A Calculation Method of vCPU Occupancy Rate of Virtual Machine Forwarding Process

1st Hua Lu[1], 2nd Qinshu Chen[1], 3rd Jidong Zhang[1*], 4th Xuefei Duan[1]
{lh_zqh@aliyun.com[1], chenqinshu@gdcni.cn[1], zhangjidong@gdcni.cn[1],
duanxuefei@gdcni.cn[1] }

Guangdong Communications & Networks Institute[1]

Abstract. Network Function Virtualization (NFV) is a technology that implements network functions through virtualization on x86 servers. In order to improve the forwarding performance of a VM(virtual machine), usually the most effective method is to use multiple cores and exclusive methods of the vCPUs in the VM to monopolize the physical CPU resources for multiple forwarding processes to use independently. The forwarding process discards the interrupt-based asynchronous signal sending mechanism to avoid the impact of interrupt switching on the forwarding overhead. Instead, it uses a while 1 dead-loop to poll the packet receiving queue. Once there is a packet in the packet receiving queue, the packet is immediately forwarded. Because the vCPU bound to the forwarding process works in an infinite loop polling mode, the vCPU occupancy display is 100% regardless of whether it is in the no-load or full-load phase. Because the VM cannot obtain the real load of each forwarding vCPU in real time and expand it in time, it will cause VM to lose a lot of packets due to overload operation, which will affect the quality of the service carried by the VM. This article studies how to measure the real utilization of vCPU in real time. The results show that the real vCPU occupancy can be accurately calculated using this solution.

Keywords: DPDK, SR-IOV, OVS-DPDK, NFV, vCPU, VM, Occupancy Rate, Forwarding Process.

1 Introduction

Network Functions Virtualisation aims to transform the way that network operators architect networks by evolving standard IT virtualisation technology to consolidate many network equipment types onto industry standard high volume servers, switches and storage, which could be located in Datacentres, Network Nodes and in the end user premises. It involves the implementation of network functions in software that can run on a range of industry standard server hardware, and that can be moved to, or instantiated in, various locations in the network as required, without the need for installation of new equipment [1].

With the continuous advancement of operator network reconstruction, innovative technology architectures represented by SDN/NFV are accelerating the network towards full cloudification, and the transformation of the network into a data centered cloud-based network has become the focus of industry attention, such as foreign AT&T Domain 2.0, China Telecom's CTNet2025, China Mobile's Novonet, and China Unicom's CO reconstruction. At present, the general consensus reached in the industry is that 5G networks will be based on SDN/NFV technology and cloud computing technology to implement network virtualization

EAI MOBIMEDIA

DOI 10.4108/eai.27-8-2020.2294712

and cloud deployment. The "China Telecom 5G technology White Paper" released by China Telecom puts forward the logic architecture of 5G target network, referred to as "three clouds" network architecture. The "three clouds" architecture meets the characteristics of flexible, intelligent, integrated and open 5G network in the future, and is also built on the technology foundation of SDN/NFV and network cloud[2]. With the maturity of virtualization technology, considering the issues of bandwidth and latency, the concept of edge computing has begun to be introduced [3]. With edge computing, a large number of computing tasks can be processed directly near the source of the data generation, which greatly eases the transmission pressure on the network. At the same time, tasks are processed at the edge, which reduces the delay caused by data transmission speed and bandwidth limitations, and users get faster response time. Edge computing is generally a micro data center composed of several servers. Under the condition of limited resources, the dynamic elastic scalability of NFV virtualization technology is used to achieve the efficient use of edge computing resources.

The problem of insufficient software forwarding performance in virtualization has been greatly improved after the rise of DPDK technology. In this article, we start with the analysis of virtualization technology and introduce several typical software forwarding models. In order to improve the software forwarding performance, the DPDK needs to bind the core to exclusive vCPU resources for the while loop packet processing. The vCPU utilization counted by the general operating system = (1-percentage of time occupied by the idle process). Since the vCPU bound by the DPDK runs in an endless loop, the vCPU will not enter the idle process, so the vCPU utilization is always 100%.

