DAQS: Dynamic and Accurate QoS for SR-IOV

Sungmin Kang
College of Computing and Informatics
Sungkyunkwan University
Suwon, Republic of Korea
ksmin1114@g.skku.edu

Joonyong Hwang
College of Computing and Informatics
Sungkyunkwan University
Suwon, Republic of Korea
brian0316@skku.edu

Younghoon Kim
College of Computing and Informatics
Sungkyunkwan University
Suwon, Republic of Korea
yhoon@skku.edu

***Abstract*—Single Root Input/Output Virtualization (SR-IOV) is a key feature in the PCI Express (PCIe) specifications, which facilitates the shared use of the same device in a virtualized environment by different virtual machines (VMs) and enables network traffic to bypass the usual virtualization stack, reducing interference among Virtual Functions (VFs). However, guaranteeing Quality of Service (QoS) as mandated by Service Level Agreements (SLAs) in these environments is challenging due to the host stack bypassing. This paper discusses these challenges and proposes an approach to ensure accurate QoS in SR-IOV environments by monitoring and regulating traffic from the host side through QEMU Guest Agent. The proposed approach automatically manages active VFs to meet SLA while fully utilizing the host bandwidth.**

***Index Terms*—Network Virtualization, SR-IOV, Service Level Agreement, Quality of Services**

I. Introduction

Single Root Input/Output Virtualization (SR-IOV), a feature specified in the PCI Express specifications [1], significantly improves system performance by allowing a single Network Interface Card (NIC) to be shared within a virtualized environment [2]. It accomplishes this by direct hardware-level virtualization, which enables physical NIC resources to be shared across multiple virtual environments. As shown in Fig. 1, the traditional network stack for virtual machines (VMs) requires network traffic to be routed through a software-based virtualization layer. In contrast, SR-IOV bypasses this layer and enables direct communication from the Virtual Function (VF) to the VF device driver within the VMs. This results in improved network and Input/Output (I/O) performance, along with reduced CPU utilization. Furthermore, SR-IOV provides better resource isolation and security by assigning unique VFs to each VM. These advantages make SR-IOV a preferred choice in a large data center [3].

Despite the apparent benefits of SR-IOV, ensuring Quality of Service (QoS) with a given Service Level Agreement (SLA) is not easy due to its inherent properties [2]. Bypassing the host stack, a key performance enhancement feature of SR-IOV, makes ensuring QoS more difficult by making the network monitoring of the VFs hard from the host. But there are several workarounds, such as host-driven controls such as `ip link` in the `iproute2` open-source project [4], or `mlnx_qos` [5] introduced by Mellanox, a major manufacturer of high-speed NICs. However, these approaches are challenging to guarantee minimum bandwidth, as they cannot dynamically account for idle guests, or require complex setup for external devices such as network switches.

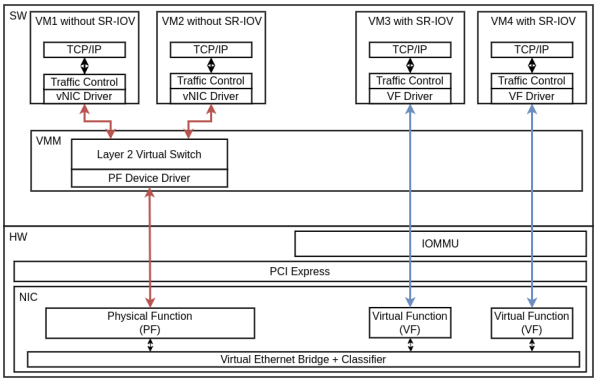

Fig. 1. Network stack for VM with and without SR-IOV

To address these challenges, we introduce a novel method named DAQS, Dynamic and Accurate QoS for SR-IOV, to ensure precise QoS in the SR-IOV environment. DAQS incorporates Traffic Control (TC) within the guest kernel, under the host's supervision. It continuously monitors active VMs and each VF's network resource usage, adjusting the guest's TC via the QEMU Guest Agent to meet the SLA. This strategy offers automatic QoS control, ensuring consistent minimum bandwidth while dynamically optimizing the overall network resource use across all VMs.

