Network I/O Latency on VMware vSphere® 5

Performance Study

TECHNICAL WHITE PAPER

Table of Contents

Executive Summary 3

Introduction....3

Test Setup 4

Other Tunings....5

Workloads....5

Performance Results.... 6

Virtual Machine vs. Physical Machine Round-Trip Latency.... 6

Round Trip Latency Between Two Virtual Machines on the Same vSphere Host 8

Virtual Machine Round-Trip Latency with DirectPath I/O....8

Impact of Resource Contention on Virtual Machine Response Time 9

Impact of CPU Contention 9

Impact of Network Bandwidth Contention....12

Performance Results Summary and Best Practices....13

Conclusion 14

References 14

Executive Summary

High throughput applications are very commonly run on the VMware vSphere platform. Previous VMware performance studies have shown that the combined throughput of multiple virtual machines running on vSphere can easily go beyond 30Gbps[1, 2]. This paper, by characterizing the latency overhead of vSphere under various scenarios, proves that vSphere can readily handle applications that are sensitive to network I/O latency with very little performance overhead. Scenarios explored in this paper model real-world performance issues, including CPU contention and network bandwidth contention. Our results show that in most cases, vSphere latency overhead is only a few tens of microseconds and rises to only a few hundreds of microseconds in heavily over-committed cases. This paper also presents guidelines for running latency-sensitive applications on vSphere 5.0.

Introduction

Network I/O latency is an important measurement, especially in latency-sensitive applications like those found in industries such as finance and trading, healthcare, transportation, and telecom (in particular, applications using Voice over IP). The end-to-end latency requirements of these applications vary from microseconds to milliseconds. And some of these applications, such as those utilizing Voice over IP, are more sensitive to jitter than end-to-end latency. So an important measurement to consider for vSphere performance in this scope is latency and jitter added due to virtualization.

Packets sent or received from a virtual machine experience some increase in network I/O latency compared to a bare metal (native) environment. This is because packets need to traverse through an extra layer of virtualization stack in the case of virtual machines. The additional latency due to the virtualization stack can come from several different sources. The most commons ones include

- **Emulation Overhead:** Certain privileged instructions and some operations to access I/O devices that are executed by the virtual machine are intercepted by the hypervisor. This adds some overhead and contributes to network I/O latency as well.
- **Packet Processing:** The network virtualization stack forwards a network packet from the physical NIC to the virtual machine and vice versa. This requires some computation and processing such as switching decisions at the virtual switch, inserting and stripping the VLAN tag, and packet copy if necessary. This processing adds some latency on both transmit and receive paths.
- **Scheduling:** Packet transmission and reception involves multiple hypervisor threads and virtual CPUs. The vSphere scheduler has to schedule and then execute these threads (if they are not already running) upon receipt and transmission of a packet. On an idle system this takes a couple of microseconds but on a busy system, if there is high CPU contention, this may take tens of microseconds and occasionally milliseconds.
- **Virtual Interrupt Coalescing:** Similar to physical network interface card (NIC) interrupt coalescing, vSphere does virtual interrupt coalescing. That is, a virtual machine may not be interrupted immediately after receiving or transmitting a packet. Instead, vSphere may wait to receive or transmit more than one packet before an interrupt is posted. Also, on the transmit path, the virtual machine may wait until a few packets are queued up before sending the packets down to the hypervisor. The transmit queue depth, receive interrupt queue depth, and transmit completion interrupt queue depth is determined dynamically based on the workload. The maximum wait time is bounded and the current default bound is 4 milliseconds. Since vSphere adjusts the previously mentioned depths automatically based on workload, most packets will not be delayed that long. However, interrupt coalescing can add some jitter, and in some cases, may have a noticeable impact on average latency.
- **Network Bandwidth Contention:** If a latency sensitive virtual NIC (vNIC) is put on a physical NIC that is shared with other vNICs, there will be contention for the network bandwidth. The packets for the latency-sensitive vNIC may be delayed because of head-of-line blocking due to traffic from other vNICs. The delay will depend on the number of packets that are queued and hence on each vNIC's traffic. Some physical NICs (typically 10Gbps NICs) support multiple network queues (NetQueue) and each vNIC gets its own queue as long as the number of queues is more than the number of vNICs sharing the physical NIC. But if the number

of vNICs is more than the number of queues, multiple vNICs may share a queue and hence their I/O latency may be affected by each other's traffic.

- **Halt-Wakeup Physical CPU:** Modern hardware and operating systems are power efficient. That is, operating systems put the physical CPU in sleep mode (halt) when the CPU is idle. Waking up the CPU from halt mode adds some latency and it varies from CPU to CPU. Depending on the load of the system, the CPU may go into deeper sleep states from which it takes even longer to wake up. This overhead is generally common between non-virtualized and virtualized platforms. However, sometimes the side effect of this can be worse in virtualized environments than physical. For example, if CPU flushes its cache on halt, it may impact a virtualized system more than a physical system because the cache footprint is larger in a virtual environment.
- **Halt-Wakeup Virtual CPU:** When the guest operating system issues a halt instruction, the vSphere scheduler may de-schedule the vCPU. And rescheduling the vCPU—putting it back in the run queue—can take a couple of microseconds. Latency introduced due to virtual CPU halt-wakeup or physical CPU halt-wakeup is more prominent when the CPU is lightly loaded; that is, when the CPU halts frequently.

