NETRONOME

WHITE PAPER

Hardware Acceleration for Network Services

FINDING A MORE EFFICIENT WAY TO EXECUTE VIRTUAL SWITCHING IS CRITICAL TO MAKING SERVICE NODES PERFORM AT TODAY'S NETWORK SPEEDS.

CONTENTS

INTRODUCTION

Service nodes typically switch network traffic from one port to another, perhaps modifying headers or inspecting traffic along the way. In a traditional environment, this switching is done in software in the host operating system or hypervisor. This software or 'virtual' switching can consume more than 50% of the CPU, leaving few cycles to run actual applications. Finding a more efficient way to execute this virtual switching is critical to making service nodes perform at today's network speeds. This paper explores a method of offloading virtual switching from host CPUs to create an efficient platform for network service nodes.

VIRTUALIZATION REVOLUTION

Virtualization is an important concept in the computing industry. We started by virtualizing servers, which abstracts the hardware from the software that sits above it. This allowed multiple users and applications to share physical hardware, which drastically increased the efficiency of servers. Network service functions like firewalls, intrusion detection/prevention, lawful intercept, or network analytics that were previously made from dedicated hardware for individual organizations can now be virtualized to support multiple users, or tenants, simultaneously. In a multitenant environment, multiple groups of users can share a single firewall by keeping each tenant logically separate. Although a single physical switch may connect the service node to the network, virtual switching takes place within the firewall to keep the tenants' traffic logically separate from each other.

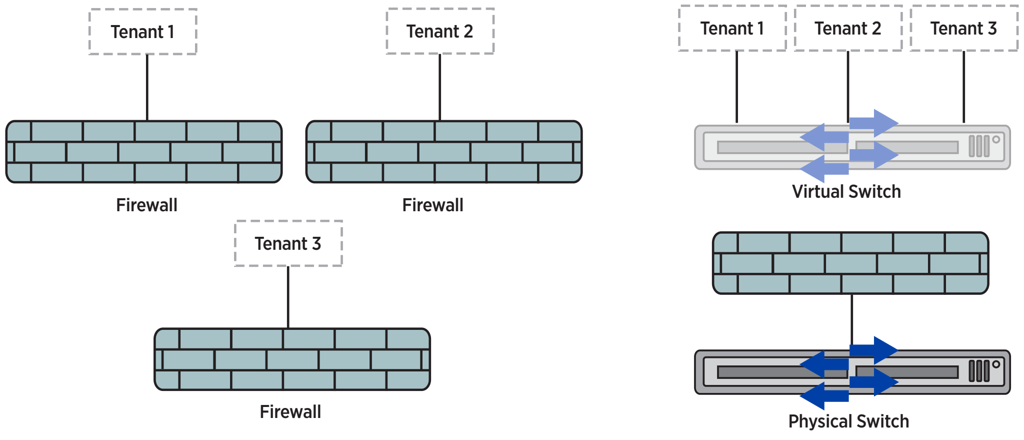


Figure 1. Virtualization of Applications for Multi-Tenancy

This virtual switch runs in software on each physical service node. It inspects incoming network traffic and determines which of the tenants should receive it. But the tasks of the virtual switch are not as simple as that. To mirror the functionality of a physical switch, several operations including encapsulation/decapsulation of headers, address translation, priority queuing, shaping and policing are required. These functions consume significant compute resources; in some cases over half of the available CPU cycles, leaving few remaining to run the virtual applications. Virtual switching is at the heart of a virtualized system, so it stands to reason that efficient systems must perform efficient virtual switching.

VIRTUAL SWITCHING IS AT THE HEART OF A VIRTUALIZED SYSTEM, SO IT STANDS TO REASON THAT EFFICIENT SYSTEMS MUST PERFORM EFFICIENT VIRTUAL SWITCHING

OPEN VSWITCH IN NETWORK SERVICE NODES

The most commonly deployed virtual switching software is Open vSwitch (OVS). OVS is a production quality, multilayer virtual switch licensed under the open source Apache 2.0 license and is available at openvswitch.org. It is the default switch in XenServer 6.0, the Xen Cloud Platform and also supports Xen, KVM, Proxmox VE and VirtualBox. It has also been integrated into many virtual management systems including OpenStack, openQRM, OpenNebula and oVirt. The kernel datapath is distributed with Linux, and packages are available for Ubuntu, Debian, and Fedora. OVS is also supported on FreeBSD and NetBSD.

Much effort has been invested to improve the performance of OVS, including porting portions to run in user space rather than kernel space. While these enhancements indeed offer some performance improvements, they still require CPU resources to execute the virtual switching functions.

