NetVM: High Performance and Flexible Networking using Virtualization on Commodity Platforms

Jinho Hwang[†] K. K. Ramakrishnan[*] Timothy Wood[†]
[†]*The George Washington University* [*]*WINLAB, Rutgers University*

Abstract

NetVM brings virtualization to the Network by enabling high bandwidth network functions to operate at near line speed, while taking advantage of the flexibility and customization of low cost commodity servers. NetVM allows customizable data plane processing capabilities such as firewalls, proxies, and routers to be embedded within virtual machines, complementing the control plane capabilities of Software Defined Networking. NetVM makes it easy to dynamically scale, deploy, and reprogram network functions. This provides far greater flexibility than existing purpose-built, sometimes proprietary hardware, while still allowing complex policies and full packet inspection to determine subsequent processing. It does so with dramatically higher throughput than existing software router platforms.

NetVM is built on top of the KVM platform and Intel DPDK library. We detail many of the challenges we have solved such as adding support for high-speed inter-VM communication through shared huge pages and enhancing the CPU scheduler to prevent overheads caused by inter-core communication and context switching. NetVM allows true zero-copy delivery of data to VMs both for packet processing and messaging among VMs within a trust boundary. Our evaluation shows how NetVM can compose complex network functionality from multiple pipelined VMs and still obtain throughputs up to 10 Gbps, an improvement of more than 250% compared to existing techniques that use SR-IOV for virtualized networking.

1 Introduction

Virtualization has revolutionized how data center servers are managed by allowing greater flexibility, easier deployment, and improved resource multiplexing. A similar change is beginning to happen within communication networks with the development of virtualization of network functions, in conjunction with the use of software defined networking (SDN). While the migration of network functions to a more software based infrastructure is likely to begin with edge platforms that are more "control plane" focused, the flexibility and cost-effectiveness obtained by using common off-the-shelf hardware and systems will make migration of other network functions attractive. One main deterrent is the achievable performance and scalability of such virtualized platforms compared to purpose-built (often proprietary) networking hardware or middleboxes based on custom ASICs.

Middleboxes are typically hardware-software packages that come together on a special-purpose appliance, often at high cost. In contrast, a high throughput platform based on virtual machines (VMs) would allow network functions to be deployed dynamically at nodes in the network with low cost. Further, the shift to VMs would let businesses run network services on existing cloud platforms, bringing multiplexing and economy of scale benefits to network functionality. Once data can be moved to, from and between VMs at line rate for all packet sizes, we approach the long-term vision where the line between data centers and network resident "boxes" begins to blur: both software and network infrastructure could be developed, managed, and deployed in the same fashion.

Progress has been made by network virtualization standards and SDN to provide greater configurability in the network [1–4]. SDN improves flexibility by allowing software to manage the network control plane, while the performance-critical data plane is still implemented with proprietary network hardware. SDN allows for new flexibility in how data is forwarded, but the focus on the control plane prevents dynamic management of many types of network functionality that rely on the data plane, for example the information carried in the packet payload.

This limits the types of network functionality that can be "virtualized" into software, leaving networks to continue to be reliant on relatively expensive network appliances that are based on purpose-built hardware.

Recent advances in network interface cards (NICs) allow high throughput, low-latency packet processing using technologies like Intel's Data Plane Development Kit (DPDK) [5]. This software framework allows end-host applications to receive data directly from the NIC, eliminating overheads inherent in traditional interrupt driven OS-level packet processing. Unfortunately, the DPDK framework has a somewhat restricted set of options for support of virtualization, and on its own cannot support the type of flexible, high performance functionality that network and data center administrators desire.