II. Problem Statement

To control QoS in the SR-IOV environment, utilities such as `ip link` or Mellanox's `mlnx_qos` are often considered. These tools regulate traffic for individual VFs by setting bandwidth caps using two main mechanisms: 1) sysfs-based configurations and 2) VLAN tagging. For example, the command '`ip link set vf max_tx_rate`' configures bandwidth limits per VF through `sysfs` file updates, ultimately adjusting the register value of NIC. The VLAN-based approach classifies traffic by assigning VLAN tags to each VF and hierarchically prioritizing traffic based on these classes [5]. Both packet classification techniques fundamentally depend on the NIC hardware implementation. As a result, some '`ip link`' features such as `min_tx_rate` are not supported by certain hardware, and QoS with VLAN tagging requires complex management of host and network switch configurations.

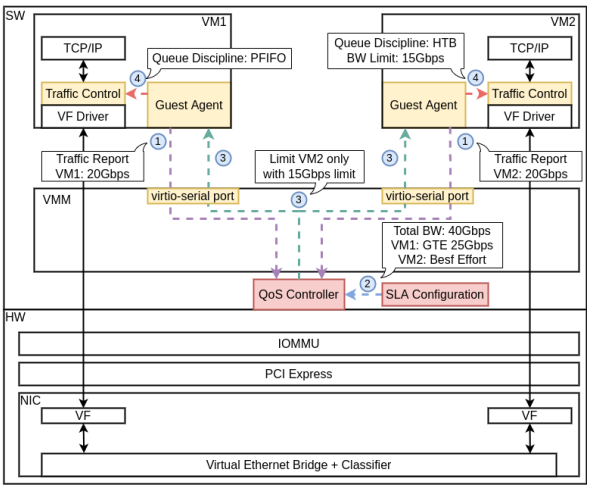

Fig. 2. DAQS system design and traffic managing scenario

In addition, these two utilities are fundamentally designed to restrict the maximum bandwidth, leading to certain limitations when to guarantee minimum bandwidth. Consider a dual-VM scenario where one VM is confined to 10Gbps to allocate 30Gbps for the second VM; the first VM's bandwidth cannot use 40Gbps while remains limited to 10Gbps, regardless of whether the second VM is active. This results in an overall underutilization of network resources. In the context of a cloud environment, where maximizing resource utilization and maintaining flexibility are paramount [6], these limitations highlights the need for more dynamic and precise tools to ensure minimum network bandwidth.

III. System Design

As mentioned above, SR-IOV bypasses the host network stack as shown in Fig. 1 and the host cannot observe the guest bandwidth. In utilizing `mlnx_qos`, several challenges emerge, such as intricate configurations with switches and the lack of complete availability across various hardware and system configurations. To overcome these problems, we propose DAQS, a new methodology that continuously manages traffic by monitoring network usage and regulating bandwidth.[1] DAQS ensures minimum bandwidth while maximizing bandwidth utilization for all guests. As presented in Fig. 2, a dedicated QoS Controller daemon operates in the guest. This daemon handles the following tasks:

1) Checking the current network bandwidth being used by active guests from the Guest Agent
2) Identifying which guest requires a certain minimum bandwidth from the prescribed SLA Configuration
3) Calculating the optimal policy based on this information
4) Enforcing the TC setting with each guest's Guest Agent via the virtio-serial port.

`QEMU virsh` and `Guest Agent` are used to check the status of the VM and network usage status, while TC, a Linux traffic control tool, is used for bandwidth control.

[1]The DAQS Model, testing and evaluation scripts, and daemons are available at *https://github.com/Brian-Hwang/DAQS*.

The host daemon acquires each VM's network usage, reads the SLA Configuration, and categorizes VMs into Guarantee VM and Best-Effort VM. Guarantee VMs are target VMs with SLA to guarantee minimum bandwidth, whereas Best-Effort VMs utilize leftover bandwidth. If a Guarantee VM exceeds its allocated bandwidth by more than a certain amount, it loosens the TC constraint on Best-Effort VMs. If the Guarantee VM's throughput is below the specified SLA, the daemon evaluates the total bandwidth to ascertain whether the Guarantee VM can fulfill the SLA. When the host bandwidth is underutilized and the Guarantee VM's throughput is below the SLA, the daemon determines that it is actually generating lower bandwidth than the given SLA. Thus the daemon loosens limits on Best-Effort VMs and maximizes overall bandwidth utilization. Conversely, if all VMs utilize more than a certain percentage of host bandwidth, the daemon imposes stricter TC constraints on Best-Effort VMs to enforce the Guaranteed VM's SLA.

