# Leveraging NIC Technology to Improve Network Performance in VMware vSphere

Performance Study

TECHNICAL WHITE PAPER

## Table of Contents

Introduction ..... 3

Hardware Description ..... 3

List of Features ..... 4

- NetQueue ..... 4
- Receive Side Scaling (RSS) ..... 4
- SplitRxMode ..... 6
- TCP Segmentation Offload and Checksum Offload of VXLAN Packets ..... 7
- Hardware Large Receive Offloads (LRO) ..... 8
- Dynamic NetQueue ..... 9
- Native Device Driver ..... 10

Conclusion ..... 10

References ..... 11

# Introduction

VMware vSphere® is constantly evolving to add many new features and functionality to incorporate the latest technology, and one area of this trend is network performance. vSphere supports a variety of network interface controllers (NICs). Each NICs primary job is to deliver packets in and out of the host; however, NICs also include technology that vSphere can leverage to perform network transfer faster and more efficiently. This paper examines scenarios of virtual machines sending and receiving network traffic across different hosts and how vSphere can take advantage of the NIC technology to improve network performance in a vSphere environment.

# Hardware Description

Two different sets of hardware were used for this performance study in order to accommodate several different NICs. Tests for Intel Corporation 82599EB 10-Gigabit, Broadcom Corporation NetXtreme II BCM57810 10 Gigabit Ethernet, and Emulex Corporation OneConnect 10GbE NICs were carried out on two identical HP DL380 G7 systems with 2-socket X5690 @ 3.47 GHz with 64GB RAM. Tests for Mellanox Technologies MT27500 Family [ConnectX-3] NICs were performed on two identical Dell PowerEdge R720 servers with Intel E5-2667 @ 2.90GHz and 64GB of memory. Both systems were installed with vSphere 5.5 and each system ran Red Hat Enterprise Linux 6 virtual machines with the VMXNET3 onboard driver included with the Linux release.

Each test was run with 1 virtual machine and 16 virtual machines, and an identical number of receiver virtual machines were configured on the client system. The single virtual machine was configured with 4 vCPUs, and each of the 16 virtual machines was configured with 1 vCPU. Each virtual machine was sending and receiving traffic from an identical virtual machine on the receiver side.

