

Tenant Based Dynamic Processor Mapping in the Cloud Network Function Virtualization System

Kang Il Choi, Bhum Cheol Lee

Network Computing Convergence Lab., ETRI, Daejeon, Korea

forerunner@etri.re.kr, bclee@etri.re.kr

Abstract— In this paper, we present a novel processor mapping method for Cloud Network Function Virtualization System which ensures that network traffic processing of virtual machines belonging to the same tenant is not affected by congestion in network traffic of virtual machines belonging to different tenant. In this paper, we provide a method of dynamically mapping a processor, the method including extracting tenant information on a tenant and information on a virtual machine generated by a Cloud OS or controller; classifying virtual machine queues and processors to process the virtual machine queues by tenant; and dynamically mapping the virtual machine queues onto the processors by tenant. The dynamically mapping of the virtual machine queues onto the processors by tenant may include dynamically mapping the processors to process the VMQs in proportion to either a total number of virtual machine queues belonging to the same tenant, or a total number of virtual machine queues belonging to the same tenant. Finally, we describe the operation of the Tenant based Dynamic Processor Mapping including both spreading process and coalescing process with flow chart.

Keywords— Dynamic Processor Mapping, Cloud Network Function Virtualization System

I. INTRODUCTION

A processing technique for speedup of network traffic in a network interface card (NIC) of a multiprocessor-based server system is constantly advancing. One of the most serious problems is that traffic received by the NIC is not effectively processed by a multiprocessor in a server.

A technology for resolving an RSS (Receive Side Scaling) problem in a virtualized environment is Virtual Machine Queue (VMQ). VMQ technology allows distributed processing of network traffic by a multiprocessor even in a virtualized environment using a method of generating a VMQ in an NIC and mapping each VMQ onto a processor in a host, making high-speed traffic processing possible[1,2].

FIG. 1 illustrates a configuration of conventional VMQ technology. A server system sets a VMQ and assigns a processor to process the VMQ. The server system receives incoming packets input to a server through an NIC, classifies the packets based on L2 information, for example, destination media access control (MAC), and sends the packets to the VMQ. The processor to process the VMQ processes the packets sent to the VMQ and transmits the packets to a virtual

machine through an L2 virtual switch, and the virtual machine processes the received packets.

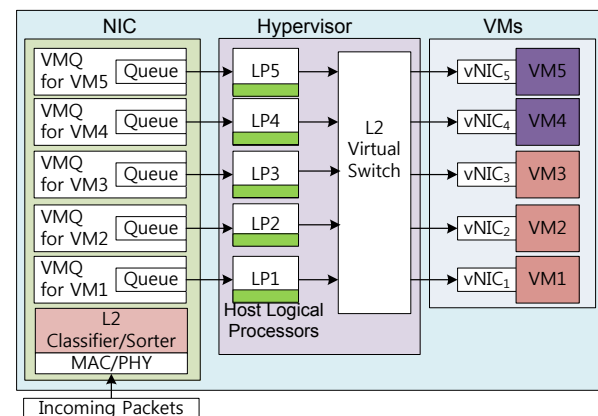


Figure 1 Virtual Machine Queue technology

Since VMQ technology statically assigns VMQs, a too small number of processors may be assigned to a plurality of queues which involve traffic congestion or a small number of queues which hardly use traffic may be assigned to a too large number of processors [2].

To solve a statistical assignment problem of VMQ, Dynamic Virtual Machine Queue (DVMQ) is provided. DVMQ technology is a method of dynamically assigning a VMQ to a multiprocessor based on network traffic amount and total use of a processor [3].

When network traffic amount or total use of a processor exceeds a set upper threshold, a multi-machine queue assigned to the processor needs dynamically reassigning. When the network traffic amount or total use of the processor is below a set lower threshold, the multi-machine queue assigned to the processor is dynamically reassigned to remove the processor from assigning, thereby optimizing network traffic processing performance and power consumption.

Meanwhile, network virtualization in a cloud system means ensuring that network traffic of virtual machines belonging to the same tenant is not affected by network traffic of virtual machines belonging to different tenants. However, conventional VMQ or DVMQ technology provides a technique for optimizing network processing performance and power consumption in a multiprocessor environment but does not consider a network virtualization technique required by the cloud system in the optimization process.

Thus, a method of providing network virtualization is needed which ensures that network traffic processing of virtual machines belonging to the same tenant is not affected by congestion in network traffic of virtual machines belonging to different tenants.

The paper is structured as follows. Section II describes an overview of the Tenant Based Dynamic Processor Mapping (TBDPM). In Section III, we describe the operation of the Tenant Based Dynamic Processor Mapping method with flowcharts so that we can provide novel network virtualization which guarantee tenant based QOS requirement.

