VXLAN Performance Evaluation on VMware vSphere® 5.1

Performance Study

TECHNICAL WHITEPAPER

vmware®

Table of Contents

Introduction

Cloud computing services have experienced rapid growth over the past few years because they can keep costs down by allowing multiple tenants to share system resources. One requirement of making this multiple tenancy possible is to provide each tenant with network isolation. Segmenting the traffic using VLAN is a typical solution to this problem. However, service providers also need to keep up with customer demand by being able to move workloads to those servers that have spare resources. To do so, network traffic needs to be encapsulated so that the workload is not tied to the underlying hardware. This can be a problem because the networking architecture ties the workloads to the underlying hardware, which restricts the movements of these workloads and limits where these workloads can be placed. In addition, segmenting the physical LAN using VLANs does not scale beyond a certain limit.

Virtual extensible LAN (VXLAN) is a network encapsulation mechanism that enables virtual machines to be deployed on any physical host, regardless of the host's network configuration. It solves the problems of mobility and scalability in two ways:

- It uses MAC in UDP encapsulation, which allows the virtual machine to communicate using an overlay network that spans across multiple physical networks. It decouples the virtual machine from the underlying network thereby allowing the virtual machine to move across the network without reconfiguring the network.
- VXLAN uses a 24-bit identifier, which means that a single network can support up to 16 million LAN segments. This number is much higher than the 4,094 (limit imposed by the IEEE 802.1Q VLAN specification.

Since VXLAN is an additional encapsulation mechanism introduced at the hypervisor layer, there are certain performance implications. This paper demonstrates that the performance of VXLAN on vSphere 5.1 is very close to a configuration without VXLAN, and vSphere 5.1 with VXLAN configured can meet the demands of today's network-intensive applications. We used industry-standard benchmarks to conduct our experiments that demonstrate:

- A virtual machine configured with VXLAN achieved similar networking performance to a virtual machine without VXLAN configured, both in terms of throughput and CPU cost.
- vSphere 5.1 scales well as we add more virtual machines on the VXLAN network.

VXLAN Performance Considerations

VXLAN introduces an additional layer of packet processing at the hypervisor level. For each packet on the VXLAN network, the hypervisor needs to add protocol headers on the sender side (encapsulation) and remove these headers (decapsulation) on the receiver side. This causes the CPU additional work for each packet.

Apart from this CPU overhead, some of the offload capabilities of the NIC cannot be used because the inner packet is no longer accessible. The physical NIC hardware offload capabilities (for example, checksum offloading and TCP segmentation offload (TSO)) have been designed for standard (non-encapsulated) packet headers, and some of these capabilities cannot be used for encapsulated packets. In such a case, a VXLAN enabled packet will require CPU resources to perform a task that otherwise would have been done more efficiently by physical NIC hardware. There are certain NIC offload capabilities that can be used with VXLAN, but they depend on the physical NIC and the driver being used. As a result, the performance may vary based on the hardware used when VXLAN is configured.

Test Configuration

The test configuration (shown in Table 1 and Figure 1) consisted of two identical HP ProLiant DL380 G7 machines running vSphere 5.1. Both systems ran the same number of virtual machines for the tests. The physical NIC we chose assists with Tx CKO/TSO of encapsulated packets.

	SERVER	CLIENT
Make	HP ProLiant DL380 G7	HP ProLiant DL380 G7
CPU	2X Intel Xeon X5690 @ 3.47GHz, 6 cores per socket	2X Intel Xeon X5690 @ 3.47GHz, 6 cores per socket
RAM	64GB	64GB
NICs	Intel Corporation 82599EB 10-Gigabit SFI/SFP+ Network Connection	Intel Corporation 82599EB 10-Gigabit SFI/SFP+ Network Connection