This brings up a question: What is the actual vCPU utilization rate, and does it need to be expanded? Virtualization has the ability to flexibly scale in/out, so how to accurately calculate the actual vCPU occupancy rate at the current moment makes it possible to implement automatic elastic scaling of virtual network elements based on a preset threshold of vCPU occupancy. In this paper, the method of real-time quantitative measurement of vCPU's real occupancy is researched and described, in order to achieve real-time accurate display of the vCPU under the situation of dead loop running vCPU real vCPU occupancy.

Chapter 2 of this article first introduced virtualization-related technologies, and described several typical general server-based forwarding models. Finally, in Chapter 3, the vCPU load indeterminacy when the VM implements soft forwarding is studied in depth.

2 Virtual machine forwarding

2.1 Virtualization technology

Virtualization currently uses Intel's x86 architecture. Intel's virtualization technology (VT) mainly includes VT-x technology for CPU processors, VT-d technology for I/O chipset and VT-c technology for network interface cards. These are hardware-assisted virtualization technologies that provide strong underlying technical support for the widespread development of NFV.

Virtualized network elements in the CT industry generally use the open source Linux operating system, which uses "network interface card reception-> network interface card interruption-> kernel reception-> send messages to user processes-> switch to user mode-> user process processing packet" packet processing mode, which involves multiple memory copies, kernel mode/user mode switching, and process scheduling, which is very inefficient.

What is more serious is that the VM runs on the Host machine, which means that the VM user application layer receives a packet and needs to pass through the kernel processing of the Host OS kernel and the VM OS kernel twice. For virtualized network elements with large forwarding traffic, the performance loss is unacceptable. The DPDK (Data Plane Development Kit) technology was born [6]. DPDK runs in user mode and uses the PMD (Poll Mode Driver). PMD consists of APIs provided by specific drivers in user space for setting up devices and their corresponding queues. Abandoning the interrupt-based asynchronous signaling mechanism brings great cost savings to the architecture. Avoiding interruption performance bottlenecks is one of the keys for DPDK to improve packet processing speed. At the same time combined with zero-copy, memory huge pages, NUMA, SR-IOV and other technologies, greatly improved software forwarding capabilities [4][8].

2.2 Virtualized forwarding model

In the pure server case, software forwarding mainly has the following three forms:

1) VM forwarding based on OVS-DPDK

OVS-DPDK [7] runs in Host OS user mode, DPDK on a VM runs in Guest OS user mode, and VF is a virtual network interface card assigned to the VM. As shown in Fig 1: Packet interaction for two VMs in the same Host, it is forwarded through the OVS-DPDK of the Host, see the curve marked by the label a in the figure; For packet interaction between two VMs across the Host, the OVS-DPDK bypass kernel receives packets directly from the physical NIC. OVS-DPDK sends packets to the corresponding VM through table matching, and the VM user mode DPDK bypasses the VM OS kernel to directly receive packets. See curve marked with label b in the figure. This forwarding model is very flexible, and because the Host and Guest kernel processing is bypassed, forwarding performance is also guaranteed to some extent, but because OVS-DPDK also needs to occupy a certain amount of CPU resources, it will affect the VM's resource occupation allocation to a certain extent.. At the same time, OVS-DPDK serves as a unified output for all VMs on the Host, and the forwarding performance of all VMs cannot exceed the forwarding performance of OVS-DPDK, which is a bottleneck point in forwarding performance.