II. TENANT BASED DYNAMIC PROCESSOR MAPPING

FIG. 2 is a structural diagram illustrating a Tenant Based Dynamic Processor Mapping (TBDPM) in the Cloud Network Function Virtualization System [4].

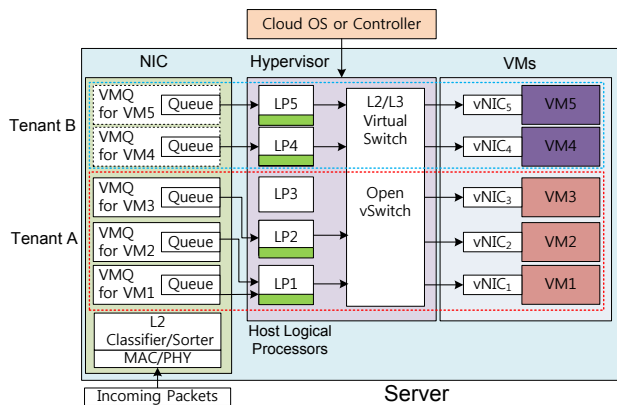


Figure 2 A Structural diagram of the TBDPM

FIG. 2 illustrates the structural diagram of dynamically mapping virtual machine queues (VMQs) onto processors to process the VMQs in proportion to total numbers of virtual machines (VMs) belonging to the same tenants.

A cloud operating system (Cloud OS) or controller may generate tenant information and VM information and extract tenant information and VM information from a hypervisor or VM manager.

VMQs and processors to process the multiple queues are classified by tenant, and a VMQ belonging to each tenant may be dynamically mapped onto a multiprocessor belonging to the tenant.

Here, the VMQs and the processors to process the VMQs may dynamically be mapped onto each other in proportion to total numbers of VMs belonging to the same tenants.

For example, when VMs VM1, VM2 and VM3 belong to tenant A, VMs VM 4 and VM5 belong to tenant B and five processors to process VMQs are provided, a total of three VMs are present for tenant A and a total of two VMs are present for tenant B in total, and thus the VMQs and the processors to process the VMQs may dynamically be mapped at a ratio of tenant A to tenant B of 3:2. That is, among the processors to process a total of five VMQs, three processors may be mapped onto tenant A and two processors may be

mapped onto tenant B in proportion to the total numbers of VMs belonging to the tenants.

In the foregoing situation, LP1, LP2 and LP3 may be mapped onto tenant A, wherein a VMQ for VM1 and a VMQ for VM2 of tenant A may be mapped onto LP1 and a VMQ for VM3 may be mapped onto LP2. No VMQ may be mapped onto LP3. Also, LP4 and LP5 may be mapped onto tenant B, wherein a VMQ for VM4 of tenant B may be mapped onto processor LP4 and a VMQ for VM5 may be mapped onto LP5.

According to the FIG. 2, the server system may include a network interface card (NIC), a hypervisor, and a plurality of VMs.

First, the NIC may include a media access control (MAC)/physical layer (PHY) processor, a layer 2 (L2) classifier/sorter, and VMQs. The hypervisor may include processors to process the VMQs and virtual switches. The VMs may be generated by the Cloud OS through the hypervisor and have virtual network interfaces (vNICs).

The server system sets a VMQ by tenant generated by the Cloud OS or controller and assigns a processor to process the VMQ. The server system receives incoming packets input to the server through the NIC, classifies the incoming packets based on L2 information, for example, destination MAC, and sends the packets to the VMQ.

The processor to process the VMQ processes the incoming packets and transmits the packets to a VM through an L2 virtual switch. The VM may process the received packets.

Table 1 illustrates a VMQ/LP mapping table which dynamically maps VMQs onto processors to process the VMQs in proportion to total numbers of VMs belonging to the same tenants.

TABLE 1. VMQ/LP MAPPING TABLE 1

Tenant	Virtual Machine Queue	Logical Processor
Tenant A	VMQ1	LP1
	VMQ2	LP1
	VMQ3	LP2
	NONE	LP3
Tenant B	VMQ4	LP4
	VMQ5	LP5

We may also dynamically map VMQs and processors to process the VMQs onto each other in proportion to total numbers of virtual CPUs (vCPUs) used for VMs belonging to the same tenants.

III. THE OPERATION OF THE TENANT BASED DYNAMIC PROCESSOR MAPPING

FIG. 3 illustrates a flowchart illustrating a tenant-based dynamic processor mapping method.

The Cloud OS or controller may generate a tenant. To generate VMs for the tenant, the Cloud OS may transmit a command to the hypervisor, and the hypervisor receiving the command may generate VMs.