Table 1. Test configuration for vSphere host and client machine

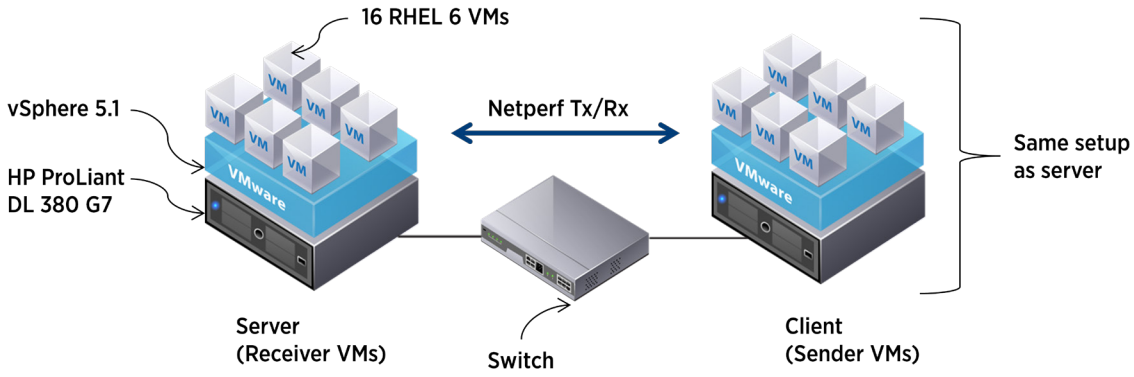

Figure 1. Test-bed setup

We used netperf to generate TCP traffic between the vSphere host virtual machines and the client virtual machines. Netperf is a light user-level process that is widely used for networking measurements. It consists of two binaries:

- netperf – a user-level process that connects to the server and generates traffic
- netserver – a user-level process that listens and accepts connection requests

We installed each virtual machine with 64-bit Red Hat Enterprise Linux 6 and configured each with one virtual CPU and 2GB RAM. Each virtual machine was configured to use vmxnet3 (driver version 1.0.29.0-NAPI) as the virtual NIC. We configured the same number of sender and receiver virtual machines for each test. Each virtual machine was sending to its corresponding receiver virtual machine. Since a single netperf session was not enough for achieving line rate at 10Gbps, we ran five sessions per virtual machine. To maintain consistency, we used the same number of sessions per virtual machine as we added more virtual machines to our test.

We measured the performance impact of VXLAN for both large and small messages being sent from 1 to 16 virtual machines. For each configuration, we ran the tests with and without VXLAN configured. We then measured the CPU utilization of the vSphere 5.1 host and the combined throughput for all netperf sessions. To measure small packet size performance, we used TCP_NODELAY as a socket option. TCP_NODELAY allows the sender to send packets immediately, and the TCP/IP stack of the guest operating system cannot aggregate small packets into large packets. As a result, we get packet sizes which are very close to the message size.

Since VXLAN adds data to the standard MTU frames, we increased the MTU size for the physical NICs to 1600. We used INTEL NICs because they can be used to do checksum offloads for encapsulated packets. We also enabled RSS for the Intel physical NIC so that traffic could be distributed among various queues for VXLAN traffic. RSS is only present for Intel 10Gb adapters on vSphere 5.1, and can be enabled by unloading and loading the module with vmkload_mod ixgbe RSS="4" for each 10Gb Intel adapter on the server. For the default configuration we used the standard MTU and default ixgbe driver (no RSS configured).

Results

The test results demonstrate the following:

- Network throughput of a virtual machine configured with VXLAN is very close to that of a default (no VXLAN) configuration.
- For small size packets, throughput scales linearly as more virtual machines are added.
- The CPU overhead for VXLAN depends upon the packet size. CPU overhead refers to the extra CPU utilized by VXLAN when compared to the system without VXLAN configured. For the large message test case, there is almost no CPU overhead. For the small message test cases, the overhead is about 5% of the overall system CPU utilization.

Throughput for Large and Small Message Sizes

We used a 64K socket size and a 16K message size to evaluate large packet size performance. For the small message size we used an 8K socket size and a 128 byte message size. We then compared the performance with the default configuration of no VXLAN.