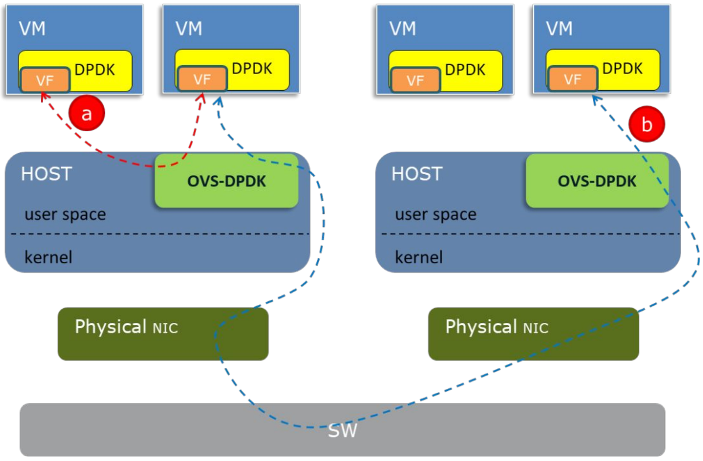

Fig. 1. VM forwarding model based on OVS-DPDK

2) Virtual forwarding based on SR-IOV

The VM uses SR-IOV technology to directly take packets from the physical network interface card for processing, completely bypassing the kernel mode and user mode processing on the Host, and its performance is almost equivalent to the processing of receiving and sending packets on the Host. A physical network interface card supporting VT-c technology can virtualize multiple vfs. Each vf corresponds to an independent packet sending and receiving memory space on the physical network interface card. These vf are allocated to the corresponding VMs respectively. One correspondence makes it possible for the VM to read and write packets directly from the physical NIC memory. This SR-IOV method, combined with DPDK, completely bypasses the processing of the forwarded packets by the Host OS and Guest OS kernels, and the software forwarding performance is greatly improved [5]. The VM forwarding model of SR-IOV+DPDK is also the current choice to pursue the ultimate pure software forwarding. As shown in Fig2: Packet interaction for two VMs in the same Host, skip the Host OS and forward it directly by the switching chip on the physical network interface card, see the curve marked by the label a in the figure; For packet interaction between two VMs across the Host, the VM directly writes the packet to the physical NIC memory and sends it to the external switch, and the other VM reads the packet directly from the physical NIC memory of the Host, and combines the packet processing with the user mode DPDK, see the curve marked with label b in the figure. This forwarding model is the optimal model of VM pure software forwarding, but it is less flexible than the above model 1 due to the requirement of hardware characteristics support for the physical network interface card.

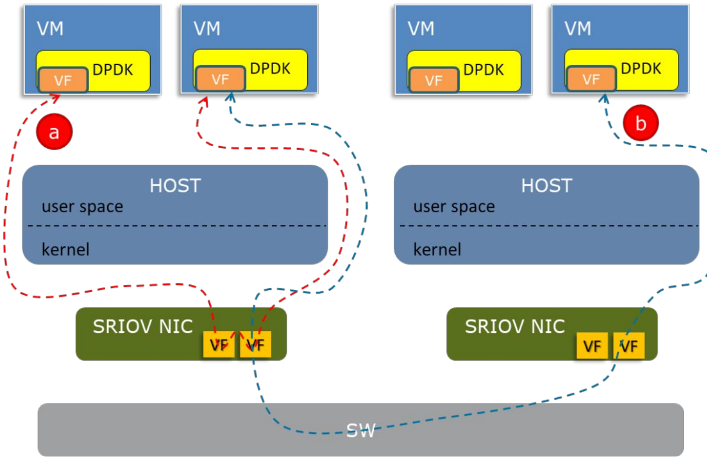

Fig. 2. SR-IOV-based VM forwarding model

3) VM forwarding based on Intelligent Ethernet Card

This model is derived from model 1, which sinks ovs-dpdk into the Intelligent Ethernet Card to release the occupation of OVS-DPDK on the Host CPU resources in model 1. The powerful throughput capacity of the special forwarding chip in the Intelligent Ethernet Card will not become the forwarding performance bottleneck of the VM belonging to the Host. In addition to OVS-DPDK, some services in the VM can also be sunk, such as tunnel encapsulation and decapsulation, fragmentation and reassembly, load balancing, and so on.

According to the actual service deployed by the VM, the corresponding service sink processing is performed to valuable CPU resources are released for use by VM. As shown in Fig 3: Packet interaction for two VMs in the same Host, skip the Host OS and forward it directly by the switching chip on the Intelligent Ethernet Card, see the curve marked by the label a in the figure; For packet interaction between two VMs across the Host, the VM directly writes the packet to the Intelligent Ethernet Card memory and sends it to the external switch. The other VM reads the packet directly from the Host's physical network interface card memory and combines the packet processing with the user mode DPDK, see the curve marked by label b in the figure; Some services functions sink the Intelligent Ethernet Card, for this type of forwarded packet, it can be forwarded directly after being processed by the Intelligent Ethernet Card without being sent to the CPU for processing, which can save valuable CPU resources, see the curve marked by label c in the figure.

