



# Testing Virtual Machine Performance with VMware vSphere 4 on 10 Gigabit Networks

Design Guide

# Contents

<b>Introduction.....</b>	<b>4</b>
<b>Preliminary Verification Steps .....</b>	<b>6</b>
VMware Performance Reporting Requirements.....	6
PCI Express Connectivity .....	6
BIOS Configuration .....	7
System Information .....	7
Minimum Hardware Requirements.....	8
Minimum Software Requirements.....	8
Measuring the Disk I/O Capabilities of the Server.....	8
<b>Network-Performance Concepts .....</b>	<b>9</b>
TCP Window-Size Tuning .....	9
Achievable Throughput Convergence Time.....	10
TCP Send and Receive Buffers and Socket Buffers .....	10
Jumbo Frames .....	12
Read and Write Block Size .....	13
Transmit Performance Compared to Receive Performance .....	13
Typical Offload Technologies.....	14
<b>VMware vSphere 4 Tuning .....</b>	<b>14</b>
Physical NICs (vmnics) Capabilities and Troubleshooting .....	15
Distinguishing VMware VMkernel Interface Performance from vNIC Performance.....	16
VMware VMkernel and Service Console TCP/IP Stack.....	16
VMXNET Generation 3 and TSO .....	16
VMware Tools .....	19
Intel VMDq and NetQueue .....	21
Intel VMDq .....	21
NetQueue .....	21
Complete Configuration Example for Intel 82598 with Intel VMDq.....	23
Removing the CPU Bottleneck.....	24
Large-Receive Offload .....	24
Increasing vCPU and Using RSS .....	24
Memory Management .....	25
Large Memory Pages .....	25
Removing the Memory Bottleneck.....	26
<b>Tuning the Guest OS .....</b>	<b>28</b>
Microsoft Windows 2008 .....	28
Tuning the TCP Stack .....	28
Socket Buffers .....	29
Microsoft Windows Firewall.....	30
Linux .....	31
Tuning the Client Side.....	32
<b>Measuring Network Performance in Virtual Machines .....</b>	<b>32</b>
Preliminary Steps.....	32
Reference Performance.....	33
Performance with a Legacy Four Cores Machine .....	33
Measure Disk I/O .....	34
Monitoring Performance.....	35
VMware esxtop.....	35
NIC Statistics.....	37
Increasing the Throughput .....	37

Testing from Memory Compared to Testing from Disk .....	37
Running Multiple Sessions or Workers.....	38
Using Nonpersistent Disks and Running Multiple Virtual Machines Concurrently .....	38
Testing with VMware VMmark .....	39
Testing with Netperf.....	39
Testing with FTP .....	40
Testing with SMB .....	40
Testing with NFS.....	41
Testing with DD.....	42
Testing with Rsync.....	42

## Introduction

The Cisco Nexus® 5000 Series Switches provide line-rate 10 Gigabit forwarding and work well with converged network adapters (CNAs) and VMware ESX servers as described and advertised in public documents (see, for instance, [http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9670/white\\_paper\\_c11-496511.pdf](http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9670/white_paper_c11-496511.pdf)).

In a VMware environment using the Cisco Nexus 5000 Series and a server with enough processing cores, you can achieve up to 10 Gbps of throughput by following the testing guidelines provided by VMware (see [http://www.vmware.com/pdf/10GigE\\_performance.pdf](http://www.vmware.com/pdf/10GigE_performance.pdf)).

Even if the network provides 10-Gbps bandwidth, the performance of file transfers is constrained by the following factors (which are not network dependent):

- **Disk:** The I/O performance of disk storage limits the maximum achievable performance, so any file application test benchmark should be accompanied by information about the disk I/O performance.
- **TCP stack:** TCP stack implementation varies by OS, and newer OS versions support features such as the TCP window scale option and autotuning based on the measured latency. These features are critical because even with low latency, due to the high speed, the Bandwidth Delay Product (BDP) value exceeds the standard maximum receive window of 64 KB; hence, you need TCP tuning.
- **Socket buffer size:** The file application's socket buffer size limits the maximum size of the TCP window. For an application to achieve more than 200 to 400 Mbps, you need to modify the settings of **SO\_RCVBUF** and **SO\_SNDBUF** to values similar to those indicated in the VMware document "10Gbps Networking Performance," ([http://www.vmware.com/pdf/10GigE\\_performance.pdf](http://www.vmware.com/pdf/10GigE_performance.pdf)) with configurations which are application dependent.
- **Processing cores:** In a VMware environment, a benchmark from an individual virtual machine will not provide a valid measurement of server performance. A valid test should run as many virtual machines as the number of cores, with each virtual machine affinitized to each core, or at the very least an individual virtual machine should have multiple virtual CPUs (vCPUs) and the guest OS should be configured for receive-side scaling (RSS). Ultimately, the test should look at the aggregate throughput from all virtual machines running on the same machine and not at the throughput of a single virtual machine. Please refer to VMware documentation for instructions on how to run a proper server benchmark measurement.
- **Transmit (TX) compared to receive (RX) performance:** Network adapters may offer better performance when transmitting than when receiving because the transmission may offer more offload capabilities (such as large-segment offload). As a result, a back-to-back test can potentially be throttled by the receive server. Given a machine with enough cores, the effect of this problem is negligible, but with a four-core system as an example, the server could transmit close to 10 Gbps but not be able to receive more than 4.5 Gbps. When engineering a proper test with such a system, it may be better to use more machines on the receive side.
- **Configuration:** A direct implication of the scenario just discussed is that to measure the performance of a server virtual machine, the client (or preferably multiple clients) needs to be configured to allow the virtual machine to achieve the appropriate performance level.
- **Test tool:** The correct way to test the performance capabilities of a server system that abstracts from the server storage I/O capabilities and from the file applications constraints is to test from memory with tools like Netperf and NetIQ Chariot, which allow you to set the proper buffer socket size. In addition, you should run several instances of the testing tool (for example, Netperf) to make use of all the available server cores.