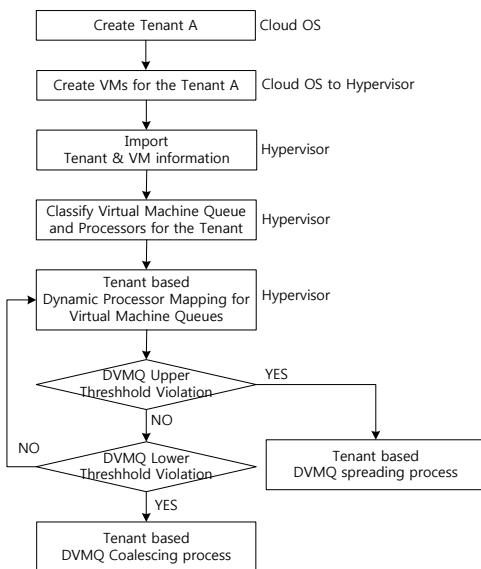


Figure 3 A Flowchart of TBDPM

Tenant information on the tenant generated by the Cloud OS or controller and information on the VMs may be extracted. VMQs and processors to process the VMQs may be classified for each generated tenant. The VMQs may dynamically be mapped onto the processors for each tenant.

When the hypervisor dynamically maps the VMQs, VMQs belonging to the same tenant may be set to be mapped onto the same processor. Alternatively, the VMQs may be mapped by a conventional VMQ mapping method.

Also, the VMQs and the processors to process the VMQs may dynamically be mapped onto each other in proportion to total numbers of VMs belonging to the same tenants.

Alternatively, the VMQs and the processors to process the VMQs may dynamically be mapped onto each other in proportion to total numbers of virtual CPUs (vCPUs) used for VMs belonging to the same tenants.

A method of dynamically mapping VMQs and processors in each tenant may conform to a conventional DVMQ manner. To support network virtualization by removing interference between different tenants, when VMQ/LP mapping is dynamically changed, VMQs belonging to the same tenant may be mapped restrictively onto a processor (LP) belonging to the tenant.

A. Tenant Based Dynamic Process Mapping Spread Process

FIG. 4 is a flowchart illustrating a TBDPM spreading process.

Referring to FIG. 4, due to an increase in traffic transmitted to VMs belonging to a tenant T_i , a processor p to process a corresponding VMQ may exceed a TBDPM upper threshold.

It may be verified whether TBDPM spreading is possible using all processors to process VMQs belonging to the tenant T_i .

When TBDPM spreading is possible using all the processors to process the VMQs belonging to the tenant T_i , a

TBDPM spreading process may be performed using all the processors to process the VMQs belonging to the tenant T_i and then the TBDPM spreading process may be finished.

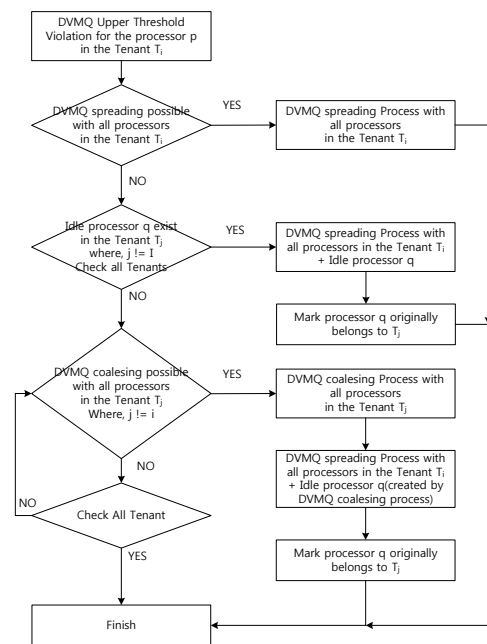


Figure 4 Spreading Process of TBDPM

When the TBDPM spreading process is impossible using all the processors to process the VMQs belonging to the tenant T_i , it may be verified whether an idle processor q is available for all other tenants in the server.

As a result of verifying whether the idle processors q is available for the other tenants, when the idle processor q to process a VMQ belonging to a tenant T_j is available, the TBDPM spreading process may be performed using all the processors to process the VMQs belonging to the tenant T_i and the idle processor q to process the VMQ belonging to the tenant T_j .

The idle processor q to process the VMQ belonging to the tenant T_j may be marked as originally belonging to the tenant T_j and the TBDPM spreading process may be finished.

When the TBDPM spreading process is impossible using all the processors to process the VMQs belonging to the tenant T_i and the idle processor q is unavailable for all the other tenants in the server, it may be verified whether the TBDPM coalescing process is possible for all tenants in the server.

When the TBDPM coalescing process is possible for the tenant T_j , the TBDPM coalescing process may be performed using all processors to process VMQs belonging to tenant T_j .