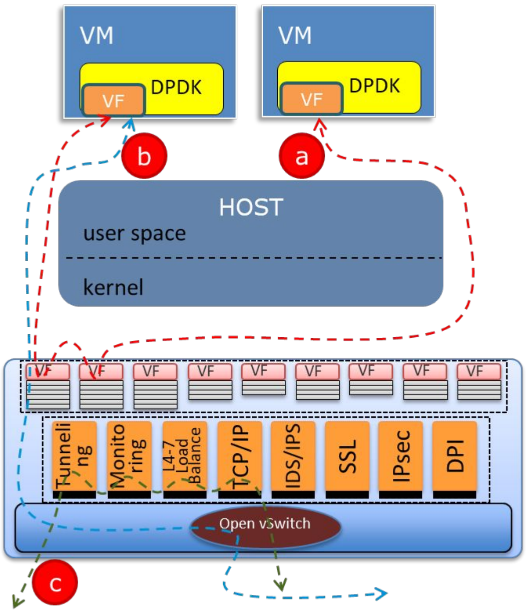

Fig. 3. VM forwarding model based on Intelligent Ethernet Card

Among the above three virtual forwarding models, the VM dpdk and Host ovs-dpdk in model 1 and the VM dpdk in models 2 and 3 will all involve vCPU binding operations to improve software forwarding capabilities.

2.3 Soft forward CPU load uncertainty

Network Function Virtualization is a technology that implements network functions through virtualization on x86 servers. In order to improve the forwarding performance of a VM, usually the most effective method is to use multiple cores and exclusive methods of the vCPUs in the VM to monopolize the physical CPU resources for multiple forwarding processes to use independently. The forwarding process discards the interrupt-based asynchronous signal sending mechanism to avoid the impact of interrupt switching on the forwarding overhead. Instead, it uses a while 1 dead-loop to poll the packet receiving queue. Once there is a packet in the packet receiving queue, the packet is immediately forwarded. As shown in Fig.4, in the VM for network function virtualization, there are multiple vCPUs, and the forwarding process is bound to these vCPUs in an exclusive way of binding cores, and an endless loop polling method is used for efficient forwarding service processing.

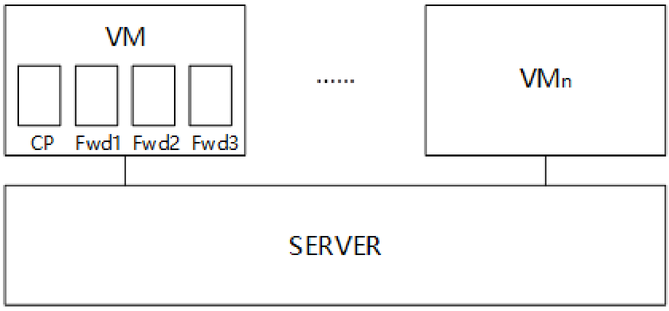

Fig. 4. Multi-vCPU VM

Because the vCPU bound to the forwarding process works in an infinite loop polling mode, the vCPU occupancy display is 100% regardless of whether it is in the no-load or full-load phase.

When a network-capable VM is deployed, it estimates the processing throughput required by the VM based on information such as the current business scenario and the number of users, and deploys redundant capabilities. As business scenarios change and the number of users increases, the redundant processing capabilities of the originally deployed VMs will become less and less redundant, or even exceed the original processing VM's forwarding processing capabilities, causing packet processing to be discarded in a timely manner.

