloud systems with different network acceleration techniques. The second experiment shows the performance of different traffic flows. The traffic flows include PVP and PVVP scenarios. The third experiment evaluates the throughputs with different packet sizes. The fourth experiment shows the impact of different VM sizes. Because OVS-DPDK requires additional configurations of lcore and pmd-cpu, the last experiment investigates the performance impact of the number of lcores and pmd-cpus in OVS-DPDK tests.

Table 3. Experimental settings in performance tests.

Network Technique	**Generic OVS/OVS-DPDK/SR-IOV**
Scenario	PVP/PVVP
Packet Size	64/128/256/512/1024/1280/1518
vCPU (DPDK)	2/4
lcore + pmd-cpu (DPDK)	2 + 2/4 + 4

Table 4 summarizes the experiment to estimate the performance of vIMS with different network techniques. The vCPU of a VM is set to 1 or 2. The call rate varies from 80 to 140. In the OVS-DPDK scenario, all the components are deployed with the features of CPU pinning and hugepages. This experiment also considers the impact of lcore and pmd-cpu.

In this experiment, the pmd-cpu and lcore are set to 4. Each NUMA node has 2 lcores and 2 pmd-cpus.

Table 4. Experimental settings in application tests.

Enhancement Tech	Generic OVS/OVS-DPDK
vCPU	1/2
Call Rate	80/100/120/140
lcore + pmd-cpu (DPDK)	4 + 4

4.2. Results of Functional Verifications

After the deployment of a cloud system, the first experiment is to verify the operation of the SUT. All test cases should be passed if no error occurs during the functional tests. The results of the functionality test are summarized in Table 5. The test cases of Healthcheck and Rally_Sanity are shown in Appendices A and B. The result of Healthcheck verifies all the operations of OpenStack are correct. Rally_Sanity checks if all the tests meet the requirements of the SLA. Because all the scenarios pass the tests of Rally_Sanity, the OpenStack services can finish their jobs in time.

Table 5. Result of the functionality test.

Test Suite	Pass/Total	Result of Test Cases
Healthcheck	9/9	Appendix A
Rally_Sanity	56/56	Appendix B

4.3. Results of Performance Verifications

This section presents the performance evaluation and analysis. Firstly, the performance results of different network techniques are evaluated in the experiments. Afterward, the impacts of each verification factor are discussed in terms of packet size, VM size, traffic flow, and enhanced data plane.

4.3.1. Different Network Techniques

The experiment uses a VM with two cores and different improvement technologies. The test scenario is PVP. As a baseline, the VM in the OVS scenario does not apply enhancement techniques, while the others enable CPU pinning or hugepages. Experimental results in different packet sizes are shown in Figures 11–13. The reason behind each experiment is discussed as follows.

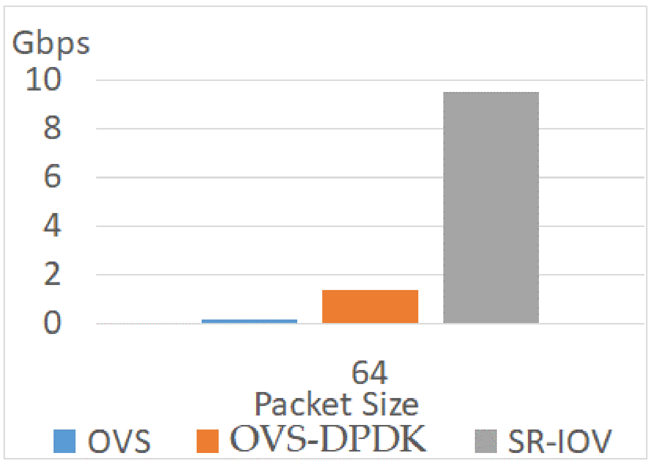

Figure 11. Throughputs of different technologies in PVP scenario when packet size is 64.

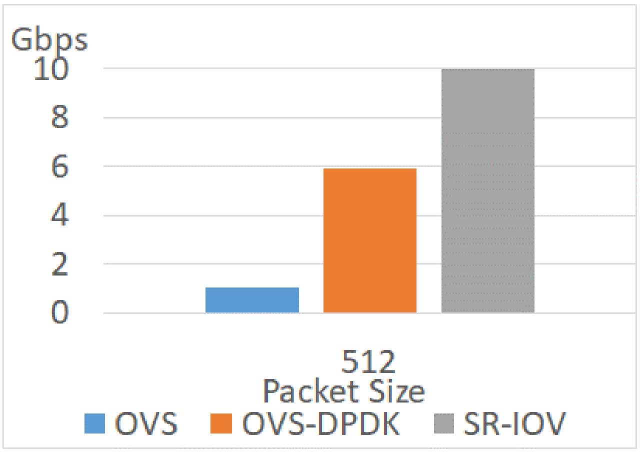

Figure 12. Throughputs of different technologies in PVP scenario when packet size is 512.