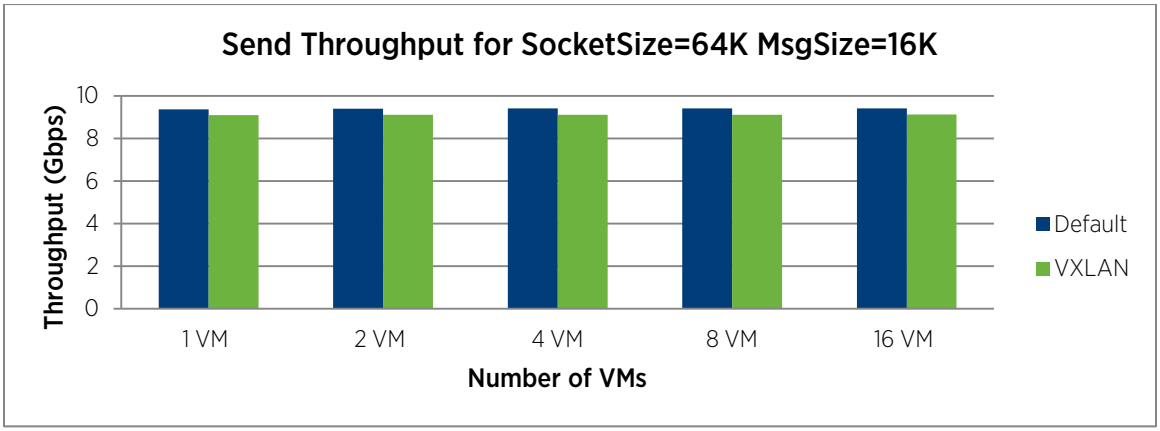

Figure 2. Multi-VM large message size throughput

Figure 2 shows that VXLAN can reach line rate for all configurations with a large socket size–message size configuration. The networking throughput reported by VXLAN is slightly lower than the default because of additional protocol overhead.

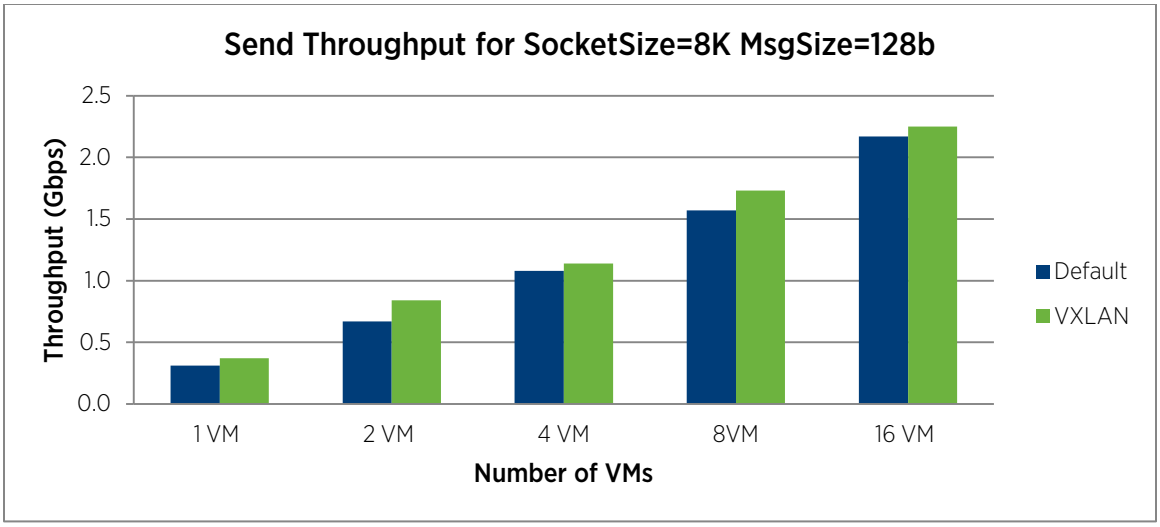

Figure 3. Multi-VM small message size throughput

For the small message size (Figure 3) we observed that throughput with VXLAN was better than or comparable to the default for all test case scenarios. Since VXLAN was configured to use RSS, each virtual machine, when configured with VXLAN, can use all hardware queues present. The default configuration, on the other hand, allows only one queue per virtual machine. As a result, VXLAN throughput was slightly higher.

CPU Overhead for Large and Small Message Sizes

For the same tests that measured throughput, we also collected CPU utilization on the vSphere 5.1 host. We then measured the additional CPU resources utilized when VXLAN was configured and reported it as the CPU overhead of using VXLAN. Since the throughput between VXLAN and default test cases was different, we compared the percentage of additional CPU used on a single core for every 1Gbps of traffic and reported the difference as the CPU overhead of using VXLAN.