To improve this situation, we have developed NetVM, a platform for running complex network functionality at line-speed (10Gbps) using commodity hardware. NetVM takes advantage of DPDK's high throughput packet processing capabilities, and adds to it abstractions that enable in-network services to be flexibly created, chained, and load balanced. Since these "virtual bumps" can inspect the full packet data, a much wider range of packet processing functionality can be supported than in frameworks utilizing existing SDN-based controllers manipulating hardware switches. As a result, NetVM makes the following innovations:

1. A virtualization-based platform for flexible network service deployment that can meet the performance of customized hardware, especially those involving complex packet processing.
2. A shared-memory framework that truly exploits the DPDK library to provide zero-copy delivery to VMs and between VMs.
3. A hypervisor-based switch that can dynamically adjust a flow's destination in a state-dependent (e.g., for intelligent load balancing) and/or data-dependent manner (e.g., through deep packet inspection).
4. An architecture that supports high speed inter-VM communication, enabling complex network services to be spread across multiple VMs.
5. Security domains that restrict packet data access to only trusted VMs.

We have implemented NetVM using the KVM and DPDK platforms– all the aforementioned innovations are built on the top of DPDK. Our results show how NetVM can compose complex network functionality from multiple pipelined VMs and still obtain line rate throughputs of 10Gbps, an improvement of more than 250% compared to existing SR-IOV based techniques. We believe NetVM will scale to even higher throughputs on machines with additional NICs and processing cores.

2 Background and Motivation

This section provides background on the challenges of providing flexible network services on virtualized commodity servers.

2.1 Highspeed COTS Networking

Software routers, SDN, and hypervisor based switching technologies have sought to reduce the cost of deployment and increase flexibility compared to traditional network hardware. However, these approaches have been stymied by the performance achievable with commodity servers [6–8]. These limitations on throughput and latency have prevented software routers from supplanting custom designed hardware [9–11].

There are two main challenges that prevent commercial off-the-shelf (COTS) servers from being able to process network flows at line speed. First, network packets arrive at unpredictable times, so interrupts are generally used to notify an operating system that data is ready for processing. However, interrupt handling can be expensive because modern superscalar processors use long pipelines, out-of-order and speculative execution, and multi-level memory systems, all of which tend to increase the penalty paid by an interrupt in terms of cycles [12, 13]. When the packet reception rate increases further, the achieved (receive) throughput can drop dramatically in such systems [14]. Second, existing operating systems typically read incoming packets into kernel space and then copy the data to user space for the application interested in it. These extra copies can incur an even greater overhead in virtualized settings, where it may be necessary to copy an additional time between the hypervisor and the guest operating system. These two sources of overhead limit the the ability to run network services on commodity servers, particularly ones employing virtualization [15, 16].

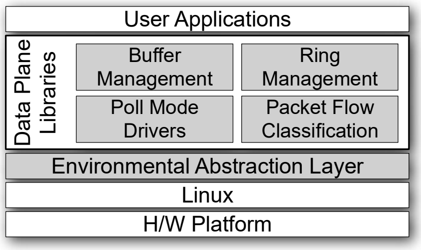

Figure 1: DPDK's run-time environment over Linux.

The Intel DPDK platform tries to reduce these overheads by allowing user space applications to directly poll the NIC for data. This model uses Linux's huge pages to pre-allocate large regions of memory, and then allows applications to DMA data directly into these pages. Figure 1 shows the DPDK architecture that runs in the application layer. The poll mode driver allows applications to access the NIC card directly without involving kernel processing, while the buffer and ring management systems resemble the memory management systems typically employed within the kernel for holding `sk_buffs`.

While DPDK enables high throughput user space applications, it does not yet offer a complete framework for constructing and interconnecting complex network services. Further, DPDK's passthrough mode that provides direct DMA to and from a VM can have significantly lower performance than native IO[1]. For example, DPDK supports Single Root I/O Virtualization (SR-IOV[2]) to allow multiple VMs to access the NIC, but packet "switching" (i.e., demultiplexing or load balancing) can only be performed based on the L2 address. As depicted in Figure 2(a), when using SR-IOV, packets are switched on

[1] Until Sandy-bridge, the performance was close to half of native performance, but with the next generation Ivy-bridge processor, the claim has been that performance has improved due to IOTLB (I/O Translation Lookaside Buffer) super page support [17]. But no performance results have been released.

[2] SR-IOV makes it possible to logically partition a NIC and expose to each VM a separate PCI-based NIC called a "Virtual Function" [18].