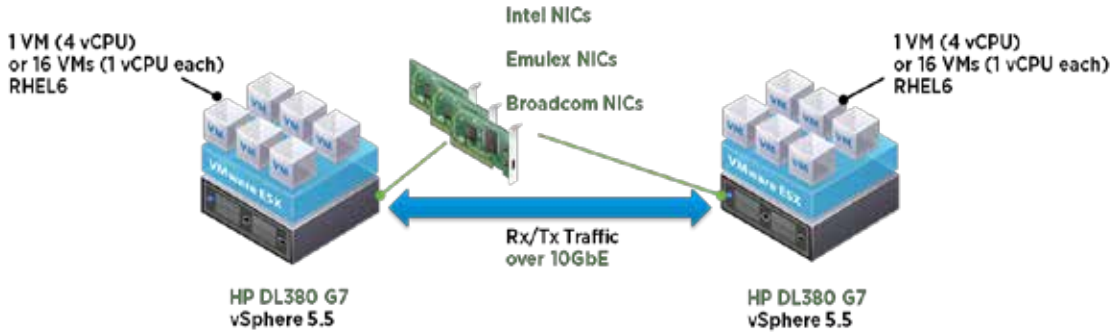


Figure 1. Test bed for Intel, Emulex, and Broadcom NIC experiments

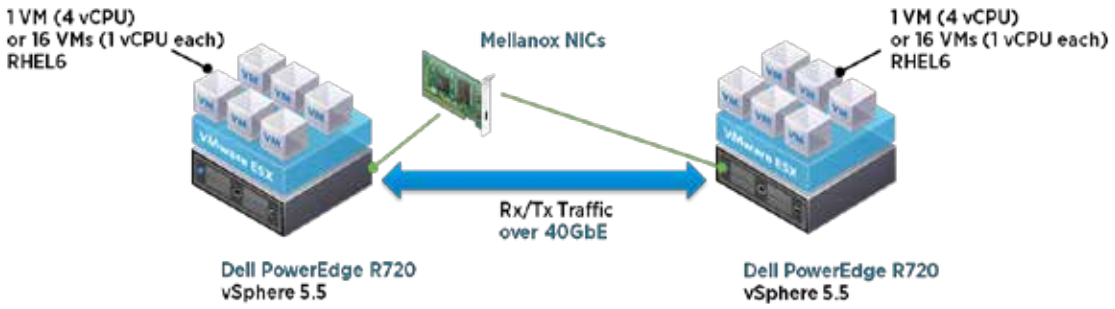


Figure 2. Test bed for Mellanox NIC experiments

# List of Features

## NetQueue

The first important feature available in current 10G/40G NICs is NetQueue. This feature was introduced in ESX 3.5 and is present in most NICs. This feature allows packets to be distributed among different physical queues and each queue gets a separate ESX thread for packet processing. vSphere programs the physical NIC to distribute packets based on certain attributes, called filters. Most NICs support outer MAC addresses as filters and as a result, packets with the same destination MAC addresses are handled by a single queue. This can be a huge limitation for VXLAN traffic where packets addressed to multiple virtual machines will have a single destination MAC address in the outer header. To overcome this problem, some physical NICs can filter based on the inner packet MAC address.

## Receive Side Scaling (RSS)

Receive side scaling was introduced in vSphere 5.1. RSS can be confused with NetQueue because the basic functionality is the same: both technologies distribute packets among different queues. However, there is a difference in the implementation. NetQueue operates on MAC address filters, which means that packets belonging to same MAC address will be processed by the same queue. Thus a virtual NIC (vNIC) can receive packets from only one queue and as a result can limit performance. RSS allows distribution based on flows and as a result a single vNIC with multiple flows can leverage parallelism in the pNIC and the ESX stack using multiple queues. The distribution can vary among various NICs as some NICs support 5-tuple hashing techniques while the others don't.

As seen in Figure 3, single virtual machine receive performance was evaluated for Mellanox 40GbE NICs. Without RSS, the throughput is limited to only 15Gbps for large packets, but when RSS is turned on for the pNIC and the virtual machine, the throughput increases by 40%. The configuration also helps in the small packet receive test case, and the throughput is increased by 30%.

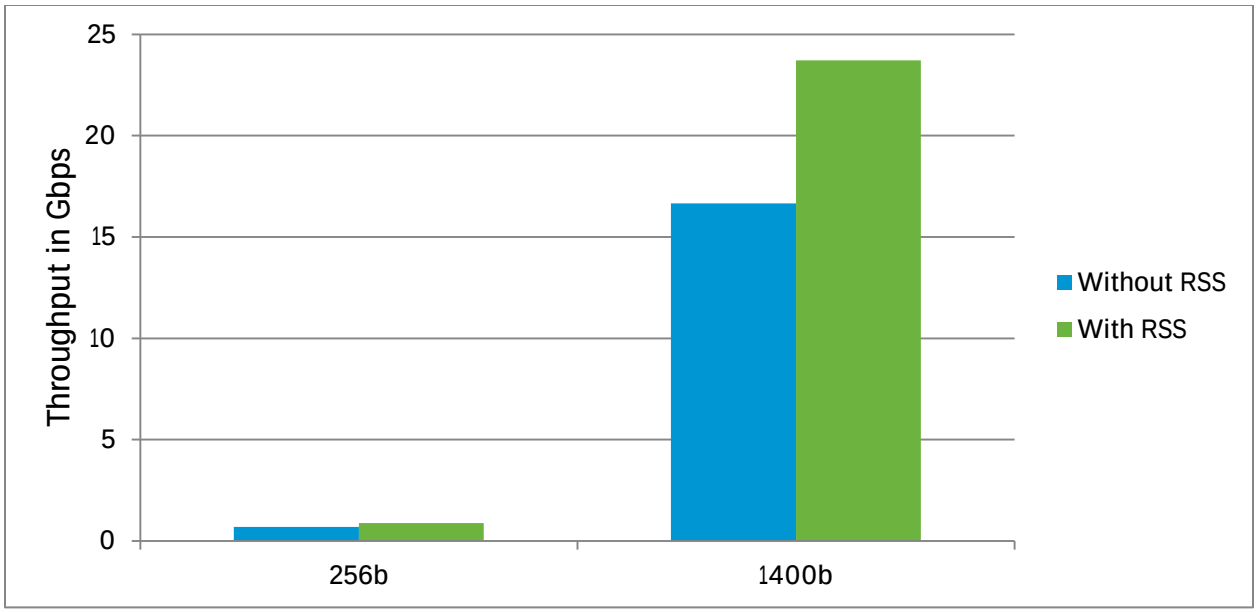


Figure 3. Throughput comparison for 1 VM with and without RSS with 40GbE NIC (higher is better)