The following graph shows measured results of throughput using Open vSwitch in a standard Linux distribution with GRE encapsulation. Especially at smaller packet sizes, the throughput falls far short of the theoretical maximum for 10 Gigabit Ethernet. The picture only gets worse at higher data rates.

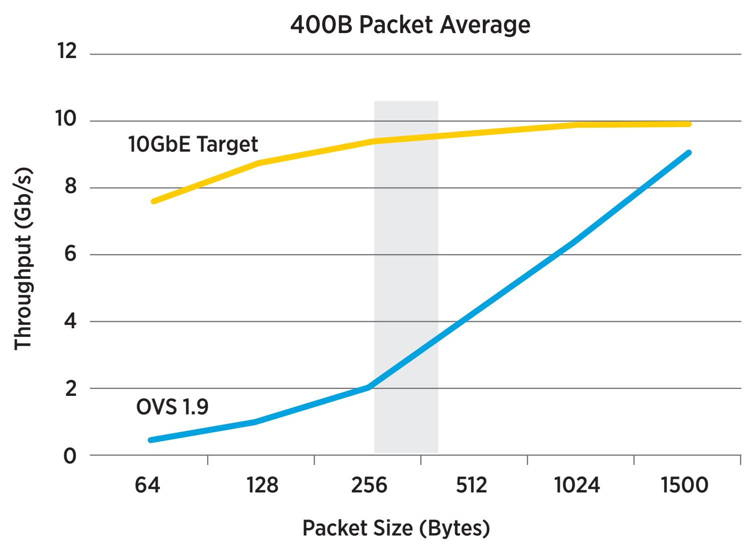


Figure 2. Standard Open vSwitch Throughput with GRE

OVS INTERNALS

In laymans terms, OVS in Linux consists of two major parts: a fastpath and a slow path. The slow path is taken whenever a new flow is encountered and decisions must be made about how to manage the traffic for that flow. Because this path need only be taken on the first packet of each new flow, the overhead associated with the 'slow path' is generally considered acceptable. After a flow is established, and entry is made in a Flow Table (FT) with a corresponding action for packets associated with that entry. All subsequent packets on that established flow can take the fastpath, which finds the flow entry in the flow table and executes the associated action. Examples of actions could be to forward the packet from a certain port, change an IP or Ethernet address, or drop the packet from further processing.

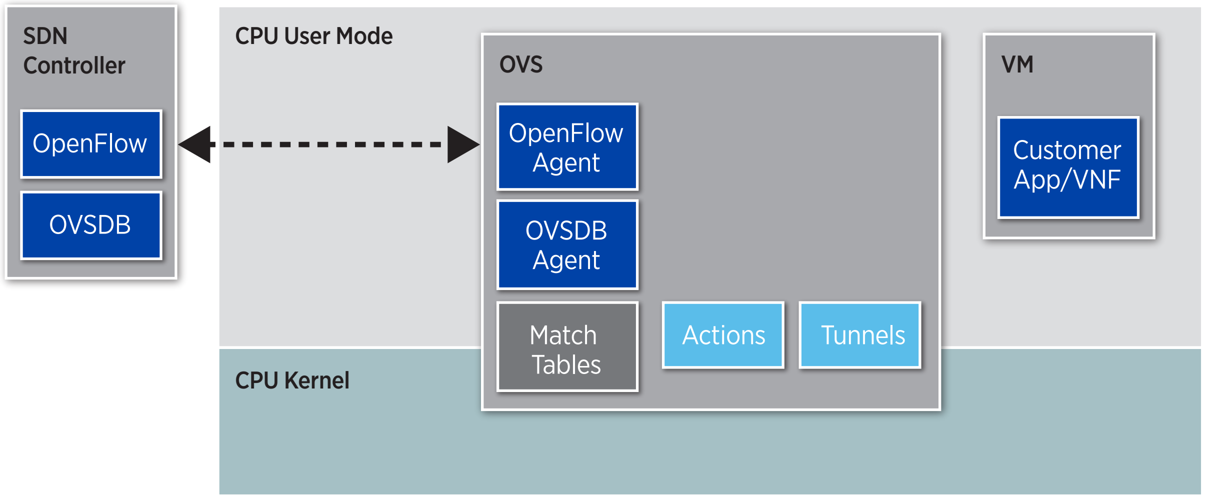


Figure 3. Open vSwitch Architecture

The 'fastpath' is so named relative to the 'slow path' and requires fewer time consuming cycles to traverse. However, even the fastpath of common OVS implementations may not be so fast in the context of overall system performance. In fact, adding tunneling overhead and multiple rules can expose OVS bottlenecks on network service nodes.