IV. Evaluation

In order to identify the bottleneck of SR-IOV in data transmission and to see how much each traffic control tool affects the overall bandwidth while ensuring minimum bandwidth to specific VMs, we divided the experiment into *baseline*, which does not use any traffic control tool, *ip link*, which controls the `max_tx_rate` of individual VFs via `ip link`, and *DAQS*, the proposed traffic management method. In this experiment neither `mlnx_qos` nor `min_tx_rate` in `ip_link` is compared. `mlnx_qos` needs to set up the Enhanced Transmission Selection (ETS) configuration throughout VLANs on the host and switch, making hardly achievable to immediately change and apply the switch configuration in an environment where the number of VMs changes. Most of all, both two do not support all hardware and system configuration environments.

The bandwidth was measured by sending TCP traffic using `iperf2`. We used a Mellanox Connect X-5 40Gbps NIC (MCX516A-BDAT) on each host. One port on each host was connected to a Mellanox SN2100 switch via a QSFP28 interface. QEMU/KVM was used for virtualization technology, and QEMU Guest Agent was installed on all guests to connect to the hosts via the virtio-serial port. Each host was configured with SR-IOV enabled, Intel VT-d, and IOMMU, and eight VFs were set on the ports of each NIC.

A. Utilization of Total Bandwidth

We conducted an experiment to see how bandwidth is guaranteed by each method in a situation where a Guarantee VM and a Best-Effort VM are in contention. In the baseline with no bandwidth control, we found that bandwidth tends to be distributed proportionally to the number of CPU threads used. To simulate a situation where the Guarantee VM's SLA is violated by the Best-Effort VM, the Guarantee VM was given one thread using the `-P` option in `iperf2`, and the Best-Effort VM was given five threads using the same option. Various bandwidth guarantees were tested ranging from 10Gbps to 30Gbps, and for the `ip link`, we set a maximum bandwidth limit on the Best-Effort VM with `max_tx_rate`

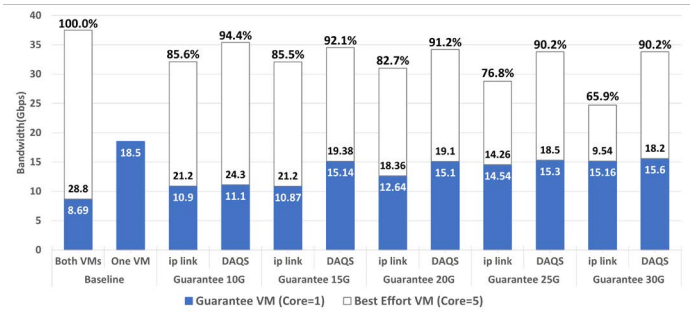

Fig. 3. Utilization of Guarantee VM, Best-Effort VM, and total bandwidth

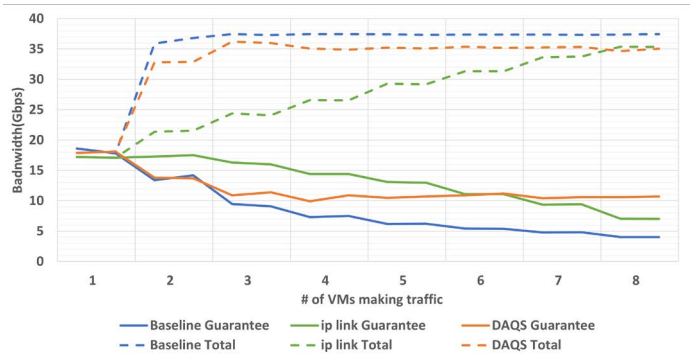

Fig. 4. Adaptability in multi-tenant environment

of the difference between the total bandwidth, 40Gbps, and the bandwidth to guarantee.

As shown in Fig. 3, The throughput of the Guarantee VM is compromised at the baseline (Leftmost bar). For the 10Gbps and 15Gbps guarantee cases, `ip link` can successfully guarantee 10Gbps but not 15Gbps, while DAQS can guarantee both well. For `ip link` to guarantee a 15Gbps on the guarantee VM, we can see that it requires down to a 10Gbps limit on the Best-effort VM, resulting in 65.9% of the total bandwidth utilization.