Currently, the maximum throughput of VMs with network functions in typical scenarios is generally provided. In the daily operation and maintenance process, various statistical (VM throughput, packet loss, and other statistical information) of VMs that rely on human resources to detect network functions are used. Plan for subsequent expansion. This method relying on human guarantees is inefficient, and is subject to human factors, which has greater risks. Also, different types of x86 CPUs have different processing capabilities, which increases the difference in the maximum throughput value of the VM originally given.

3 Technical solutions and implementation

In order to improve the operation and maintenance efficiency in the virtualization scenario and reduce the risks caused by differences in physical equipment and human factors, this paper proposes a calculation method for the real CPU occupation rate of the vCPU bound by the endless loop operation of the VM forwarding process of network functions , Which can conveniently and intuitively display the real CPU usage of each vCPU of the VM to which the network function virtualization belongs, and make a VM capacity expansion plan in advance. The technical scheme is as follows:

First, we need to set the weight value according to the different packets that the VM needs to process. The packet weight value is obtained based on the CPU resources consumed by one packet processing. The size of the weight value is determined by the implementation complexity of different services, so the weight value remains unchanged after the service function goes online.

Secondly, it is necessary to obtain the crystal clock of the physical CPU of the general server as the reference clock, and based on the reference clock, simulate the processing of forwarded packets to obtain the processing time of the baseline service (IPV4) packet, according to the reference clock and the processing time of the baseline service packet, get the total number of packets that can be processed in a vCPU usage calculation cycle.

Finally, based on the statistical values of different service packets processed by the VM in a calculation cycle, and combining different weight values of different service packets, the number of limit service packets can be calculated. According to the formula:

$$CPU\,usage = \frac{number\ of\ baseline\ service\ packets}{total\,number\ of\ packets} \times 100\,\%$$

We can get the real vCPU usage in the current calculation cycle.

It should be noted that hardware resources such as the CPU computing unit and CACHE are shared resources. Due to the different load of the bearer service in different time periods, the use of shared hardware resources is different. This has an impact on the accuracy of the originally calculated baseline service packet processing time.. To reduce the impact of shared hardware resource usage on the accuracy of baseline service packet processing time, the baseline service packet processing time needs to be updated regularly. Therefore, it is necessary to set a system timer, periodically simulate the processing of forwarded packets, and periodically obtain the processing time of the baseline service (IPV4) packets to use more accurate values in the calculation of the next cycle.

In order to facilitate the understanding of this technical solution, this section describes the specific implementation of the technical solution in combination with specific service functions. The definition symbols used in the text are explained here:

Table 1. Symbol Description

Symbol	Description
Fwd_i	Identifier of the forwarding process i, used to identify multiple forwarding processes. The value of i is greater than 0 and less than the number of vCPUs occupied by the virtual machine (VM)
T_{cpu}	Crystal clock of server physical CPU
$T_{std,i}$	Baseline packet processing time. Here, the IPV4 packet is used as the baseline packet to obtain the processing time. i is the number of forwarding processes
T_{load}	One vCPU usage calculation cycle
T_{cyc}	Running timer period
$Num_{total,i}$	The total number of baseline packets that can be processed in a vCPU usage calculation period. i is the number of forwarding processes
β_{v4}	IPV4 service weight value. Since IPV4 is a baseline service, the weight value is 1

β_{ipsec}	Weight of the ipsec service compared to the IPV4 baseline packet processing time
β_{nat}	Weight of the NAT service compared to the IPV4 baseline packet processing time
$Counter_{v4,i}$	Baseline IPV4 packet statistics of the forwarding process Fwd_i during a vCPU usage calculation period. i is the number of forwarding processes
$Counter_{ipsec,i}$	Statistics of ipsec service packets of the forwarding process Fwd_i during a vCPU occupancy calculation period. i is the number of forwarding processes
$Counter_{nat,i}$	Statistics of nat service packets of the forwarding process Fwd_i during a vCPU occupancy calculation period. i is the number of forwarding processes
$Counter_{std,i}$	Number of packets in the forwarding process Fwd_i after a baseline in a vCPU occupancy calculation period obtained after weighting various service packets. i is the number of forwarding processes
$Load_i$	vCPU percentage occupancy, the maximum i is the number of vCPUs occupied by the VM