We designed and ran several performance experiments to show under what conditions the different sources of latency become dominant and how they affect end-to-end latency. The results will help you make proper design choices for your environment such that it fulfills your end-to-end latency requirements.

Test Setup

Figure 1 describes the test setup. We had two physical machines: one running vSphere and the other running a bare metal Linux operating system. Both machines had a 10Gbps physical NIC installed and the NICs were connected through a 10Gbps switch. The vSphere machine had two 10Gbps NICs: one for the latency-sensitive virtual machine traffic and the second for the traffic of other virtual machines. Client1 ran latency tests against the latency-sensitive VM, and client2 generated traffic for other background VMs to keep them busy. The specifications of the machines follow.

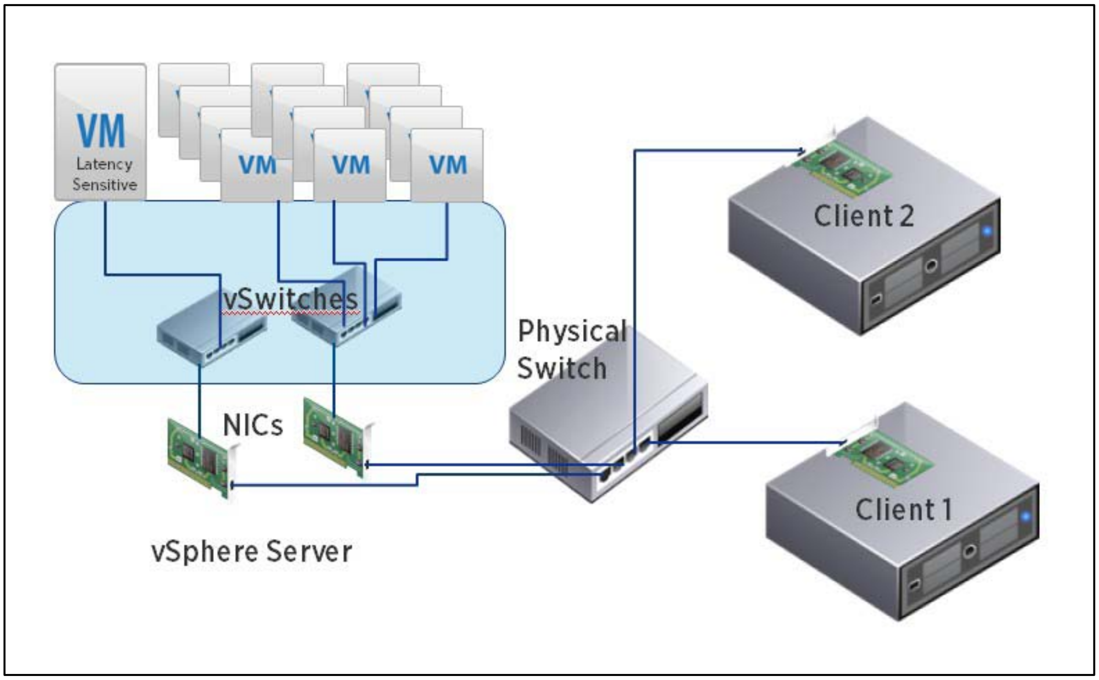

Figure 1. Test bed setup

Table 1 details the hardware setup for the vSphere host and two client machines.

	VSPHERE SERVER	CLIENTS
Make	HP DL 380 G6	IBM x3650
CPU	2-Socket Quad-Core Intel Xeon X5570 @ 2.93GHz	2-Socket Quad-Core Intel Xeon E5355 @ 2.66GHz
Hyper-threading	Enabled	Enabled
RAM	80GB	16GB
NICs	2X Intel 82598EB 10Gbps Adapter	2X Intel 82598EB 10Gbps Adapter
OS	vSphere 5.0	SLES 11 SP1

Table 1. Hardware setup details

Table 2 shows the configuration of the latency-sensitive virtual machine.

OS	RHEL6 SP1
RAM	2GB
vNIC	VMXNET3
vCPU	2

Table 2. Latency measurement VM configuration

Other Tunings

- We disabled physical NIC interrupt moderation on both the client and the server. We chose to disable it because we were interested in understanding the latency added by the hypervisor. Interrupt moderation may work differently for different NICs, and we wanted to isolate latency added due to that.
- We also disabled power management in the BIOS (including C-states and QPI link power management) for isolating latency added due to power management.

Workloads

We used ping and netperf-2.4.5[3] benchmarks for our experiments. We chose micro-benchmarks because we were more interested in measuring the latency overhead added by vSphere instead of measuring the end-to-end latency for different workloads. The details of these benchmarks are as follows:

- **Ping request-response:** In this test, an ICMP ECHO request is sent from a client to the server at regular intervals and the server sends an ECHO reply back. The client records the round-trip time (RTT) for each request. We set the interval to 1 millisecond and ran with one and ten sessions to get packet rates of 1K (ping.1) and 10K (ping.10) respectively.
- **Netperf request-response:** Netperf is a simple client-server based tool. We chose a request response (TCP_RR) test of this benchmark. In this test, the client sends a TCP request to the server and the server replies back. The client records the response time for each request. Netperf provides a burst size parameter to control the number of outstanding requests. We used burst sizes of 2 and 4 in our experiments (netperf.2 and netperf.4). We also set the TCP_NODELAY flag and enabled the DIRTY_BUFFER option. With the

DIRTY_BUFFER option enabled, netperf touches the data before sending and after receiving it. The packet rate for the netperf test was typically in the range of 40K to 80K packets per second (PPS) and depended on the latency.