Since a single MAC address can use multiple queues, this behavior is advantageous for achieving line rate performance for VXLAN traffic too. For VXLAN traffic, multiple destination virtual machines will have the same destination MAC address belonging to the VXLAN tunnel end point. As a result, traffic for these virtual machines will be processed by a single queue. This can be a huge limitation for throughput especially for large numbers of virtual machines. RSS helps in this case because traffic can be distributed among multiple hardware queues for VXLAN traffic.

As seen in Figure 4, NICs that offer RSS have a throughput of 9.1Gbps for large packet test cases, but NICs that do not only have a throughput of around 6Gbps. The throughput improvement for the small packet test case is also similar to that of the large packet test case.

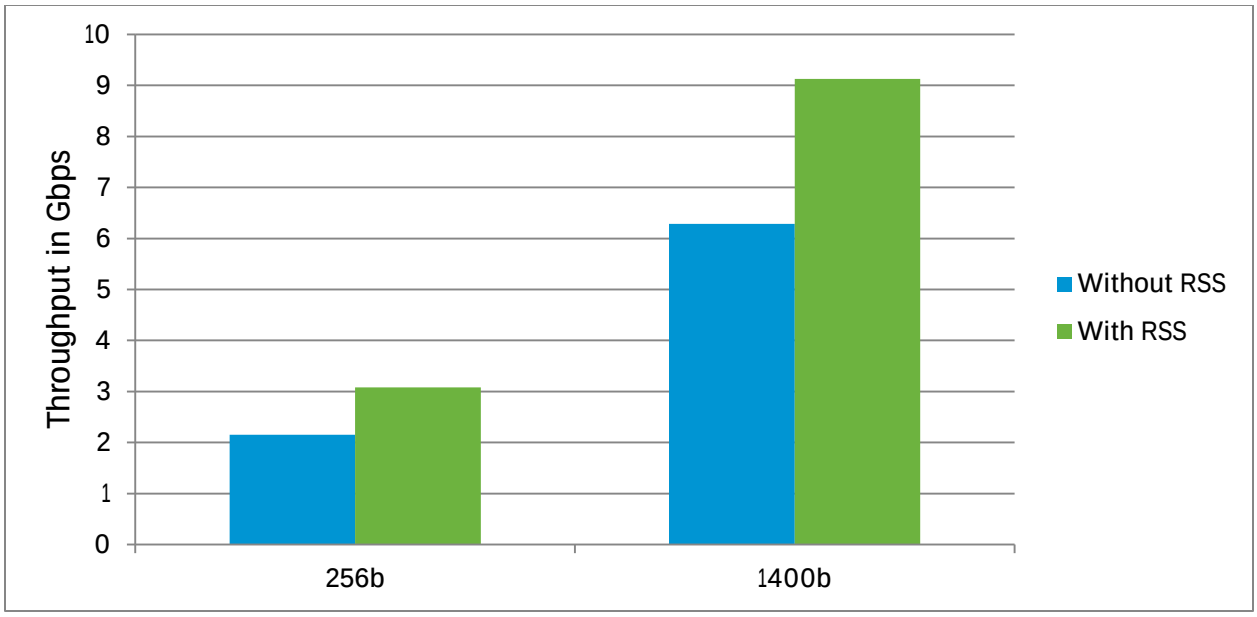


Figure 4. Receive throughput with VXLAN with and without RSS with 10GbE for 16 VMs

## SplitRxMode

For NICs that don't support RSS, there is a software configuration that could be equally beneficial. vSphere 5.0 introduced SplitRxMode, which allows some packet processing to be offloaded from the physical NIC kernel threads. It was beneficial for multicast because a single kernel thread has to copy packets for multiple virtual machines and can quickly create a CPU bottleneck. SplitRxMode will offload some work to a dedicated per VM kernel thread and as a result the hardware queue can process more packets. Since the behavior is similar in VXLAN too where a single kernel thread is processing packets for multiple VMs, the same benefit can be realized using splitRxMode.