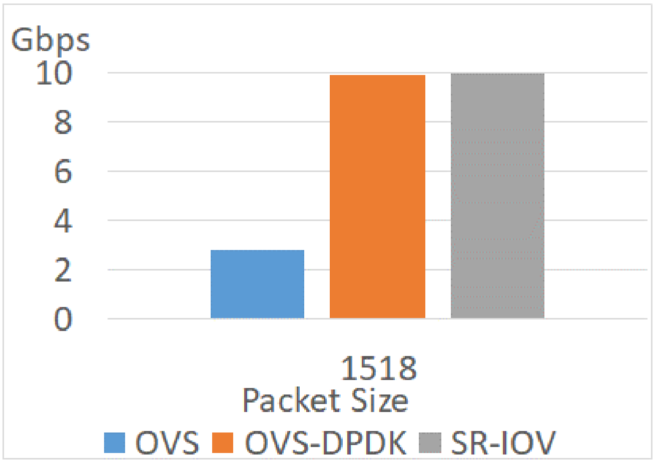

Figure 13. Throughputs of different technologies in PVP scenario when packet size is 1518.

Because OVS receives packets in kernel space, this method still needs a context switch if a packet does not match any rules. The kernel-user space interrupt degrades the network performance. As shown in these three figures, the throughputs with OVS are the lowest. When the packet size is 64, the throughput is only 0.16 Gbps. The throughput varies with the increment in packet size. The maximum throughput is 2.8 Gbps when the packet size is 1518. These results show a low performance without any enhancement.

The difference between OVS and OVS-DPDK is to exploit vhost-net or vhost-user for NIC emulation. OVS uses vhost-net to enable the virtual switch to process the packets in kernel space. This method introduces the overhead of context switches and deteriorates the performance. On the contrary, OVS-DPDK uses vhost-user to enable the guest to communicate with the virtual switch. Therefore, OVS-DPDK mitigates the performance degradation. On the other hand, the VM with OVS is not improved with CPU pinning and hugepages. CPU pinning helps the VM possess dedicated cores. These cores avoid the VM sharing CPU resources with other processes in the host. Therefore, the overhead of context switches is relieved. Hugepages are used to speed up memory access.

Considering the enhancement of the virtual switch, the throughputs of using OVS-DPDK are better than that of OVS. The throughput of OVS-DPDK is 1.4 Gbps when the packet size is 64. This throughput of OVS-DPDK is 8.2 times as high as the throughput of OVS. The maximum throughput with OVS-DPDK is 9.9 Gbps when the packet size is 1518. This value is 3.5 times as high as the throughput with OVS. Therefore, the throughput of OVS-DPDK ranges from 3.5 to 8.2 times as high as the OVS. However, the throughputs with OVS-DPDK are lower than a hardware-based solution.

SR-IOV is the hardware-assisted solution. This technique improves network performance significantly and allows a VM to access the physical NIC directly via the virtual function. The throughput of SR-IOV approaches 9.5 Gbps when the packet size is 64. When the packet size is higher than 64, the throughputs are 9.9 Gbps. As shown in the results, the performance of SR-IOV is near the line rate. With SR-IOV, the system assigns dedicated virtual functions to the VMs. These virtual functions help the VM access the physical NIC directly. Therefore, the impact of VMM is relieved.

4.3.2. Different Packet Sizes

As shown in Figures 11 and 13, the throughputs of OVS are 0.16 Gbps and 2.8 Gbps when the packet sizes are 64 and 1518. In the OVS network, the throughput in packet size 1518 is 16.4 times as high as the throughput in packet size 64. In addition, the throughput of OVS-DPDK in packet size 64 is only 1.4 Gbps. When the packet size is 1518, the throughput of OVS-DPDK is 9.93 Gbps. In the OVS-DPDK network, the throughput in packet size 1518 is 7 times as high as the throughput in packet size 64. Regarding SR-IOV, the throughputs are 9.5 Gbps, 9.96 Gbps, and 9.96 Gbps when the packet sizes are 64, 512, and 1518. Therefore, the impact of packet size is less significant when packet processing capacity is higher.

Because the VM accesses the physical NIC directly, the performance of SR-IOV is near the line rate in all cases of different packet sizes. On the contrary, the throughputs of OVS and OVS-DPDK dramatically decrease when the packet size is small. That is because the sender generates more packets when the packet size is small. A large workload of packet processing is not affordable for virtual switches because the capacity of each virtual switch for processing packets is limited. The more packets per second the sender generates, the more packets are dropped. Therefore, the throughput of small packet size is lower than that of a large one.

4.3.3. Different VM Sizes

Considering that the packet processing consumes CPU resources, the next experiment tests the throughputs of VMs with different numbers of cores. Because the throughput of the VM with SR-IOV is too high while the throughput of the VM with OVS is too low, this experiment only presents the performance of applying OVS-DPDK, as shown in Figure 14. Results show that the throughput grows in all packet sizes when the VM size increases.