Many performance enhancements have been proposed to accelerate the fastpath, including moving the match/action logic into user space. These offer some advantages, but it's important to remember that any logic executed by the x86 burns CPU cycles, and those cycles are then not available to run network functions like firewalls, DPI, or virtual Evolved Packet Core (vEPC). CPU cores are not free or infinitely available to process virtual switching of network traffic in network service nodes.

NETRONOME OVS OFFLOAD

Netronome takes a different approach to accelerating OVS for network service nodes. Rather than use host CPU cycles to process virtual switching, Netronome offloads the entire fastpath to a SmartNIC. The first packet of each new flow takes the slow path, just as in all implementations of OVS. Subsequent packets which can be handled by the offload implementation will be processed entirely on the SmartNIC, while other packets are passed to the host CPU where they are processed as if they arrived via a traditional NIC. Flow Tables are kept in synchronization between the Agilio® SmartNIC and the host. This offload implementation mirrors the OVS kernel fastpath and is implemented in another kernel module using generic hooks in the OVS software. This offload is completely transparent to the OVS user-space daemons and tools: no modifications are necessary and the offload is simply enabled by loading the offload kernel module. Flow table entries can be populated via an OpenFlow controller which specifies which actions to take for any given flow. The OVS implementation can be configured via the Open vSwitch Database Management Protocol, or OVSDB.

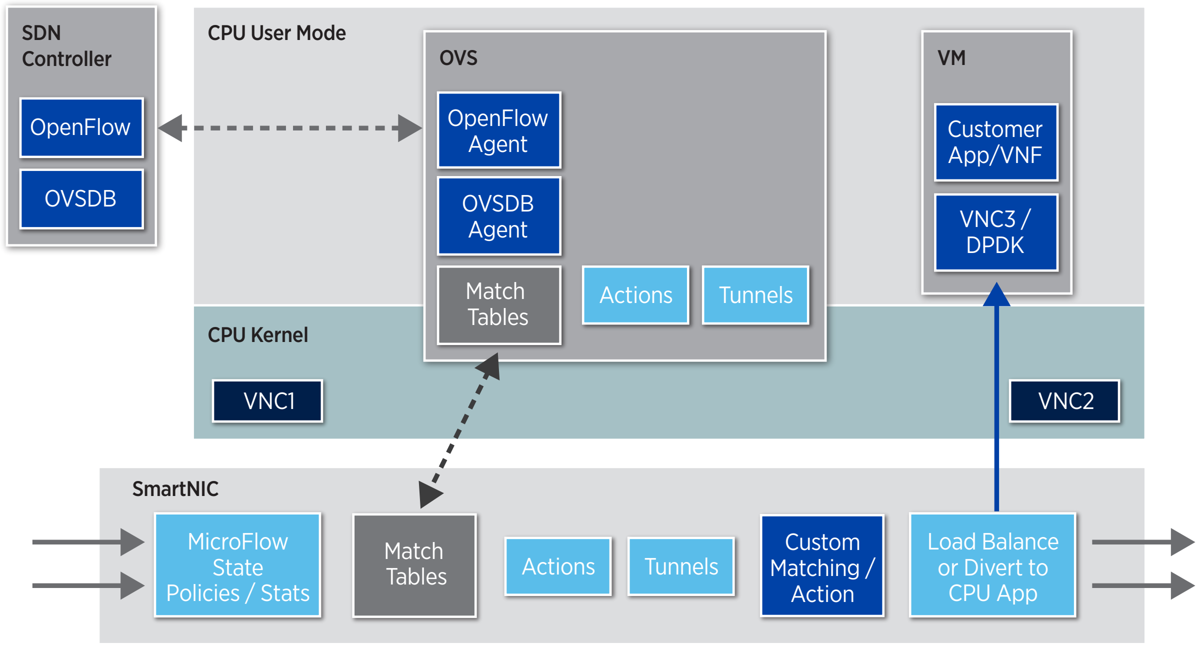


Figure 4. Netronome OVS Offload Solution

AGILIO SMARTNIC

The PCIe card used for Netronome OVS offload is the Agilio SmartNIC, and it features a Netronome Flow Processor. This flow processor employs over 200 individual processing engines to classify, prioritize, repackage, meter and route network traffic. The Agilio SmartNIC can provide efficient 10Gb/s, 25Gb/s, 40Gb/s and 100Gb/s Ethernet network interfaces. Netronome offloads OVS processing to the SmartNIC, freeing up CPU cycles in the host. Linux drivers and the OVS kernel offload module come standard with the Agilio SmartNIC. Although the SmartNIC is programmable, this standard Agilio software provided is sufficient to accommodate most use cases. The benefit of programmability is that new protocols can be supported with future software updates, and custom applications can be supported.