As seen in Figure 5, splitRxMode helps improve performance for VXLAN traffic when no RSS is present. However there is a disadvantage of using splitRxMode, especially for the small packet test case. As work is queued up, the latency for small packets increases substantially and as a result the throughput for the small packet test case is reduced. Testing showed a 15% higher ping latency for virtual machines with splitRxMode turned on.

The other thing to note is that enabling SplitRxMode causes higher CPU overhead per Gbps of throughput and should only be turned on when necessary (when the system requires 10Gbps of traffic). For large packets, the CPU overhead is around 15%; while for small packets, the CPU overhead increases to about 50%.

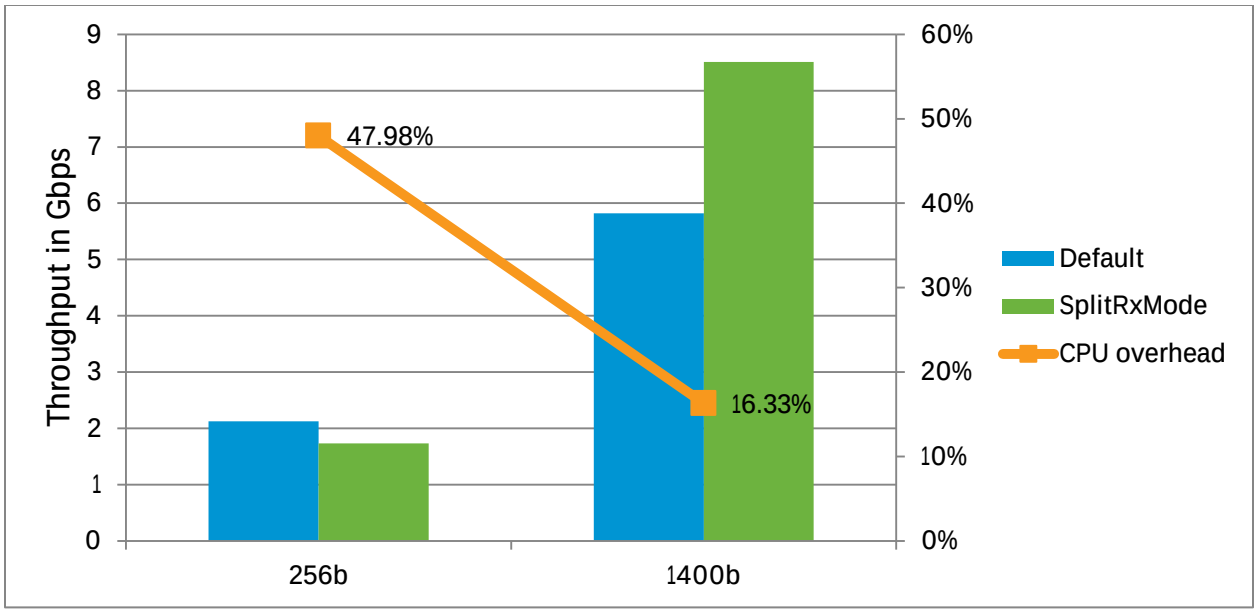


Figure 5. 16 Virtual machine max receive throughput with VXLAN and SplitRxMode

## TCP Segmentation Offload and Checksum Offload of VXLAN Packets

Networking virtualization has added another layer of intelligence to the networking stack. The networking packets need to be encapsulated in order to achieve true network virtualization. However, encapsulated packets, especially VXLAN, can cause severe performance degradation especially for TCP traffic because some of the offloads do not work for UDP packets since VXLAN packets are similar to UDP packets. The two important features that don't work are TSO segmentation and Rx Checksum offloads.

TSO segmentation allows vSphere to deliver large packets to the physical NIC, and the NIC will split the TSO packet to smaller MTU-size frames. If the NIC's don't support TSO segmentation, vSphere has to do all the packet segmentation in software. This causes higher CPU overhead per packet. As seen in Figure 6, NICs with TSO for VXLAN can achieve line rate performance without any overhead, but for NICs that don't, the CPU used is almost doubled.