There are a wide range of latency-sensitive applications with varying latency and jitter requirements including:

1. Set targets for end-to-end latency
2. Set targets for percentage of requests that experience latency below a certain threshold
3. Set targets for the difference between mean and 99th-percentile end-to-end latency

We used performance metrics to cover all of the above dimensions of measurements: mean round-trip time, max, percentage of spikes above 1 millisecond (%Spikes1ms), and percentage of spikes above 5 milliseconds (%Spikes5ms) and 99th-percentile.

The test duration was 5 minutes. We ran three iterations and took the medians of three iterations. The median run was chosen based on the mean round-trip time and all other metrics were reported for that particular run. The message size for each test was set to 256 bytes unless otherwise mentioned.

Performance Results

Virtual Machine vs. Physical Machine Round-Trip Latency

First, we compare the round-trip latency of virtual machine (VM) to physical machine (Native) communication (VM-Native) with physical machine to physical machine (Native-Native) communication to measure the round-trip latency overhead per VM. We set up the server machine in dual boot mode so that we could use the same machine for vSphere and Native. We first booted this machine into ESXi and collected VM-Native latencies. Only the latency sensitive VM was running on vSphere and the system was completely idle except for the CPU used for the latency measurement test itself. Then we booted the machine into Native and collected Native-Native latencies. When the server was booted into Native, the number of physical CPUs and memory was set to the same as the virtual machine's configuration to get a fair comparison between the two.

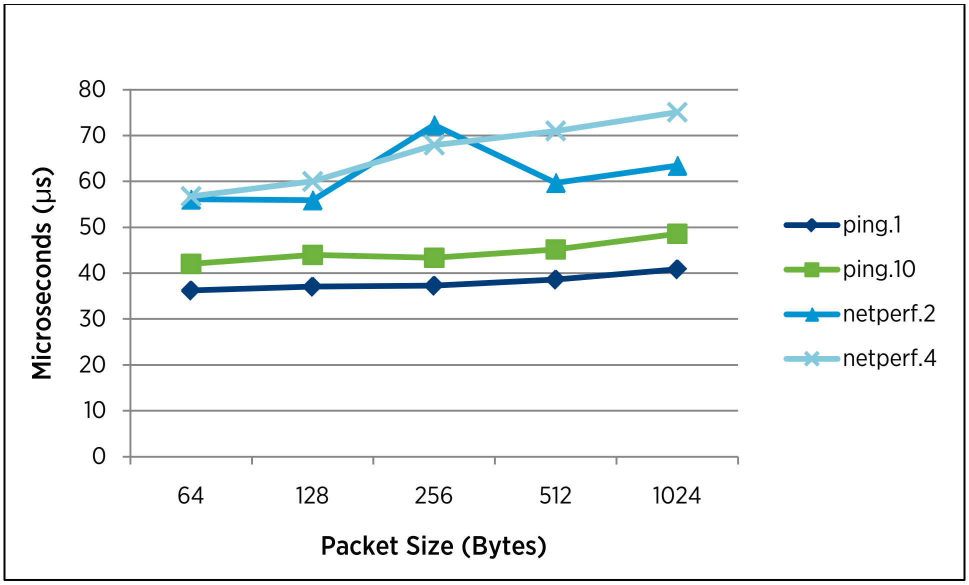

Figure 2. Mean round-trip latency VM-Native

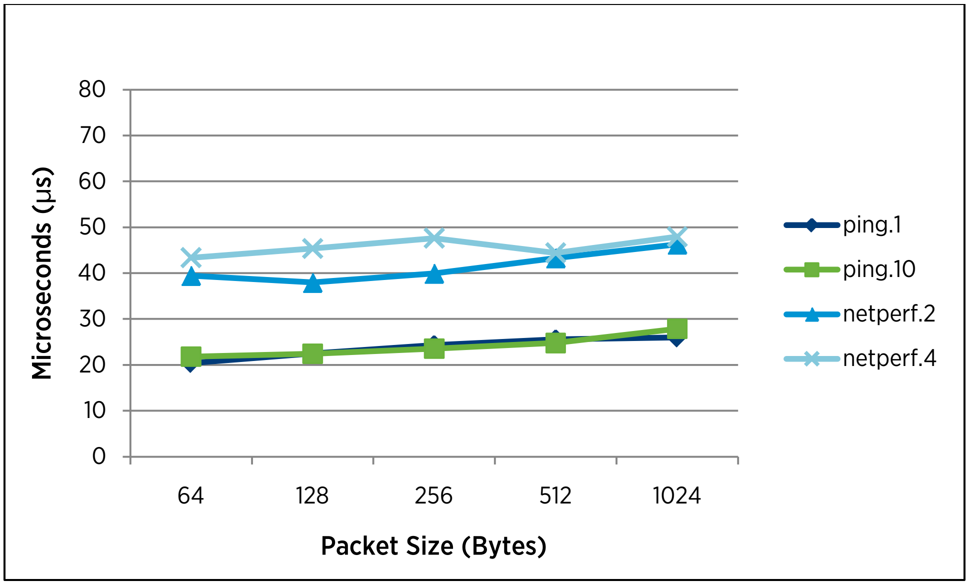

Figure 3. Mean round-trip latency Native-Native

As described in "Workloads," different workloads were run: ping.1, ping10, netperf.2, and netperf.4. Packet size for each workload was varied from 64 bytes to 1K bytes. Packet rate was in the range of 1 – 80K packets per second. Figures 2 and 3 show the mean round-trip latency for VM-Native and Native-Native communication respectively. VM-Native latency is in the range of 35-75 microseconds and Native-Native latency is in the range of 20-50 microseconds. In most cases, VM-Native latency is 15-20 microseconds higher than Native-Native latency. There are a few cases in which the gap goes up to 30 microseconds. The cases in which gap is greater than 20 microseconds are the ones in which either coalescing depth was not the optimal or the CPU utilization was so high that it started affecting latency. A similar difference was observed in the 99th-percentile. Figures 2 and 3 also show that the latency for the netperf test is higher than that of the ping test. This is because:

- Netperf sends and receives TCP packets that traverse through the TCP stack all the way up to the application level, whereas ICMP packets in the ping test are serviced by the IP stack in the kernel and packets do not go to the application level.
- The packet rate for the netperf test is a lot higher than that of the ping test (greater than two times), resulting in higher CPU utilization. Since the system is busier, latency also increases.

In addition, we found that in all of these tests for both VM and Native:

- The 99th-percentile was always within two times of the mean
- % 1ms spikes were less than or equal to .009
- % 5ms spikes were less than or equal to .001
- Max response time was in tens of milliseconds

These findings prove that jitter—that is, the variance in response time—is very low for both virtual and native machines.

Round Trip Latency Between Two Virtual Machines on the Same vSphere Host

In the previous tests, we looked at virtual machine round trip time when it was communicating with a native client over a 10Gbps NIC. However, virtual machines that are on the same ESXi host sharing the same virtual switch can talk to each other without going through a physical NIC. We measured VM-VM latency by powering on two RHEL6 VMs and running latency tests between them. We disabled virtual interrupt coalescing in this case so that we could do a comparison with VM-Native latency that was measured with physical NIC interrupt coalescing disabled.

Results are shown in Figure 4. VM-VM latency is in the range of 25 to 70 microseconds for different workloads. These results are slightly better than the latency observed for a VM-Native communication over 10Gbps link (Figure 2). It is higher than Native-Native latency because even though the small wire latency in Native-Native is replaced by memory-memory copy latency between two virtual machines, the overheads in virtualization mentioned in the "Introduction" section are still an additional source of latency.

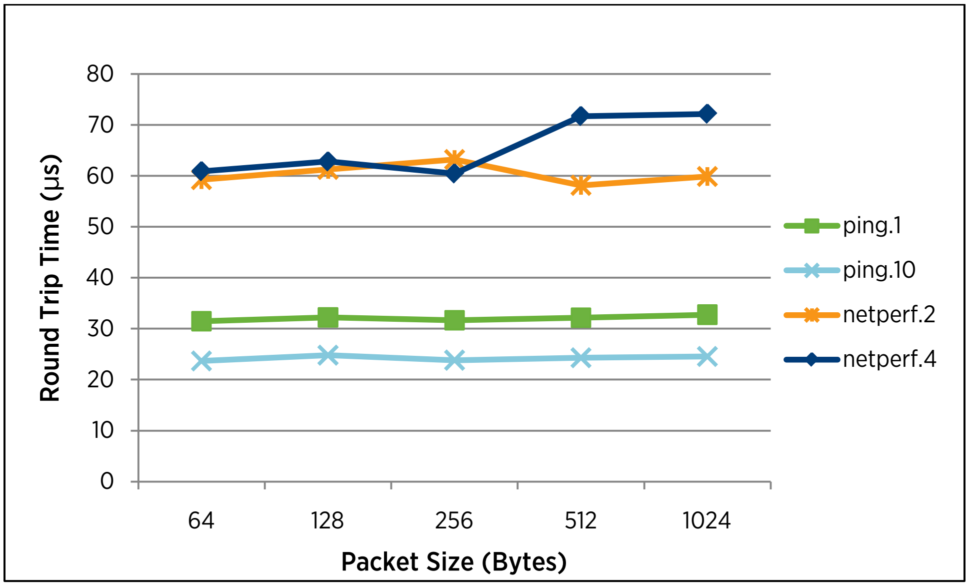

Figure 4. Mean round-trip latency VM-VM

Virtual Machine Round-Trip Latency with DirectPath I/O

A virtual machine may access the physical device in two modes: para-virtualization (emulation) and DirectPath I/O[4]. DirectPath I/O is a technology that has been available since vSphere 4.0. It allows the VM to access the physical device directly (without the intervention of the hypervisor in every I/O). Since the VM can access the physical device directly, latency is much lower in this mode. However, some vSphere features are not supported with this mode, so we recommend using it only when it is absolutely necessary to achieve the desired performance. In this experiment, we measure the latency of VM-Native communication when the VM is doing DirectPath I/O.