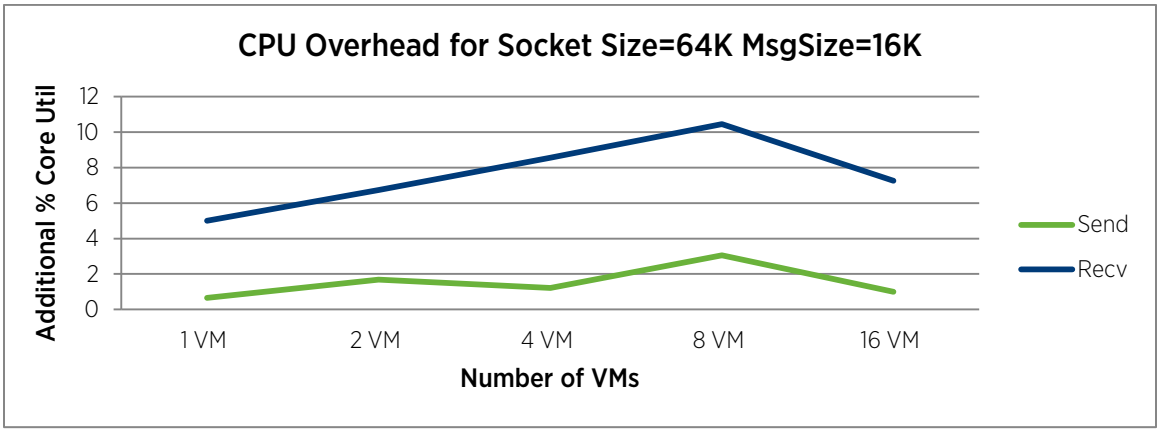

Figure 4. Multi-VM large message CPU overhead on a single core

For the large message transmit case (Figure 4), we saw that CPU overhead for VXLAN was negligible for almost all test cases. The maximum overhead we observed was close to 2% of a core for 1Gbps of traffic, and, as we added more virtual machines, the system overhead of using VXLAN remained constant. For a 12 core system, the overhead is less than 0.2% of the overall system utilization.

The CPU overhead for receiving large messages is higher for VXLAN because checksum offload capabilities of the NIC cannot be used. The overhead is highest for the 8VM test case (close to 10% of a single core), and most of the overhead is due to checksum computation being done in the software.

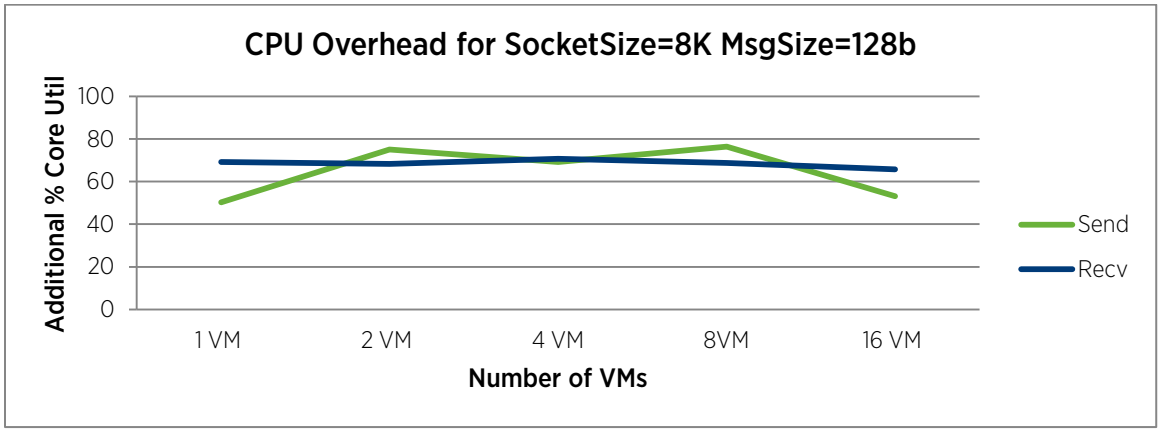

Figure 5. Multi-VM small message size CPU overhead on a single core

For the small message size (Figure 5), we saw that the CPU overhead for VXLAN was about 60% of a single core for both send and receive test cases. For a 12 core system, the overhead is approximately 5% of the overall system utilization. The overhead also remained constant as we added more virtual machines. The overhead is higher for the small message case because for every 1Gbps of traffic, vSphere 5.1 has to process higher packet rates and each packet requires encapsulation/decapsulation to be done in software.

Throughput and CPU Utilization for 16 Virtual Machines, Various Message Sizes

To measure the impact of variable packet size with VXLAN, we configured 16 virtual machines to use different socket and message sizes with netperf. We then measured the throughput and CPU utilization for the vSphere 5.1 host and compared it against the default configuration.