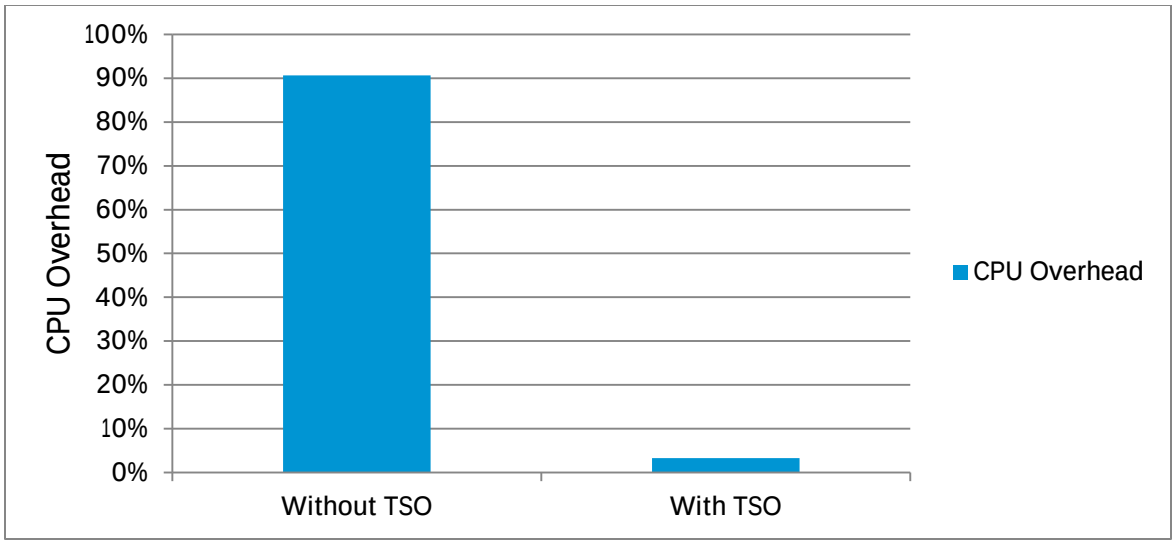


Figure 6. Send CPU comparison for NICs with and without TSO offloads for VXLAN – 16 VMs (lower is better)

Similar to send, several pNICs cannot execute receive side checksum offloads. As a result, there is a CPU penalty involved. However, since vSphere has to copy packets to the virtual machine buffers, checksum costs are not that high. As you can see in Figure 7, NICs that have checksum offloads on encapsulated packets can achieve line rate performance with 10% additional CPU overhead. The overhead for NICs that don't have this offload is about 15%.

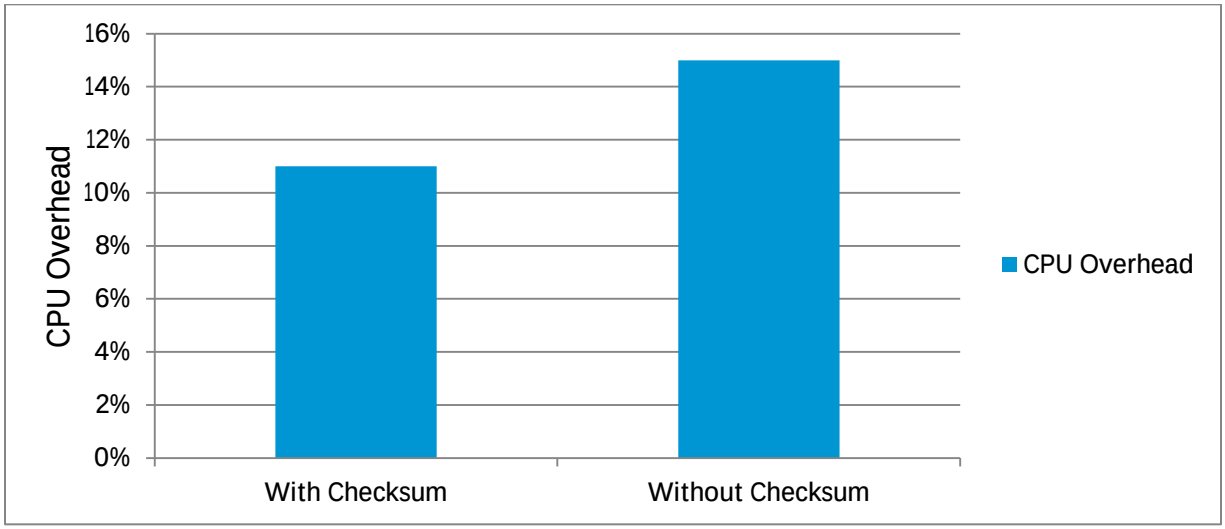


Figure 7. 16-VM CPU overhead for receive - comparison between NICs with and without Checksum (lower is better)