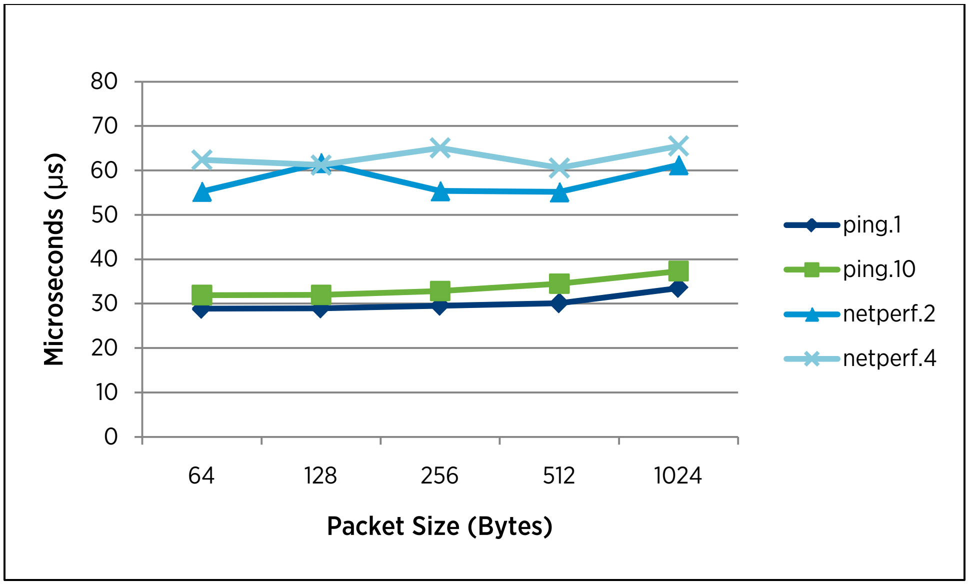

Figure 5. Mean round-trip latency between VM with DirectPath I/O and Native

Results show that for the DirectPath I/O VM, round trip time is in the range of 30-65 microseconds for different workloads (Figure 5). This is about 10 microseconds lower than that of a VM using para-virtualization (Figure 2). 10 microseconds can be a significant gain for extremely low latency applications such as stock trading applications, and we recommend using DirectPath I/O for such applications.

Impact of Resource Contention on Virtual Machine Response Time

Impact of CPU Contention

So far, all the measurements are with one or two VMs running on a vSphere host. However, typically there will be multiple VMs running on the server and contending for the CPU. It is important to understand how the network I/O latency increases as the CPU contention on the host increases.

We added CPU contention on the vSphere host by running multiple 1-vCPU VMs on the same vSphere host. The VMs were contending only for CPU with the test VM running ping or netperf for a latency measurement. These VMs, referred to as background VMs, were placed on a separate virtual switch and a separate physical NIC and were communicating with a separate client (client2). The workload in each background VM consisted of a multi-threaded server that received TCP requests from the client, carried out some CPU processing, and sent the response back to the client. The workload has several parameters—such as number of threads, packet rate per thread, and CPU computation per packet—which could be tuned to achieve a desired packet rate and CPU usage per VM.

We ran the workload in two configurations:

- Configuration 1 - Packet rate per VM 5K and CPU Usage per VM 20%.
- Configuration 2 - Packet rate per VM 8K and CPU Usage per VM 40%.

For the latency measurement, we used workloads ping.10 and netperf.4. We used a packet size of 256 bytes. Figures 6 and 7 show the increase in round-trip time as we increased the number of VMs on the server for configuration 1 and 2 respectively. We also plotted %CPUUtil, which is the average CPU utilization of the system for the ping.10 test. For the netperf.4 test, CPU utilization was slightly higher than that of the ping.10 test. Note that the number of VMs is for the VMs that are generating background load. There was an additional VM running on the server that we used for the latency measurement.

Figures 6 and 7 show:

- An increase in the round-trip latency compared to no load was within 100 microseconds as long as the average CPU utilization was less than 90%. This trend persisted even at 32 VMs; that is, a consolidation ratio of 4 (total vCPUs > 32 on 8 physical cores).

  *NOTE: One of the advantages of a virtualized system is its ability to fully make use of its physical resources by running multiple virtual machines on a single physical host. CPU usage should be as high as it can (but not beyond 100%) and not appreciably affect system performance. An indicator of this is a high **consolidation ratio**, which is the number of virtual CPUs divided by the number of physical CPUs and high CPU utilization.*

- The 99th-percentile response time was within two times of mean as long as the consolidation ratio was less than 1. But it started increasing exponentially once the average CPU utilization was more than 60% and the consolidation ratio was greater than 2.