The calculation method for the real occupancy rate of vCPUs in an infinite loop includes the acquisition of the physical CPU crystal clock, the acquisition of baseline packet processing time, the acquisition of the total number of baseline packets that can be processed by the vCPU occupancy calculation cycle, the acquisition of the weight relationship between various services and the baseline packets, obtain statistics of various service packets processed during the vCPU occupancy calculation cycle. As shown in Fig 5:

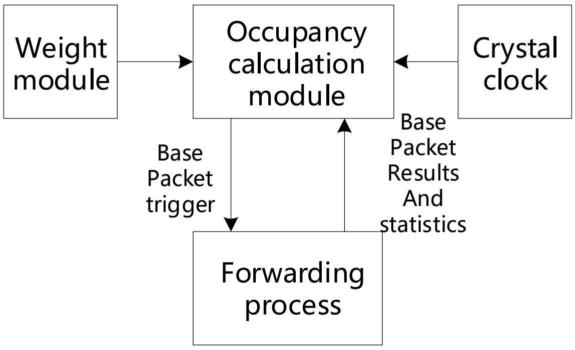

Fig. 5. Calculation of initial state occupancy

The specific implementation steps are as follows:

After the VM is powered on, the occupancy calculation module obtains the crystal clock T_{cpu} of the physical CPU of the general server where the VM is located as the reference clock of the forwarding process Fwd_i.

The occupancy calculation module uses T_{cpu} as the reference clock and records the start time T1. At the same time, it instructs the simulator to trigger the IPV4 baseline packet to the forwarding process Fwd_i for processing. After the processing of the forwarding process is complete, record the time T2 to obtain the baseline packet processing time $T_{std,i}$ = T2-T1.

Divide a vCPU occupancy calculation period by dividing the processing time of a single baseline packet to obtain the number of baseline packets that can be processed during the calculation period, that is, $Num_{total,i}=T_{load}/T_{std,i}$.

The weight presetting module presets a weight ratio relationship between processing of each service packet and processing time of the baseline packet.

The forwarding module counts the number of packets processed by each service.

The occupancy calculation module obtains the weight value of each service packet and the number of processing each service packet in each calculation cycle. According to the formula $\text{Counter}_{std,i} = \sum Counter(k) * ¿ \beta(k) ¿$ (where k is Different services, such as: ipsec, nat, etc.), get the number of baselined packets.

Divide the number of baselined packets $\text{Counter}_{std,i}$ by the total number of baseline packets $\text{Num}_{total,i}$ that a vCPU occupancy can process in a cycle to get the true vCPU occupancy, ie: $\text{Load}_i = \text{Counter}_{std,i} / \text{Num}_{total,i} * 100\%$.

Considering that in a virtualized environment, the server's physical CPU, CACHE, and other hardware resources are shared and used, the processing time of baseline packets may change under different traffic processing situations, so the timer device needs to be enabled during the running state Calculate the baseline packet processing time periodically to obtain a more accurate total number of baseline packets that can be processed in the vCPU usage calculation period. As shown in Fig 6:

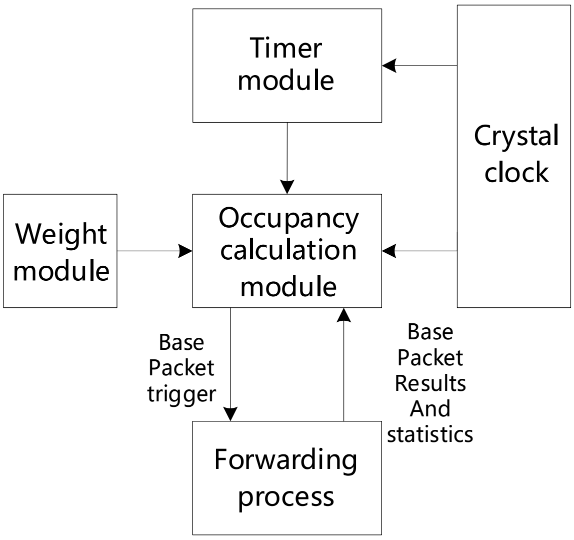

Fig. 6. Calculation of operating occupancy

In the running state, the crystal clock T_{cpu} of the server's physical CPU is used as the reference clock, the system timer is started, and the timer period is set to T_{cyc}. The timer expires, triggers a message, and dynamically adjusts the processing time of the baseline packet. By periodically adjusting the baseline value, it can reflect more real processing power according to the usage of physical resources, making the calculated vCPU usage more accurate.