## Hardware Large Receive Offloads (LRO)

Some of the pNICs can support packet aggregation on receive. This results in significant CPU savings for the hypervisor. If the pNICs don't support the feature, the vSphere platform will aggregate packets and deliver it to the guest (if the guest supports it). Most Linux distributions support LRO and Windows 2012 also recently added support for the feature. As seen in Figure 8, NICs that have hardware LRO have significantly higher CPU efficiency for the large packet receive test cases compared to those NICs which don't.

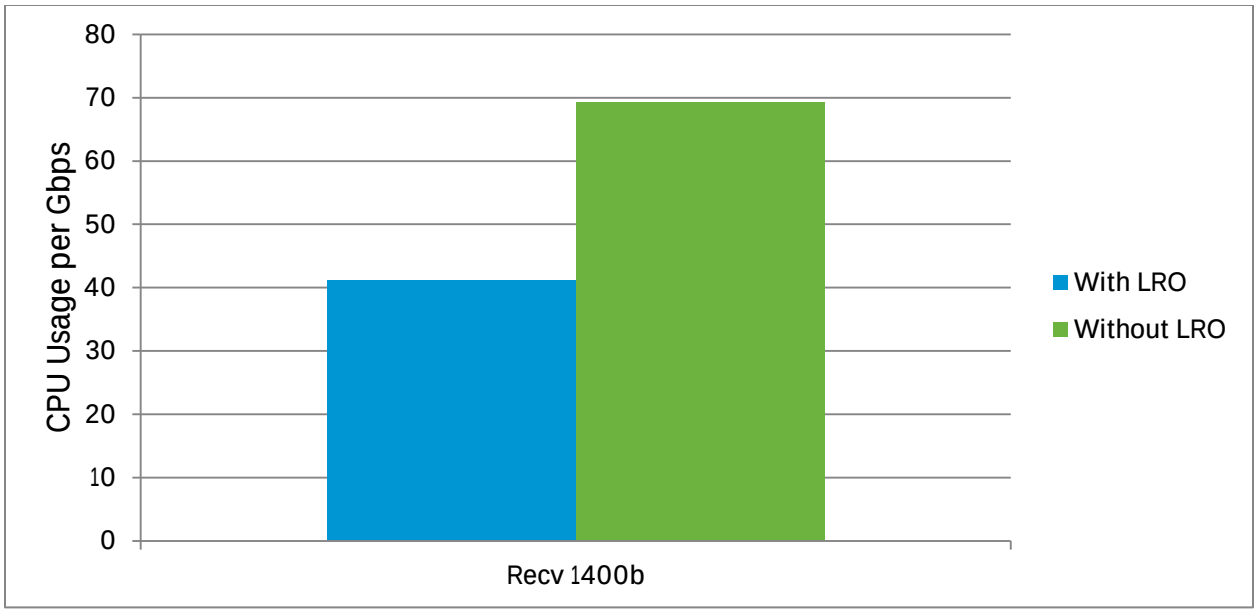


Figure 8. CPU efficiency comparison for NICs with hardware LRO and NICs without hardware LRO

## Dynamic NetQueue

vSphere 5.5 introduced Dynamic NetQueue for physical NIC adapters. Prior to vSphere 5.5, the networking load balancer would distribute NetQueues evenly among different virtual machines. This was good for scaling but can hurt CPU performance, especially due to higher interrupt rates. Dynamic NetQueue allows the load balancer to pack filters on a single queue as long as the load is not high. When the load increases beyond a certain threshold, the virtual machine filters can be distributed to the second queue. This leads to better CPU efficiency.