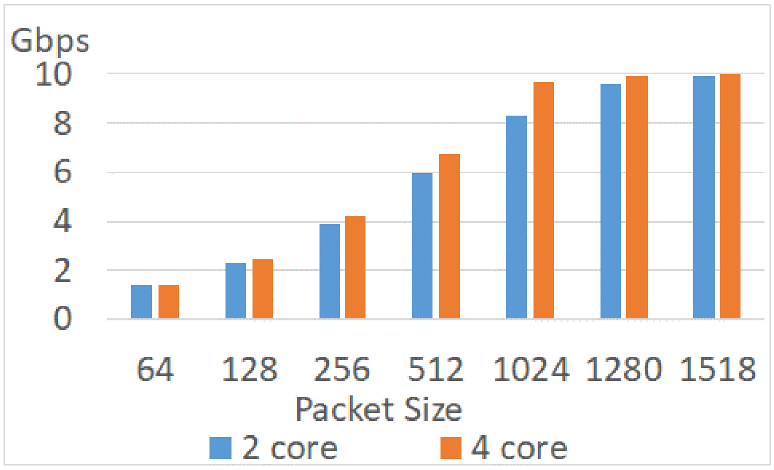

Figure 14. Throughputs of VMs with different cores in the PVP scenario.

The throughputs of a 2-core VM are 1.4 Gbps, 5.9 Gbps, and 9.93 Gbps when the packet sizes are 64, 512, and 1518, respectively. Regarding a 4-core VM, the throughputs are 1.42 Gbps, 6.7 Gbps, and 9.96 Gbps when the packet sizes are 64, 512, and 1518, respectively. The improvement ranges from 0.2% to 16% in this experiment. The reason is that if the VM only has two CPU cores, the preemption occurs frequently while the

CPU core is processing. Therefore, the throughput is decreased if the CPU resources are limited. Increasing the number of CPU cores relieves the overhead of preemption so that the throughput is enhanced.

4.3.4. Different Traffic Flows

The experiment deploys two VMs with different improvement technologies to evaluate the throughput, as shown in Figures 15–17. The SR-IOV brings the best performance among all technologies. The throughputs of OVS and OVS-DPDK increase when the packet is larger. However, the packet path in this experiment is longer than that of the PVP scenario. If the traffic flow is longer, the workload of the components in this path is too heavy to decline the throughputs. When the packet sizes are 64, 512, and 1518, the throughputs of OVS are 0.147 Gbps, 1 Gbps, and 2.58 Gbps, respectively. The throughputs are 0.8 Gbps, 3.5 Gbps, and 6.2 Gbps of OVS-DPDK when the packet sizes are 64, 512, and 1518, respectively. Therefore, in the PVVP scenario, the improvement brought by OVS-DPDK ranges from 2.4 to 5.5 times the OVS. Regarding the SR-IOV, the throughputs are above 8 Gbps in all packet sizes. However, the throughputs of SR-IOV are still the best.

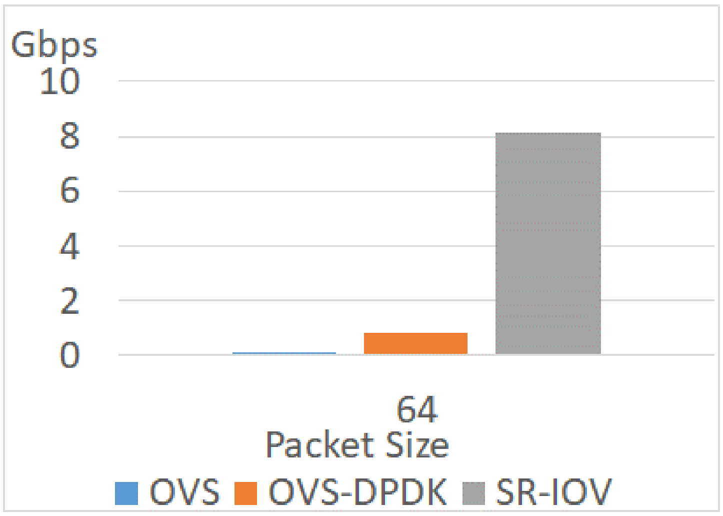

Figure 15. Throughput of different technologies in PVVP scenario when packet size is 64.

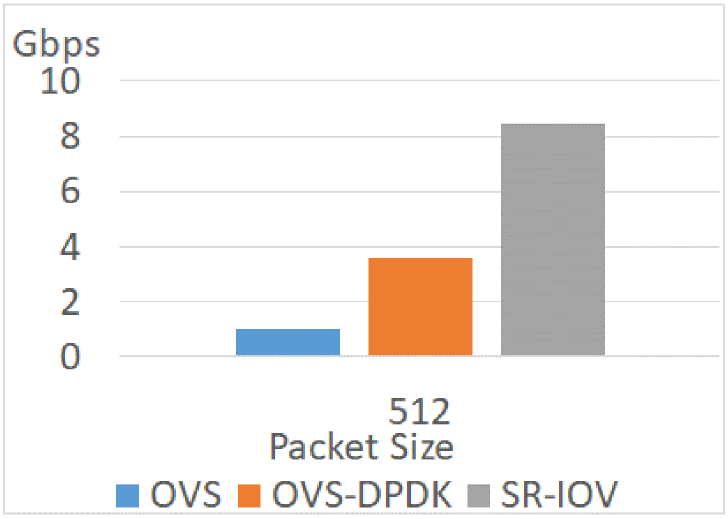

Figure 16. Throughput of different technologies in PVVP scenario when packet size is 512.