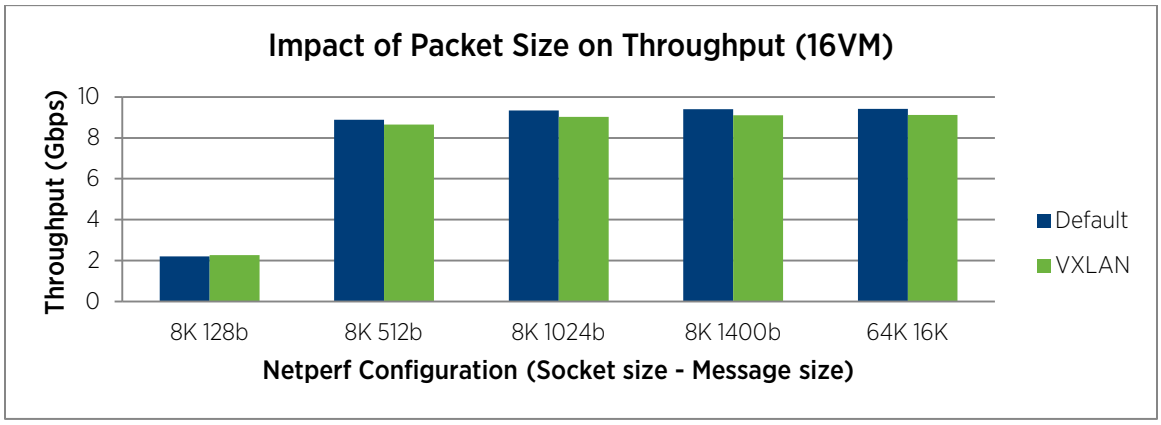

Figure 6. 16-VM various message size throughput

For the various message size throughput scenarios for 16 virtual machines (Figure 6), we noticed that throughput was comparable for all test cases. We were able to achieve line rate with a 1024 byte message size for both VXLAN and the default configuration.

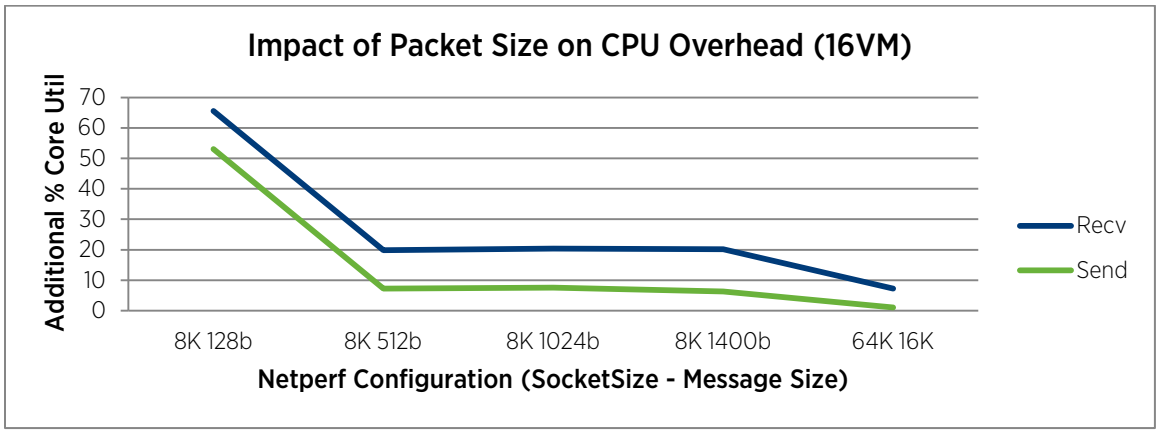

Figure 7. 16-VM CPU overhead on a single core with VXLAN

For the 16-VM test cases with varying message sizes (Figure 7), we observed that the CPU overhead was highest for the128-byte message cases (at 60% of a single core). It decreased substantially as we increased the message size and was less than 10% for the large message cases. The overall system overhead of using VXLAN for a 12 core system decreased from 5% to less than 1% as we increased the message size from 128 byte to 16K.

Conclusion

This paper evaluated VXLAN performance to demonstrate that vSphere 5.1 is readily able to host network-intensive applications on VXLAN with good performance and scalability. We compared our performance to standard virtual switch configurations and observed that throughput with VXLAN configured is comparable to throughput without VXLAN. For a small message size, throughput scaled linearly as we added more virtual machines. For large message sizes, we were able to achieve line rate for all configurations. There is some CPU overhead due to the use of VXLAN. This overhead varies based on the configuration and is due to lack of hardware offloads for encapsulated packets. After NIC vendors start supporting these offloads, this overhead may reduce significantly.

About the Author

Rishi Mehta is a Senior Member of Technical Staff 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 Jianjun Shen, Julie Brodeur, Lenin Singaravelu, Shilpi Agarwal, Sreekanth Setty, and Suds Jain (in alphabetical order) for reviews and contributions to the paper.

vmware

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com