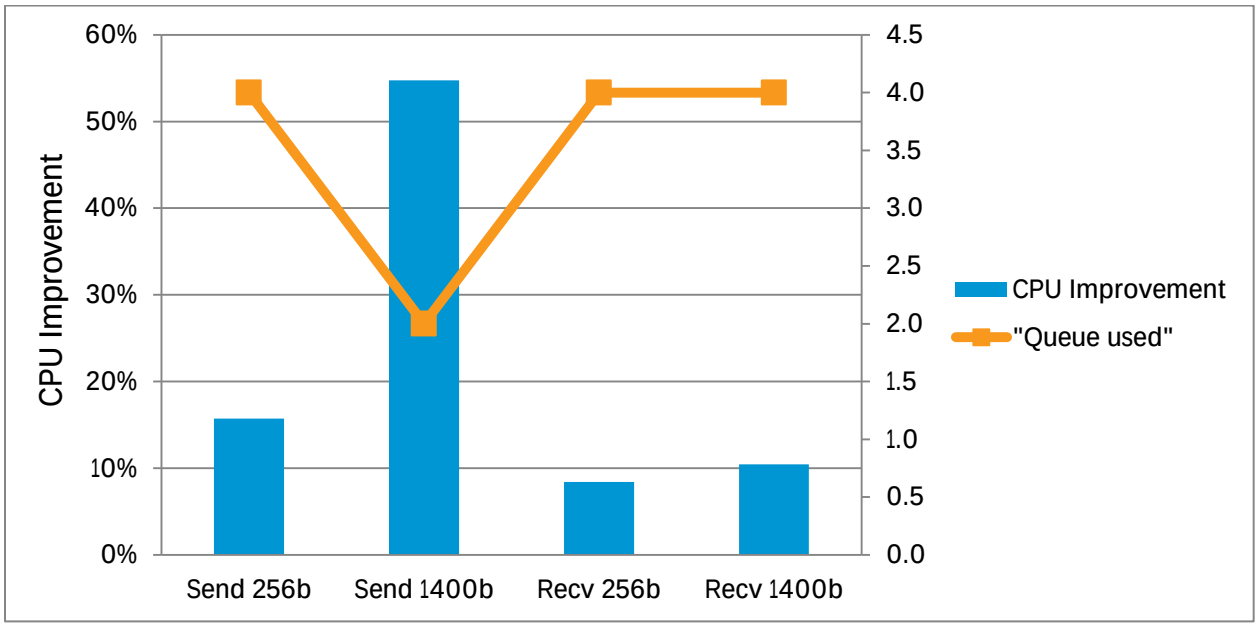


Figure 9. CPU Improvement for 16 VMs with Dynamic NetQueue

As seen in Figure 9, Dynamic NetQueue reduces CPU usage by about 10% to 50%; the maximum benefit observed is for the large packet test case. Also, the gains are directly proportional to the reduction in the queue usage. In the default case, the driver uses all 8 hardware queues, but with Dynamic Netqueue the driver can pack all the virtual machines on either 2 or 4 queues depending upon the test case.

### Native Device Driver

vSphere 5.5 added support for a native device driver. Prior to vSphere 5.5, the device drivers supported were Linux-based (called vmklinux drivers) with an open source API interacting between the driver and the platform. This API had some disadvantages because vSphere had to convert various data structures before it could talk to the device driver. With vSphere 5.5, a native driver model was introduced and Emulex NICs became the first NIC to be supported with this API. This transition translated to CPU savings.