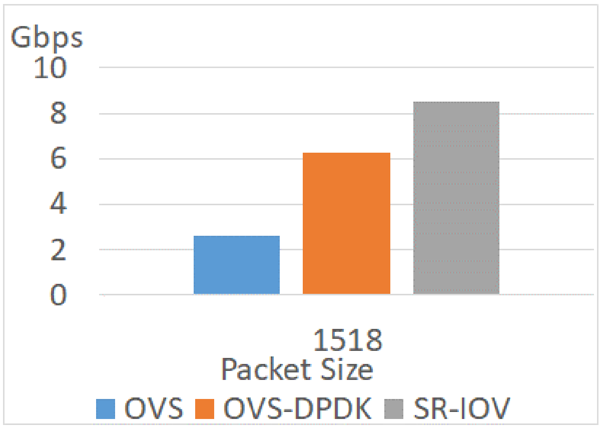

Figure 17. Throughput of different technologies in PVVP scenario when packet size is 1518.

Regarding packet loss, the loss rate of OVS or OVS-DPDK decreases when increasing the packet size. Meanwhile, SR-IOV has the lowest drop rate among the other network technologies. The reason is that SR-IOV mitigates the interference of VMM. Therefore, VM can access the physical NIC directly. On the other hand, the OVS has the highest drop rate. That is because OVS introduces a lot of context switches between the kernel-user space. OVS-DPDK mitigates the context switches between the kernel-user space and polls the queues of NICs with dedicated cores. Therefore, the drop rate of OVS-DPDK is lower than that of OVS.

4.3.5. Enhanced Data Plane

Although OVS-DPDK eliminates the context switch in the kernel-user space, the performance is still affected by the configurations of OVS-DPDK. Applying OVS-DPDK requires CPU resources for lcore and pmd-cpu. The lcore is used by non-datapath threads to remove the idle and wrong flows while the pmd-cpu needs to poll the queues of physical NICs and virtual NICs. If the number of pmd-cpu is too small, the lack of pmd-cpu will decrease the throughput. Therefore, this experiment is to verify the performance impact of lcore and pmd-cpu to enhance the data plane.

As shown in Figure 18, the throughput increases when the number of lcore and pmd-cpu grows. When the packet sizes are 64, 512, and 1518, the throughputs of 2 lcores and 2 pmd-cpus are 1.4 Gbps, 5.9 Gbps, and 9.9 Gbps. Under the same packet sizes, the throughputs of 4 lcores and 4 pmd-cpus are 1.9 Gbps, 9 Gbps, and 9.9 Gbps. Therefore, the throughputs of 4 lcores and 4 pmd-cpus are improved by up to 50%. Furthermore, the enhancement of increasing lcore and pmd-cpu is even more obvious than increasing the VM size. The reason is that each VM consumes the virtual NIC resource and increases the workload of pmd-cpu. When the number of pmd-cpus is small, the pmd-cpu may not process so many packets. Thus, the packets are dropped if pmd-cpus are not affordable.

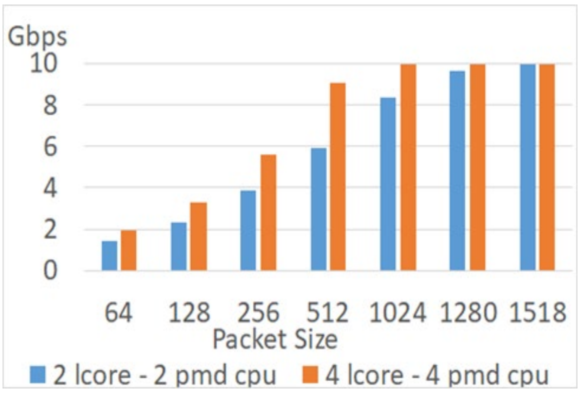

Figure 18. Throughputs of OVS-DPDK when lcore and pmd-cpu increase.

4.4. Results of Application Verifications

After testing the network performance of a SUT, this experiment deploys vIMS as a cloud application to show the improvement of a virtual switch. Subsequently, the SIP stress is installed to benchmark the successful call rate (SCR) of the vIMS with different call rates. A higher successful call rate means better performance.

In this experiment, each component of vIMS has only 1 vCPU. As shown in Figure 19, the successful call rate declines along with the increment in the call rate. When the call rate is lower than 100, the workload is affordable for both OVS and OVS-DPDK. Therefore, the successful call rates are higher than 95%. In addition, the successful call rate is significantly improved from 72% to 86% by OVS-DPDK when the call rate is 120. That is because the components of vIMS with OVS-DPDK are enhanced by DPDK, CPU pinning, and hugepages. These technologies improve both networking and computing. Therefore, the successful call rate with OVS-DPDK performs better than OVS.