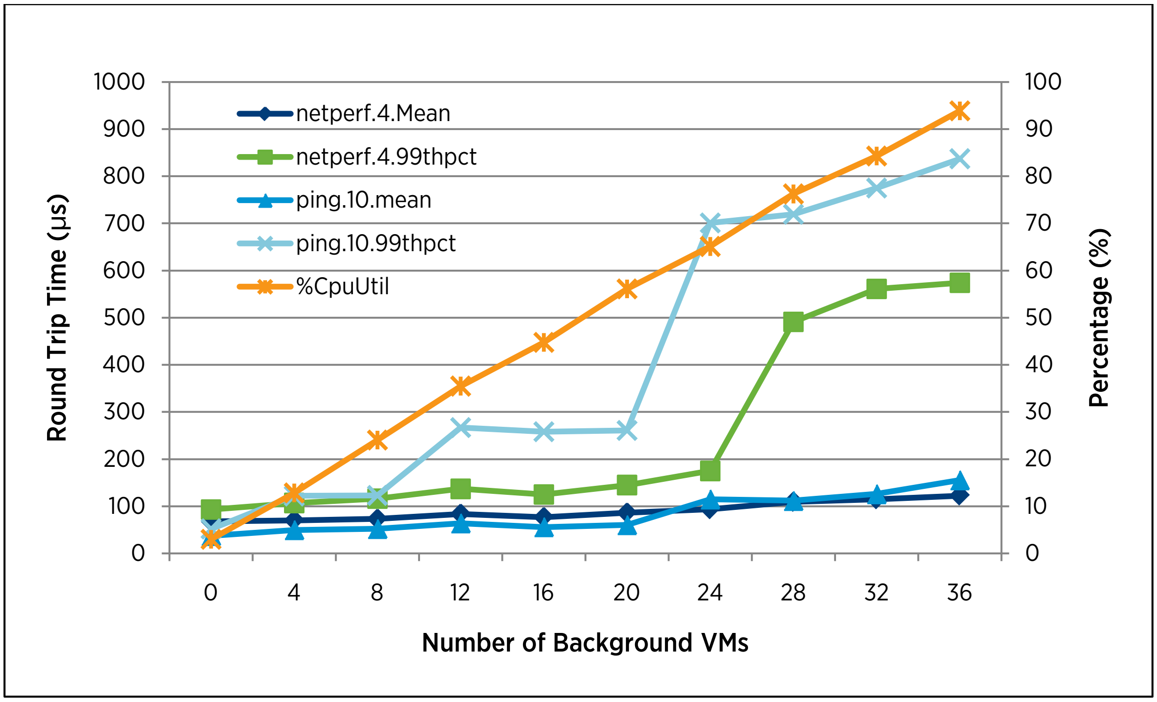

Figure 6. Round-trip time under CPU contention – Configuration 1

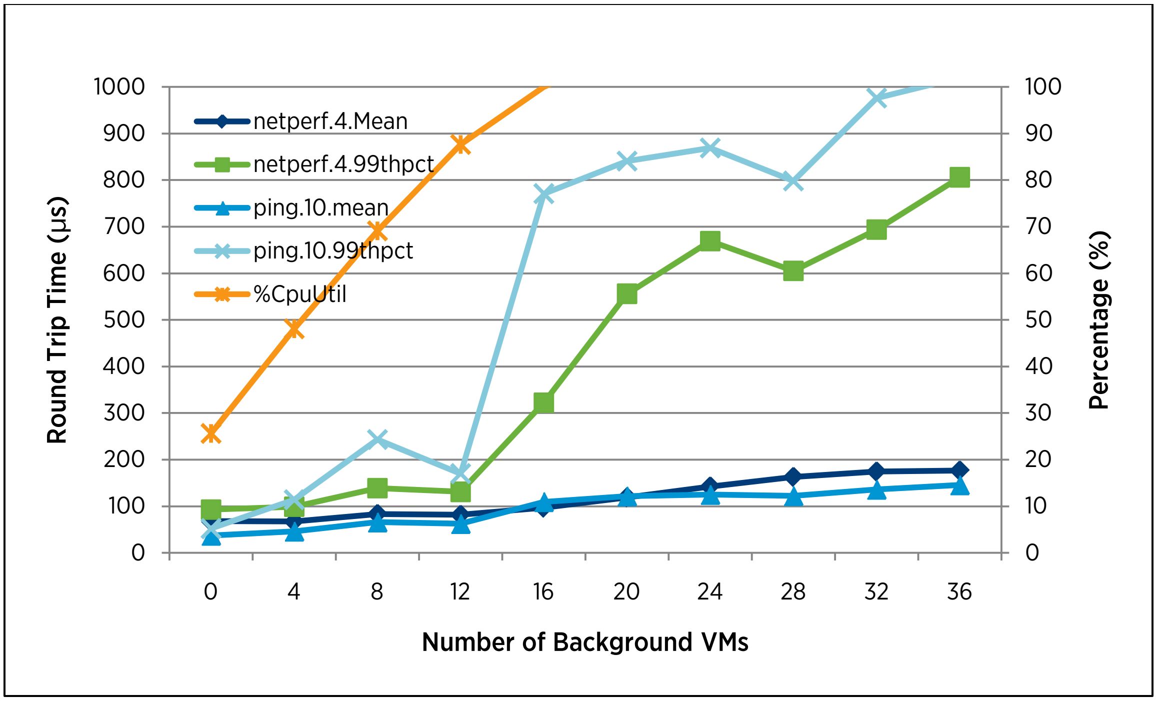

Figure 7. Round-trip time under CPU contention – Configuration 2

Figure 8 shows the percentage of 1 millisecond and 5 millisecond spikes for different tests with increasing load. The interesting observations from Figure 8 are:

- 5 millisecond spikes are negligible even at 36 VMs.
- 1 millisecond spikes become greater than 0.1% at 24 VMs (65% CPUUtil) for configuration 1 and 16 VMs (100% CPUUtil) for configuration 2. That is, if the system is not overcommitted, the number of 1 millisecond spikes is lower than 0.1% as long as the consolidation ratio is less than 3.