The TBDPM spreading process may be performed using the processor q of the tenant T_j , which becomes an idle processor as a result of performing the TBDPM coalescing process, and all the processors to process the VMQs belonging to the tenant T_i .

The idle processor q to process the VMQ belonging to the tenant T_j may be marked as originally belonging to the tenant T_j and the TBDPM spreading process may be finished.

As a result of verification, when the TBDPM coalescing process is impossible for all the tenants T_j in the server, it may be checked whether verification is performed for all the tenants and the TBDPM spreading process may be finished.

B. Tenant Based Dynamic Process Mapping Coalescing Process

On the contrary, for minimization of power consumption, when traffic use of VMs belonging to one tenant decreases so that a reduction in the number of processors belonging to the same tenant is possible, VMQs and processors to process the VMQs may dynamically mapped onto each other so that another tenant may use the processors. We call it “TBDPM coalescing process”.

Due to a decrease in traffic transmitted to VMs belonging to a tenant T_i , a processor p to process a corresponding VMQ may not reach a TBDPM lower threshold.

It may be verified whether the processor p is a processor which originally processes a VMQ belonging to a tenant T_i .

When it is determined that the processor p is not the processor which originally processes the VMQ belonging to the tenant T_i , the TBDPM coalescing process may be performed on the tenant T_i using all processors to process VMQs belonging to the tenant T_i .

The processor p may be mapped onto the tenant T_j to which the processor p originally belongs, and the TBDPM coalescing process may be finished.

When it is determined that the processor p is the processor which originally processes the VMQ belonging to the tenant T_i , it may be verified whether a processor q which does not originally belong to the tenant T_i is present among the processors to process the VMQs belonging to the tenant T_i .

When the processor q which does not originally belong to the tenant T_i is present, all VMQs mapped onto the processor q which does not originally belong to the tenant T_i may be remapped onto the processor p to process the VMQ originally belonging to the tenant T_i .

The processor q may be remapped onto the tenant T_j to which the processor q originally belongs, and the TBDPM coalescing process may be finished.

When the processor q which does not originally belong to the tenant T_i is absent, the TBDPM coalescing process may be performed using all the processors to process the VMQs belonging to the tenant T_i .

Finally, the processor p is allowed to be idle, and the TBDPM coalescing process may be finished.

IV. CONCLUSIONS

In this paper, we presented a novel dynamic processor mapping method for Cloud Network Function Virtualization System. By dynamically map the virtual machine queues onto the processors by tenant in proportion to either a total number of virtual machine queues belonging to the same tenant, or a total number of virtual machine queues belonging to the same tenant which ensures that network traffic processing of virtual

machines belonging to the same tenant is not affected by congestion in network traffic of virtual machines belonging to different tenant.

We finally described the operation of the Tenant based Dynamic Processor Mapping including both spreading process and coalescing process with flow chart.

ACKNOWLEDGMENT

This work was supported by the ICT R&D program of MSIP/IITP. [B0101-15-233, Smart Networking Core Technology Development]

REFERENCES

- [1] James T. Pinkerton, Sanjay N. Kaniyar, Bhupinder S. Sethi, “Receive side scaling with cryptographically secure hashing”, US7765405 B2, Feb 25, 2005
- [2] Sambhrama Madhusudhan Mundkur, Alireza Dabagh, “Non-disruptive, reliable live migration of virtual machines with network data reception directly into virtual machines’ memory”, US7996484 B2, Dec 11, 2008
- [3] Pankaj Garg, Alireza Dabagh, Meiying Li, “Dynamic processor mapping for virtual machine network traffic queues”, US8984526 B2, Mar 9, 2012
- [4] Kangil Choi, Bhum Cheol Lee, Jung Hee Lee, Jong Geun Park, Sang Min Lee, “Apparatus and Method for Mapping of Tenant Based Dynamic Processor”, US 14/621,877, Feb 13, 2015



Kang Il Choi received B.S. degree in Computer Science from KAIST, Korea and M.S. degree in Computer Science from Sogang University in 1992 and 1994, respectively. He is currently senior researcher of Electronics and Telecommunications Research Institute (ETRI), Korea. His research interests are Multicore Parallel Processing, Distributed Cloud Data Center, Data Plane Acceleration Technology (Intel DPDK, ODP etc) and Network Virtualization.



Bhum Cheol Lee received M.S. and Ph.D. degree in Electric Engineering from Yonsei University, Korea in 1983 and 1997, respectively. He is currently Manager of Networking Computing Convergence Lab. in Electronics and Telecommunications Research Institute (ETRI), Korea. His research interests are Smart Network, Parallel Flow Processing and Network Virtualization.