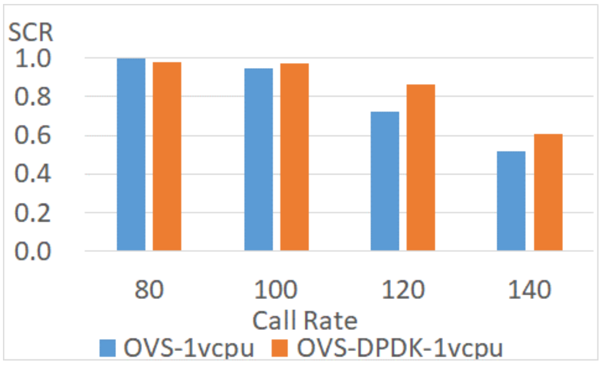

Figure 19. Successful call rate of vIMS with OVS and OVS-DPDK.

When the call rate is higher than 120, the missing call rate of OVS ranges from 27% to 48%. The reason is that interactive video applications tend to use smaller packets [47]. When many small packets are transmitted to the application, the workload of packet processing is increased. Therefore, the missing call rate also increases when the virtual switch cannot afford the loading. While applying OVS-DPDK, the missing call rate is reduced by up to 14%. The next experiment further improves the performance of vIMS with OVS-DPDK in different VM sizes. Previous experiments illustrate that the VM size brings impacts on the performance of VNF. Therefore, this experiment increases the number of CPU cores for each component in vIMS. As shown in Figure 20, the vIMS performance is improved if the VM size is increased to 2 CPU cores. The successful call rate increases from 61% to 79% when the call rate is 140, which means that the enhancement is 18%.

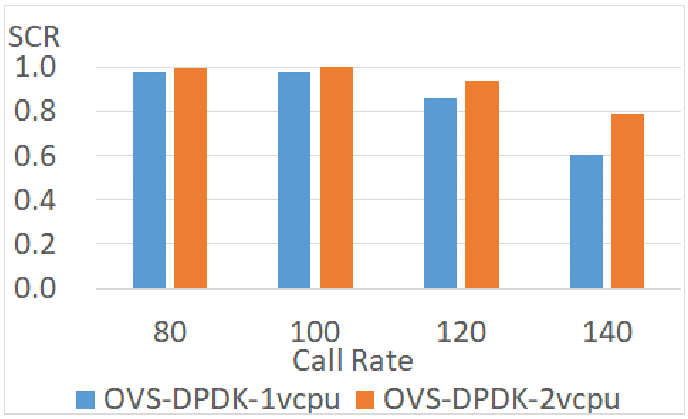

Figure 20. Successful call rate of vIMS with different VM sizes using OVS-DPDK.

5. Conclusions

As ETSI proposes the NFV architecture using virtualization technologies, the performance degradation of virtualization is a challenge to meet the SLA of the telecom industry. This paper attempts to conduct the performance effects of EPA techniques for the NFV cloud. The EPA techniques adopted in our testbed include CPU pinning, CPU isolation, hugepages, OVS-DPDK, and SR-IOV. This paper further proposes a test framework for the comprehensive verification and evaluation of the cloud system under test in terms of functionality, performance, and application. The major advantages of this framework are threefold. First, this framework is based on open-source projects to eliminate the expenditure on system verification and evaluation. Second, the framework is easy to implement and expandable. The testing procedures are triggered on the test host with the test containers for different benchmarking tools. Third, for the performance testing, the verification not only compares the impacts of different network techniques, but also considers the length of packet path, different VM sizes, and performance tuning of packet processing, e.g., lcore and pmd-cpu.

Results show that all the cases in functional tests are passed. The test cases in Rally_Sanity also fulfill the requirement of SLA. Throughputs of OVS-DPDK are up to 8.2 times as high as that of generic OVS networking. Regarding the performance at the application level, the SCR of vIMS is improved by up to 14% with OVS-DPDK. Our experiments further obverted that the throughput is limited when the packet size is small. This deterioration happens when the virtual switch cannot process a huge number of packets. The throughput declines when the packets are dropped. Regarding the end-to-end scenarios, the length of a traffic flow in PVP or PVVP has different impacts on the throughput with different networking techniques. The results and findings of this work are valuable for further reference in workload scheduling, resource placement, or load-balancing mechanisms. To extend this work, deploying a feature cloud and verifying the SUT automatically are interesting works. Further research will be to model the specific application performance based on the proposed testing method and the enhanced technologies, which are also left for future work.

Author Contributions: Conceptualization, W.-C.C.; Funding acquisition, W.-C.C.; Methodology, W.-C.C. and Y.-H.W.; Software, Y.-H.W.; Supervision, W.-C.C.; Validation, Y.-H.W.; Writing—review & editing, W.-C.C. and Y.-H.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministry of Science and Technology of Taiwan, under Grants MOST108-2218-E-033-008-MY3 and MOST111-2221-E-033-033.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Result of Healthcheck.

Test Cases	Project	Tier	Result
connection_check	functest	healthcheck	PASS
tenantnetwork1	functest	healthcheck	PASS
tenantnetwork2	functest	healthcheck	PASS
vmready1	functest	healthcheck	PASS
vmready2	functest	healthcheck	PASS
singlevm1	functest	healthcheck	PASS
singlevm2	functest	healthcheck	PASS
vping_ssh	functest	healthcheck	PASS
vping_userdata	functest	healthcheck	PASS

Appendix B

Table A2. Result of Rally_Sanity.