Through the above measurement method, a more accurate vCPU occupancy situation can be obtained in real time. Compared with the 100% displayed by the operating system vCPU monitoring tool, the vCPU occupancy rate described in this article can dynamically and accurately display the real-time vCPU occupancy situation. Fig 7 is a graph of the vCPU occupancy curve when the vCPU is running different services with the same throughput, using the operating system's own monitoring tools and using this solution's measurement method.

The general server configuration used in this article is as follows: Intel XEN Gold 5118 processors (12 cores each, 2.3GHz) * 2, 128G DDR4 Memory and an Intel X520 10GbE PCIe dual port network interface card. The software uses DPDK 18.11 version, combined with software code to achieve service functions of IPV4, NAT and IPSEC. Use Spirent TestCenter tester to directly connect with server's X520 network interface card to send and receive packets. The figure shows the vCPU usage of different services (IPV4, NAT and IPSEC) at the same throughput.

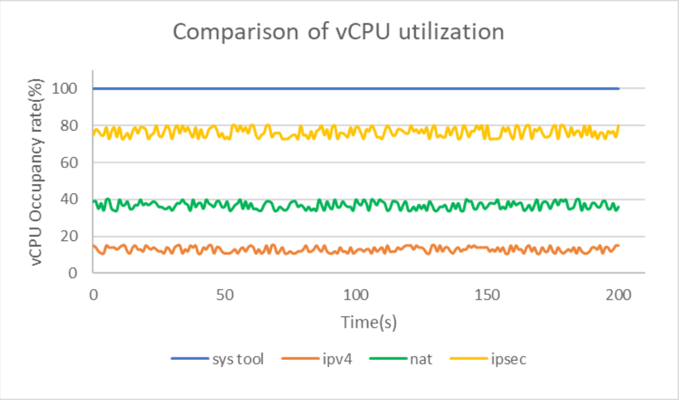

Fig. 7. Comparison of vCPU utilization

4 Conclusion

This article proposes a method for calculating the real vCPU occupancy when the 100% occupancy of the VM's forwarding process is displayed, and the baseline service packet processing time is obtained by timing the simulated packet processing. According to different service statistics and corresponding the weight value normalizes the service flow model during the calculation period of the occupancy rate, so as to obtain the real vCPU usage. In addition, considering the impact of hardware resource sharing usage, a method of periodically updating the processing time value of the baseline service packet is adopted to achieve the purpose of truly reflecting the vCPU usage. Combining alarms, statistics, and graphical displays allows operations and maintenance personnel to easily and intuitively detect changes in forwarding throughput and schedule capacity expansion in a timely manner.

Acknowledgement. This work was supported by the National Key Research and Development Program of China under Grant (2019YFB1804400).

References

[1] Network Fuctions Virtualisation–Introductory White Paper. https://portal.etsi.org/nfv/nfv_white_paper.pdf
[2] China Telecom 5G Technology White Paper

[3] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu, "Edge Computing: Vision and Challenges," IEEE INTERNET OF THINGS JOURNAL, VOL. 3, NO. 5, OCTOBER 2016.

[4] DPDK White Paper.

[5] Michail Alexandros Kourtis, Georgios Xilouris, Vincenzo Riccobene, "Enhancing VNF performance by exploiting SR-IOV and DPDK packet processing acceleration." Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on IEEE, 2015.

[6] DPDK.org. DPDK: Data Plane Development Kit. Available: http://dpdk.org/

[7] OVS-DPDK. http://docs.openvswitch.org/en/latest/intro/install/dpdk/

[8] Niu, Zhixiong , et al. "Benchmarking NFV Software Dataplanes." (2016).