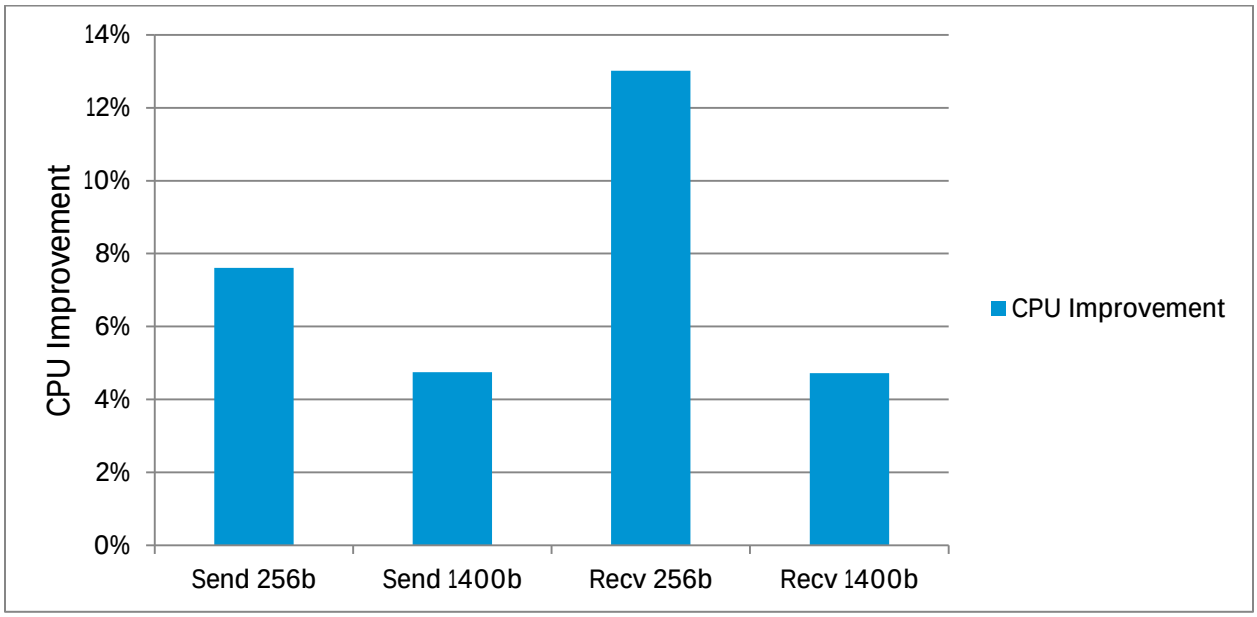


Figure 10. Throughput vs. CPU efficiency for native 10GbE Emulex driver (higher is better)

In Figure 10, an experiment was conducted with the native 10GbE driver. Using this native driver saved about 5%-10% of CPU for most cases, while delivering slightly higher throughput. Since the cost of using the open source API is per packet, the highest gains are for test cases with a high packet rate. Both small packet test cases show an improvement of about 10%, while for the large packet test cases, the improvement is about 4%.

# Conclusion

The physical NIC capabilities can cause a big impact on performance especially for highly networking intensive workloads. The difference is even higher for a virtualized network. Features like TSO offloads and Receive Checksum are important for the VXLAN environment to achieve line rate performance. Please refer to the relevant ESXi and driver documentation on how to configure these features.

# References

[1] VMware, Inc. Rishi Meta. (2013, January) VXLAN Performance Evaluation on VMware vSphere 5.1.
http://www.vmware.com/files/pdf/techpaper/VMware-vSphere-VXLAN-Perf.pdf

[2] VMware, Inc. (2014, May) Performance Best Practices for VMware vSphere 5.5.
http://www.vmware.com/pdf/Perf_Best_Practices_vSphere5.5.pdf

[3] VMware, Inc. (2012, September) Performance Best Practices for vSphere 5.1.
http://www.vmware.com/pdf/Perf_Best_Practices_vSphere5.1.pdf

[4] VMware, Inc. (2009) Performance Evaluation of VMXNET3 Virtual Network Device.
http://www.vmware.com/pdf/vsp_4_vmxnet3_perf.pdf

## About the Authors

Rishi Mehta is a Staff Engineer in the Performance Engineering group at VMware. His work focuses on improving network performance of VMware's virtualization products. He has a Master's in Computer Science from North Carolina State University.

## Acknowledgements

The author would like to thank Julie Brodeur, Amitabha Banerjee, Chien-Chia Chen, Sai Chaitanya, Subbarao Narahari, Shoby Cherian, Lenin Singaravelu, Shilpi Agarwal, Zongyun Lai, and Suds Jain for reviews and contributions to the paper.

**VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com**
**Copyright © 2015 VMware, Inc. All rights reserved. This product is protected by U.S. and international copyright and intellectual property laws. VMware products are covered by one or more patents listed at http://www.vmware.com/go/patents. VMware is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions. All other marks and names mentioned herein may be trademarks of their respective companies. Item: EN-001694-00 Comments on this document: http://communities.vmware.com/docs/DOC-28528**