Scenario	Errors	Success (SLA)
Authenticate.keystone	0	✔
Authenticate.validate_cinder	0	✔
Authenticate.validate_glance	0	✔
Authenticate.validate_heat	0	✔
Authenticate.validate_neutron	0	✔
Authenticate.validate_nova	0	✔
CinderQos.create_and_list_qos	0	✔
CinderQos.create_and_set_qos	0	✔
CinderVolumes.create_and_delete_snapshot	0	✔
CinderVolumes.create_and_delete_volume	0	✔
CinderVolumes.create_and_delete_volume-2	0	✔
CinderVolumes.create_and_delete_volume-3	0	✔
CinderVolumes.create_and_extend_volume	0	✔
CinderVolumes.create_from_volume_and_delete_volume	0	✔
CinderVolumeTypes.create_and_list_volume_types	0	✔
CinderVolumeTypes.create_volume_type_and_encryption_type	0	✔
GlanceImages.create_and_delete_image	0	✔
GlanceImages.create_and_list_image	0	✔
GlanceImages.create_image_and_boot_instances	0	✔
GlanceImages.list_images	0	✔
HeatStacks.create_check_delete_stack	0	✔
HeatStacks.create_suspend_resume_delete_stack	0	✔
HeatStacks.create_update_delete_stack	0	✔
HeatStacks.list_stacks_and_resources	0	✔
KeystoneBasic.add_and_remove_user_role	0	✔
KeystoneBasic.create_add_and_list_user_roles	0	✔
KeystoneBasic.create_and_delete_role	0	✔
KeystoneBasic.create_and_delete_service	0	✔
KeystoneBasic.create_and_list_tenants	0	✔
KeystoneBasic.create_and_list_users	0	✔
KeystoneBasic.create_tenant	0	✔
KeystoneBasic.create_tenant_with_users	0	✔
KeystoneBasic.create_update_and_delete_tenant	0	✔
KeystoneBasic.create_user	0	✔
KeystoneBasic.get_entities	0	✔
NeutronNetworks.create_and_delete_networks	0	✔
NeutronNetworks.create_and_delete_ports	0	✔
NeutronNetworks.create_and_delete_routers	0	✔
NeutronNetworks.create_and_delete_subnets	0	✔
NeutronNetworks.create_and_list_networks	0	✔
NeutronNetworks.create_and_list_ports	0	✔
NeutronNetworks.create_and_list_routers	0	✔

Table A2. *Cont.*

Scenario	Errors	Success (SLA)
NeutronNetworks.create_and_list_subnets	0	✔
NeutronNetworks.set_and_clear_router_gateway	0	✔
NeutronSecurityGroup.create_and_delete_security_group_rule	0	✔
NeutronSecurityGroup.create_and_delete_security_groups	0	✔
NovaKeypair.boot_and_delete_server_with_keypair	0	✔
NovaServerGroups.create_and_delete_server_group	0	✔
NovaServers.boot_server_and_list_interfaces	0	✔
NovaServers.boot_server_associate_and_dissociate_floating_ip	0	✔
NovaServers.boot_server_from_volume_and_delete	0	✔
NovaServers.pause_and_unpause_server	0	✔
Quotas.cinder_update	0	✔
Quotas.cinder_update_and_delete	0	✔
Quotas.neutron_update	0	✔
Quotas.nova_update	0	✔