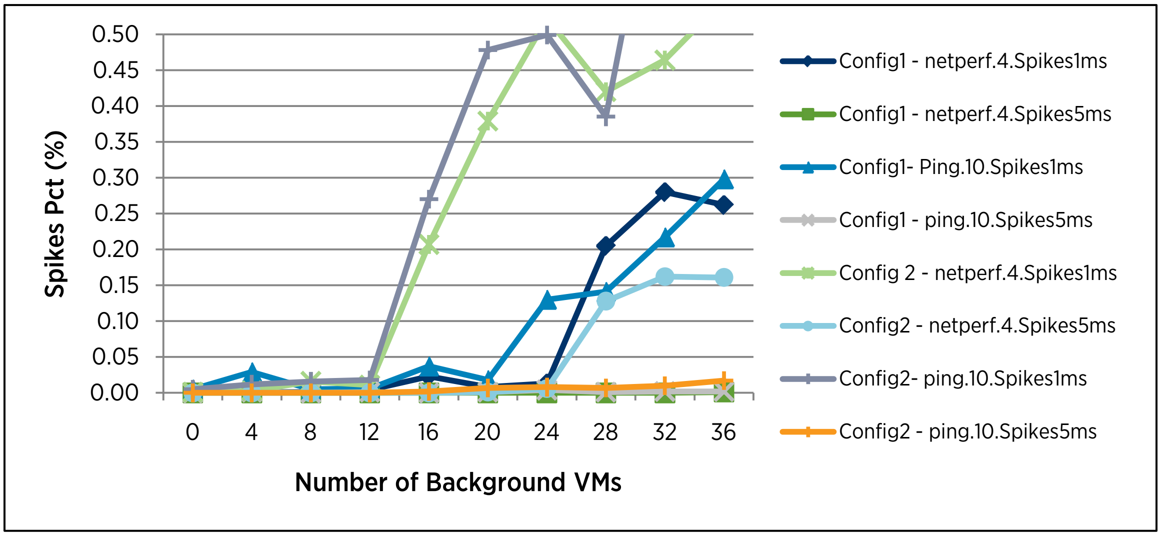

Figure 8. Spikes under CPU contention – Configurations 1 and 2

Impact of Network Bandwidth Contention

If multiple virtual NICs (vNICs) that belong to the same or different virtual machines are sharing a physical NIC, their traffic may interfere with each other. For example, if a latency sensitive, low-packet-rate vNIC is sharing a physical NIC with other vNICs that are transferring a large number of packets (send or receive), the software and hardware queuing delay of other packets will delay the packets of the latency-sensitive vNIC and vice-versa. The overhead will depend on the level of contention. In the following experiment, we powered on two VMs: VM1 was running latency tests against client1 and VM2 was receiving 256 byte UDP packets at a given rate from client2. vNICs of both VMs were connected to the same virtual switch and hence were sharing the same physical NIC. The UDP packet rate for VM2 was varied from 0 to 700K in increments of 50K. Unlike other tests, we enabled ESXi physical NIC interrupt throttling in this case since the packet rate was very high.

vSphere supports NetQueue—technology that significantly improves performance in 10 Gigabit Ethernet virtualized environments. NetQueue allows multiple transmit (Tx) and receive (Rx) queues on certain 10Gbps physical NICs. The number of queues that a NIC makes available depends on the NIC type and number of cores on the system. Based on the number of queues and number of vNICs sharing the physical NIC, a vNIC may or may not get its own queue. However, if the vNIC gets its own queue, the impact of traffic from the vNICs will be small. We first loaded the physical NIC device driver with a single queue to understand the impact of sharing a network queue and then with multiple queues to see how performance improves when a latency-sensitive vNIC gets its own queue.

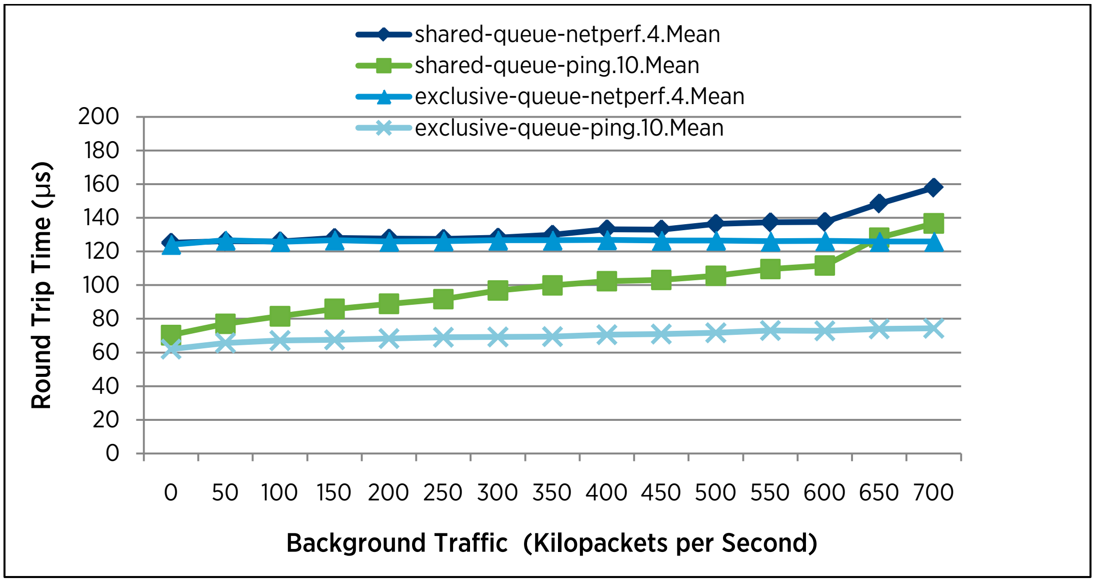

Figure 9. Round-trip time under Rx bandwidth contention

Figure 9 shows the increase in latency of VM1 as the traffic of VM2 is increased from 0 to 700K packets per second. We ran tests ping.10 and netperf.4. The results show that when the vNICs of both VMs are on different queues, there is hardly any increase in latency of VM1 as the traffic on VM2 is increased, but when they share the queue, the latency of VM1 increases by almost 5 microseconds for every 50 kilo packets per second (kpps) of traffic added to VM2. The impact on latency is lower for the netperf test because it has a high packet rate and multiple outstanding requests.