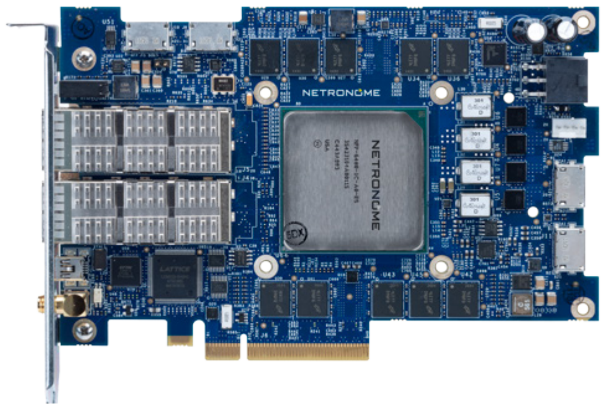


Figure 5. Netronome Agilio LX SmartNIC

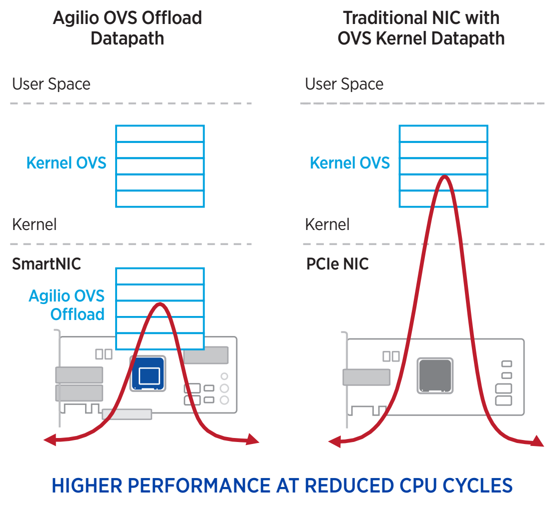
PERFORMANCE METRICS

Throughput is the typical metric considered when measuring networking performance. Posted in either gigabits per second or packets per second, this is a good indication of how efficiently traffic is being managed by the networking hardware and software. But throughput is only half of the relevant metric for network service nodes. The other half is measured in CPU utilization. The more CPU cycles are consumed by network processing, the fewer are available to execute network functions. An efficient network service node will not only deliver high throughput but will do so without consuming substantial CPU cycles.

The following diagram shows the performance differences, in both throughput and CPU cycle savings, of a Netronome Agilio LX SmartNIC and a traditional network interface card for the given number of flows and rules which might be found on a network service node.

As the diagram shows, the benefits of offloading OVS processing from the host CPU is two-fold: Throughput is higher and CPU utilization is significantly lower.

L3 Forwarding Benchmark through OVS


Agilio OVS Offload Datapath
Traditional NIC with OVS Kernel Datapath
User Space
Kernel OVS
Kernel
SmartNIC
Agilio OVS Offload
PCIe NIC
HIGHER PERFORMANCE AT REDUCED CPU CYCLES

# of OpenFlow Entries	# of Concurrent Microflows	2x40G Agilio LX w/ OVS Offload PPS / CPU Load	Traditional 2x40G NIC w/ OVS in Kernel PPS / CPU Load
100	100	51.5 Mpps < 1% CPU	10.57 Mpps ~40% CPU
1024	1024	51.7 Mpps <1% CPU	8.78 Mpps >50% CPU
9728	9728	47.5 Mpps <1% CPU	7.6 Mpps 46% CPU
65536	65536	10.85 Mpps 1-2% CPU	6.06 Mpps 28% CPU

- These performance metrics were taken with the full OVS match/action set
- With action sets optimized to use cases, performance on the order of 70Mpps can be achieved

Figure 6. Netronome OVS Offload Performance Comparison

SUMMARY

The cost benefits of using standard COTS equipment for network service nodes are attractive, as is the speed and agility of new service deployment. Virtual switching is at the heart of every network service node, and efficient switching of network traffic is essential for making efficient networking platforms.

Netronome's Network Flow Processing technology offloads virtual switching from the host CPU to a PCIe Agilio SmartNIC, providing efficient switching of network traffic and returning valuable CPU cycles to the host to run network functions like virtual firewalls, virtual load balancers, vEPC, virtual Customer Premise Equipment (vCPE), and virtual Broadband Network Gateways (vBNG).

Netronome's range of SmartNICs support 10G, 25G, 40G and 100Gb/s Ethernet and multiple PCIe interfaces for maximum efficiency in network service nodes.

NETRONOME

Netronome Systems, Inc.
2903 Bunker Hill Lane, Suite 150 Santa Clara, CA 95054
Tel: 408.496.0022 | Fax: 408.586.0002
www.netronome.com