References

1. Network Functions Virtualisation (NFV). Available online: https://www.etsi.org/technologies/689-network-functions-virtualisation (accessed on 17 September 2022).
2. Hwang, J.; Ramakrishnan, K.K.; Wood, T. NetVM: High performance and flexible networking using virtualization on commodity platforms. *IEEE Trans. Netw. Serv. Manag.* **2015**, *12*, 34–47. [CrossRef]
3. Ferrari, C.; Kovács, B.; Tóth, M.; Horváth, Z.; Reale, A. Edge computing for communication service providers: A review on the architecture, ownership and governing models. In Proceedings of the 2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Hvar, Croatia, 23–25 September 2021; pp. 1–6.
4. Google Cloud, Nokia Partner to Accelerate Cloud-Native 5G Readiness for Communication Service Providers. Available online: https://www.nokia.com/about-us/news/releases/2021/01/14/google-cloud-nokia-partner-to-accelerate-cloud-native-5g-readiness-for-communication-service-providers/ (accessed on 17 September 2022).
5. *ETSI GS NFV*; Network Functions Virtualisation (NFV): Architectural Framework. European Telecommunications Standards Institute: Sophia-Antipolis, France. Available online: https://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf (accessed on 10 October 2022).
6. Bourguiba, M.; Haddadou, K.; Korbi, I.E.; Pujolle, G. Improving network I/O virtualization for cloud computing. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 673–681. [CrossRef]
7. Aziza, H.; Krichen, S. A hybrid genetic algorithm for scientific workflow scheduling in cloud environment. *Neural Comput. Appl.* **2020**, *32*, 15263–15278. [CrossRef]
8. Mohammadzadeh, A.; Masdari, M.; Gharehchopogh, F.S.; Jafarian, A. Improved chaotic binary grey wolf optimization algorithm for workflow scheduling in green cloud computing. *Evol. Intell.* **2021**, *14*, 1997–2025. [CrossRef]
9. Mohammadzadeh, A.; Masdari, M.; Gharehchopogh, F.S.; Jafarian, A. A hybrid multi-objective metaheuristic optimization algorithm for scientific workflow scheduling. *Clust. Comput.* **2021**, *24*, 1479–1503. [CrossRef]
10. Tolia, N.; Wang, Z.; Marwah, M.; Bash, C.; Ranganathan, P.; Zhu, X. Delivering energy proportionality with non energy-proportional systems-optimizing the ensemble. In Proceedings of the 2008 Conference on Power Aware Computing and Systems, San Diego, CA, USA, 7 December 2008; p. 2.
11. Fu, X.; Zhou, C. Virtual machine selection and placement for dynamic consolidation in cloud computing environment. *Front. Comput. Sci.* **2015**, *9*, 322–330. [CrossRef]
12. Gharehpasha, S.; Masdari, M.; Jafarian, A. Power efficient virtual machine placement in cloud data centers with a discrete and chaotic hybrid optimization algorithm. *Clust. Comput.* **2021**, *24*, 1293–1315. [CrossRef]
13. Gharehpasha, S.; Masdari, M.; Jafarian, A. Virtual machine placement in cloud data centers using a hybrid multi-verse optimization algorithm. *Artif. Intell. Rev.* **2021**, *54*, 2221–2257. [CrossRef]
14. Gharehpasha, S.; Masdari, M.; Jafarian, A. The placement of virtual machines under optimal conditions in cloud datacenter. *Inf. Technol. Control.* **2019**, *48*, 545–556. [CrossRef]
15. Carpio, F.; Dhahri, S.; Jukan, A. VNF placement with replication for load balancing in NFV networks. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; pp. 1–6.
16. You, C.; Li, L.M. Efficient load balancing for the VNF deployment with placement constraints. In Proceedings of the ICC 2019—2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.