Performance Results Summary and Best Practices

We measured latency overhead added by virtualization under different load conditions. Results showed that:

- Out-of-the-box per-VM emulation overhead under no load was approximately 15-20 microseconds per round trip.
- For the workloads that we evaluated, the mean latency for a VM-Native communication was in the range of 35-75 microseconds. Using DirectPath I/O saved about 10 microseconds per VM per round trip.
- Under no competing load, round-trip latency for a VM-to-VM communication was slightly better than the latency for a VM-Native communication over a 10Gbps link.
- As the load increased on the system, overhead due to virtualization increased. The increase in the latency overhead per VM was less than 100 microseconds at 90% CPU utilization and a consolidation ratio of 4.
- 5-millisecond spikes were negligible as long as the system was not over-committed.
- 1-millisecond spikes were negligible up to a consolidation ratio of 3, as long as the system was not over-committed.
- The 99th-percentile response time was within two times of mean response time as long as the consolidation ratio was less than 1.

From the above-mentioned summary, we can infer that if the end-to-end latency is of the order of milliseconds or if the application is sensitive only to jitter, then high consolidation ratios can be achieved with very low (< 15%) virtualization overhead. A previously published whitepaper[6] also shows that VoIP applications can be virtualized successfully with consolidation ratios as high as 6 (48 virtual CPUs on 8 physical CPUs). However, if the end-to-end latency requirement is less than a few hundreds of microseconds, then it is best to use consolidation ratios of less than 1.

In addition, based on the experiments that we have done, we recommend the following best practices for virtualizing latency-sensitive applications:

- Consider disabling power management if the system is lightly loaded. This saves tens of microseconds in a round trip and makes a significant improvement in performance. Note that disabling power management has a side effect of increased power consumption. So, we recommend doing it only if it is absolutely necessary to achieve the desired performance.
- Consider disabling physical NIC interrupt coalescing if it helps improve performance. Note that this action may negatively impact performance if the packet rate for that physical NIC is high (> 20K). So, first check if disabling physical NIC interrupt coalescing helps improve performance and only then disable this feature.
- Consider disabling virtual NIC interrupt coalescing if it helps improve performance. Similar to physical NIC coalescing, disabling virtual NIC coalescing may negatively impact performance if the packet rate for that virtual NIC is high. So, first check if disabling virtual NIC interrupt coalescing helps improve performance and only then disable it.
- Contention on the physical NIC due to heavy traffic from other VMs might also affect network I/O latency. We saw an increase of approximately 5 microseconds for every 50K packets per second sent/received from the NIC. So, we recommend putting the latency-sensitive VM on a separate physical NIC. Another alternative is to use a 10Gbps NIC and make sure the number of vNICs sharing that NIC is less than the number of network queues on the NIC. Another alternative is to use NetIOC, but that will help only if the contention is on the transmit side.
- For more detailed best practices on latency-sensitive applications, please refer to *Best Practices for Performance Tuning of Latency-Sensitive Workloads in vSphere VMs*[5].

Conclusion

In this paper, we presented data for the round-trip latency overhead added by vSphere per VM under different scenarios. We found that on an idle system, round-trip latency overhead per VM is approx 15-20 microseconds. This number increases as the resource contention on vSphere increases, which is expected. Jitter was very low as long as the system was not overcommitted. In addition, we showed that using DirectPath I/O may save about 10 microseconds over emulation. We also provided guidelines and best practices for configuring vSphere based on the application latency requirements.

References

1. *VMware vSphere 4.1 Networking Performance*. VMware, Inc., 2011. http://www.vmware.com/files/pdf/techpaper/Performance-Networking-vSphere4-1-WP.pdf.
2. *10Gbps Networking Performance on ESX 3.5 Update 1*. VMware, Inc., 2008. http://www.vmware.com/resources/techresources/1078
3. Netperf manual. Hewlett-Packard Company, 1995. http://www.netperf.org/netperf/training/Netperf.html.
4. "Performance and Use Cases of VMware DirectPath I/O for Networking." *VMware VROOM! Blog*. VMware, Inc., 2010.http://blogs.vmware.com/performance/2010/12/performance-and-use-cases-of-vmware-directpath-io-for-networking.html.
5. *Best Practices for Performance Tuning of Latency-Sensitive Workloads in vSphere VMs.* VMware, Inc., 2011. http://www.vmware.com/resources/techresources/10220.
6. *Voice Over IP (VoIP) Performance Evaluation on VMware vSphere 5.0*. VMware, Inc., 2012. http://www.vmware.com/resources/techresources/10241.

.

About the Author

Shilpi Agarwal is a senior manager in the Performance Engineering group at VMware. Her work focuses on improving network performance of VMware's virtualization products. She has a master's degree in Electrical and Computer Engineering from the University of Wisconsin, Madison.

Acknowledgements

The author would like to thank Bhavesh Davda, Bhavjit Walha, Boon Seong Ang, Jin Heo, Julie Brodeur, and Lenin Singaravelu (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

Copyright © 2012 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-000786-00 Date: 8-Feb-12 Comments on this document: docfeedback@vmware.com