In addition, for both cases, the total baseline bandwidth of 37.49Gbps, DAQS utilizes about 93% compared to `ip link`'s 86%. This is because DAQS tries to determine whether the Guarantee VM is fully utilizing its share. For example, the maximum throughput for Guarantee VM is about 15Gbps because a single thread can achieve 18.5 Gbps even without any contentions. DAQS then allocates the remaining resources to the Best-Effort VM, enhancing overall bandwidth utilization. This uncapped allocation showcases DAQS's efficiency in maximizing the use of available resources.

B. Adaptability and Scalability in Dynamic Environment

In this section, we check the adaptability of DAQS with increasingly appearing bandwidth-consuming processes in separate VMs from zero to seven. All eight VMs, one Guarantee VM and seven Best-Effort VMs, are running from the beginning but only Guarantee VM has a bandwidth-consuming process. After that, each Best-Effort VM turns on a bandwidth-consuming process one by one every minute. In this scenario, the Guarantee VM uses one thread with a 10Gbps bandwidth guarantee, and each Best-Effort VM uses two threads.

Fig. 4 shows the achieved throughput of the baseline, `ip link`, and DAQS under the above experiment scenario. Here we can see that turning on a VM has no impact if it is not generating traffic. Even when other processes start generating traffic, the Guarantee VM's throughput is still guaranteed with DAQS regardless of the number of bandwidth-consuming processes, and the overall bandwidth is 93% of the baseline similar to what we observed in the prior experiment. When using `ip link`, traffic limit must be allocated to individual VFs. As it is incapable of detecting how much traffic VMs are generating, it has no choice but configures the traffic limit in advance based only on the number of existing VFs not on the amount of generated traffic. This limitation of `ip link` indicates that it cannot fully utilize the total bandwidth when only a small number of Best-Effort VMs are generating traffic as shown in Fig. 4. When numerous Best-Effort VMs are operational, the throughput of the Guarantee VM is not guaranteed similar to the previous experiment.

V. Conclusion

This paper presents a novel scheme, DAQS, to ensure QoS in SR-IOV environments by controlling the traffic from the host side to the guest. Conventional SR-IOV setups often result in an imbalance of network usage, leading to considerable inaccuracies in control and performance degradation. With the ability to guarantee minimum bandwidth with the given SLA, while fully utilizing the total bandwidth, DAQS is expected to enhance user management accuracy in multi-user scenarios on cloud platforms.

Nonetheless, `Qemu Guest Agent` used in DAQS is limited to the `QEMU/KVM` Environment, although there exist tools such as `vSphere Web Services API` from `VMWare` or the `Hyper-V Integration Services` from `Hyper-V`. Also, employing the `QEMU Guest Agent` necessitates the daemon to be operational within the guest. In instances involving malicious users, there might be apprehensions about data integrity. Hence, further research is to establish a more trustworthy, and platform-independent backchannel, perhaps by modification of driver code instead of relying on an application-level management daemon.

References

[1] S. D. Bhosale *at el.*, "IBM Power Systems SR-IOV Technical Overview and Introduction", IBM International Technical Support Organization, 2014.

[2] Y. Dong, X. Yang, X. Li, J. Li, K. Tian and H. Guan, "High performance network virtualization with SR-IOV," HPCA - 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture, Bangalore, India, 2010, pp. 1–10.

[3] G.K. Lockwood, M. Tatineni, and R. Wagner, "SR-IOV: Performance benefits for virtualized interconnects," In Proceedings of the 2014 Annual Conf. on Extreme Science and Engineering Discovery Environment, pp. 1–7, July 2014.

[4] iproute2, https://github.com/shemminger/iproute2

[5] MLNX_OFED Documentation, Rev 5.0-2.1.8.0, NVIDIA, April 23, 2020, Accessed: August 18, 2023. [Online]. Available: https://docs.nvidia.com/networking/display/OFEDv502180/

[6] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, "Efficient Resource Provisioning in compute clouds via VM multiplexing," Proceedings of the 7th international conference on Autonomic computing, Jun. 2010.