17. Wang, T.; Xu, H.; Liu, F. Multi-resource load balancing for virtual network functions. In Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, USA, 5–8 June 2017; pp. 1322–1332.
18. Ghorab, A.H.; Kusedghi, A.; Nourian, M.A.; Akbari, A. Joint VNF load balancing and service auto-scaling in NFV with multimedia case study. In Proceedings of the 2020 25th International Computer Conference, Computer Society of Iran (CSICC), Tehran, Iran, 1–2 January 2020; pp. 1–7.
19. Zamani, A.; Bakhshi, B.; Sharifian, S. An efficient load balancing approach for service function chain mapping. *Comput. Electr. Eng.* **2021**, *90*, 106890. [CrossRef]
20. OpenStack Enhanced Platform Awareness. Available online: https://networkbuilders.intel.com/docs/ice-house-openstack-enhanced-platform-awareness.pdf (accessed on 17 September 2022).
21. Tsai, M.H.; Liang, H.T.; Wang, Y.H.; Chung, W.C. Enhanced OpenStack cloud for network function virtualization. In Proceedings of the 2020 International Computer Symposium (ICS), Tainan, Taiwan, 17–19 December 2020; pp. 185–190.
22. Kourtis, M.A.; Xilouris, G.; Riccobene, V.; McGrath, M.J.; Petralia, G.; Koumaras, H.; Gardikis, G.; Liberal, F. Enhancing VNF performance by exploiting SR-IOV and DPDK packet processing acceleration. In Proceedings of the 2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), San Francisco, CA, USA, 18–21 November 2015; pp. 74–78.
23. Bonafiglia, R.; Cerrato, I.; Ciaccia, F.; Nemirovsky, M.; Risso, F. Assessing the performance of virtualization technologies for NFV: A preliminary benchmarking. In Proceedings of the 2015 Fourth European Workshop on Software Defined Networks, Bilbao, Spain, 30 September–2 October 2015; pp. 67–72.
24. Gallenmuller, S.; Emmerich, P.; Wohlfart, F.; Raumer, D.; Carle, G. Comparison of frameworks for high-performance packet IO. In Proceedings of the 2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), Oakland, CA, USA, 7–8 May 2015; pp. 29–38.
25. Kawashima, R.; Nakayama, H.; Hayashi, T.; Matsuo, H. Evaluation of forwarding efficiency in NFV-nodes toward predictable service chain performance. *IEEE Trans. Netw. Serv. Manag.* **2017**, *14*, 920–933. [CrossRef]
26. Halpern, J.; Pignataro, C. (Eds.) *Service Function Chaining (SFC) Architecture. RFC 7665*. Available online: https://www.rfc-editor.org/info/rfc7665 (accessed on 17 September 2022).
27. ETSI GS NFV-TST 001. *Network Functions Virtualisation (NFV); Pre-Deployment Testing*. Report on Validation of NFV Environments and Services. Available online: https://www.etsi.org/deliver/etsi_gs/NFV-TST/001_099/001/01.01.01_60/gs_nfv-tst001v010101p.pdf (accessed on 10 October 2022).
28. Huang, Y.X.; Chou, J. Evaluations of network performance enhancement on cloud-native network function. In Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing, Virtual Event, 21 June 2021; pp. 3–8.
29. Processor Affinity or CPU Pinning. Available online: https://www.intel.com/content/www/us/en/docs/programmable/683013/current/processor-affinity-or-cpu-pinning.html (accessed on 17 September 2022).
30. Wang, X.L.; Luo, T.W.; Hu, J.Y.; Wang, Z.L.; Luo, Y.W. Evaluating the impacts of hugepage on virtual machines. *Sci. China Inf. Sci.* **2017**, *60*, 012103. [CrossRef]
31. Paolino, M.; Nikolaev, N.; Fanguede, J.; Raho, D. SnabbSwitch user space virtual switch benchmark and performance optimization for NFV. In Proceedings of the 2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), San Francisco, CA, USA, 18–21 November 2015; pp. 86–92.
32. Linux Drivers. Available online: https://doc.dpdk.org/guides/linux_gsg/linux_drivers.html (accessed on 17 September 2022).
33. Poll Mode Driver. Available online: https://doc.dpdk.org/guides/prog_guide/poll_mode_drv.html (accessed on 17 September 2022).
34. Yao, J.; Zimmer, V.J.; Zeng, S. A Tour beyond BIOS: Using IOMMU for DMA Protection in UEFI Firmware. Available online: https://www.intel.com/content/dam/develop/external/us/en/documents/intel-whitepaper-using-iommu-for-dma-protection-in-uefi-820238.pdf (accessed on 17 September 2022).
35. Usha Devi, G.; Peduru, K.T.; Mallikarjuna Reddy, B. A novel approach to gain high throughput and low latency through SR-IOV. *Int. J. Eng. Technol.* **2013**, *5*, 1245–1251.
36. Rojas-Cessa, R.; Salehin, K.M.; Egoh, K. Evaluation of switching performance of a virtual software router. In Proceedings of the 2012 35th IEEE Sarnoff Symposium, Newark, NJ, USA, 21–22 May 2012; pp. 1–5.
37. Rizzo, L. Netmap: A novel framework for fast packet I/O. In Proceedings of the 21st USENIX Security Symposium, Bellevue, WA, USA, 8–10 August 2012; pp. 101–112.
38. Shanmugalingam, S.; Ksentini, A.; Bertin, P. DPDK Open vSwitch performance validation with mirroring feature. In Proceedings of the 2016 23rd International Conference on Telecommunications (ICT), Thessaloniki, Greece, 16–18 May 2016; pp. 1–6.
39. Xu, C.; Wang, H.; Shea, R.; Wang, F.; Liu, J. On multiple virtual NICs in cloud computing: Performance bottleneck and enhancement. *IEEE Syst. J.* **2018**, *12*, 2417–2427. [CrossRef]
40. Pitaev, N.; Falkner, M.; Leivadeas, A.; Lambadaris, I. Characterizing the performance of concurrent virtualized network functions with OVS-DPDK, FD.IO VPP and SR-IOV. In Proceedings of the ACM/SPEC International Conference on Performance Engineering, Berlin, Germany, 9–13 April 2018; pp. 285–292.
41. Ara, G.; Cucinotta, T.; Abeni, L.; Vitucci, C. Comparative evaluation of kernel bypass mechanisms for high-performance inter-container communications. In Proceedings of the 10th International Conference on Cloud Computing and Services Science (CLOSER), Prague, Czech Republic, 7–9 May 2020; pp. 44–55.

42. Wang, G.; Ng, T.S.E. The impact of virtualization on network performance of amazon ec2 data center. In Proceedings of the 2010 Proceedings IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010; pp. 1–9.
43. Callegati, F.; Cerroni, W.; Contoli, C. Virtual networking performance in OpenStack platform for network function virtualization. *J. Electr. Comput. Eng.* **2016**, *2016*, 15. [CrossRef]
44. OPNFV Functest. Available online: https://functest.readthedocs.io/en/latest/testing/user/configguide/configguide.html#functest-dockers-for-openstack-deployment (accessed on 17 September 2022).
45. NFVbench: A Network Performance Benchmarking Tool for NFVi Full Stacks. Available online: https://docs.opnfv.org/projects/nfvbench/en/latest/testing/user/userguide/readme.html (accessed on 17 September 2022).
46. Prerequisites for Installing a Cloudify Manager. Available online: https://docs.cloudify.co/4.4.0/install_maintain/installation/prerequisites/ (accessed on 17 September 2022).
47. Sengupta, S.; Yadav, V.K.; Saraf, Y.; Gupta, H.; Ganguly, N.; Chakraborty, S.; De, P. MoViDiff: Enabling Service Differentiation for Mobile Video Apps. In Proceedings of the 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal, 8–12 May 2017; pp. 537–543.