This document provides tuning tips for virtualized servers for maximum performance in the presence of 10 Gigabit adapters.

The test cases that benefit from these recommendations include:

- Virtual machine-to-virtual machine performance tests (with virtual machines running on different VMware ESX hosts and each VMware ESX host equipped with 10 Gigabit adapters) with one or more virtual machines per VMware ESX host
- Virtual machine-to-physical clients, with one or multiple clients and one or more virtual machines per VMware ESX host

For more information, refer to the following VMware publications:

- **10Gbps Networking Performance:**  
[http://www.vmware.com/pdf/10GigE\\_performance.pdf](http://www.vmware.com/pdf/10GigE_performance.pdf)
- **Performance Best Practices for VMware vSphere 4.0:**  
[http://www.vmware.com/pdf/Perf\\_Best\\_Practices\\_vSphere4.0.pdf](http://www.vmware.com/pdf/Perf_Best_Practices_vSphere4.0.pdf)
- **VMware technical document repository:**  
<http://www.vmware.com/resources/techresources/cat/91>
- **VMware vSphere 4 command line:**  
[http://www.vmware.com/pdf/vsphere4/r40/vsp\\_40\\_vcli.pdf](http://www.vmware.com/pdf/vsphere4/r40/vsp_40_vcli.pdf)

**Useful commands:**

[http://pubs.vmware.com/vsp40/server\\_config/r\\_esx\\_technical\\_support\\_commands.html](http://pubs.vmware.com/vsp40/server_config/r_esx_technical_support_commands.html)

After reading this document, you should be able to do the following to successfully measure tuning and performance:

- Make sure the necessary hardware is in place
- Measure local disk performance before running any I/O tests
- Measure the BDP in low-latency, 10-Gbps environments
- Understand the disparity between transmit and receive performance for back-to-back tests and hence the need to provision and tune enough clients
- Tune VMware VMkernel and the guest OS
- Allow the TCP stack enough time to converge (perform autotuning) for the best throughput
- Understand the importance of the socket buffer size used by a given application
- Use VMXNET3 features in the guest OS
- Use Microsoft Windows 2008 as a guest OS (in comparison to Microsoft Windows 2003)
- Use the hardware offload capabilities of the 10-Gbps adapters, in particular, RSS and segmentation offload capabilities
- Achieve the best performance for a VMware ESX machine by using multiple virtual machines (and hence, multiple virtual network interface cards vNICs) because this type of test can use all the queues of a given physical NIC
- Optimize performance with a single virtual machine
- Use virtual machine-to-core affinity
- Use some common tools for performance measurements such as Netperf and the monitoring tool VMware esxtop
- Understand the difference between running tests from memory and running tests that involve real I/O measurements
- Choose the best applications for tests that involve I/O
- Easily run tests that use multiple virtual machines

## Preliminary Verification Steps

### VMware Performance Reporting Requirements

VMware requires that certain information to be readily available for performance reporting.

- Host hardware configuration
  - Manufacturer and model of all main hardware components
  - Firmware versions for all relevant components
  - Quantity, type, and speed of CPUs
  - Amount of memory
  - Storage: array configuration, number and speed (RPM) of disks backing the logical unit number (LUN), and RAID configuration
  - Quantity and type of network adapters
  - VMware ESX version and build number
  - VMware Virtual Machine File System (VMFS) version
  - Any nondefault or configurable hardware or BIOS settings (memory interleaving configuration, for example)
  - Any nondefault VMware ESX settings
- Guest configuration
  - Number of virtual CPUs
  - Amount of virtual memory
  - Quantity and type of virtual disk: virtual Small Computer System Interface (SCSI) adapter type, size, and disk format
  - Virtual disk mode: independent persistent, independent nonpersistent, or snapshot
  - Method of creation for virtual disks (GUI or command-line interface [CLI])
  - If the virtual disks were created using the command line, list of which options were used: thick eager zeroed, thick lazy zeroed, or thin
  - Quantity and type of virtual network adapters
  - Quantity and type of virtual storage adapters
  - Shares, reservations, limits, and affinities of CPU, memory, and disk
  - Guest OS and version (including service pack) installed

For more information, please refer to [https://www.vmware.com/pdf/VI3.5\\_Performance.pdf](https://www.vmware.com/pdf/VI3.5_Performance.pdf).

### PCI Express Connectivity

Before troubleshooting performance problems make sure that 10-Gbps adapters are installed in the proper PCI Express (PCIe) slot to avoid an unnecessary bottleneck in the system. The minimum requirement to achieve 10-Gbps throughput in one direction is installation of an adapter in a PCIe 1.0 8X slot or higher.

Table 1 lists the bandwidth in each direction with increasing numbers of PCIe lanes.

**Table 1.** Bandwidth in Each Direction for PCIe 1.0 and 2.0

Number of Lanes	Bandwidth per Direction for PCIe 1.0	Bandwidth per Direction for PCIe 2.0
1	250 Mbps and 2 Gbps	500 Mbps and 4 Gbps
2	500 Mbps and 4 Gbps	1 Gbps and 8 Gbps
4	1Gbps and 8 Gbps	2 Gbps and 16 Gbps
8	2 Gbps and 16 Gbps	4 Gbps and 32 Gbps
12	3 Gbps and 24 Gbps	6 Gbps and 24 Gbps
16	4 Gbps and 32 Gbps	8 Gbps and 64 Gbps
32	8 Gbps and 64 Gbps	16 Gbps and 128 Gbps

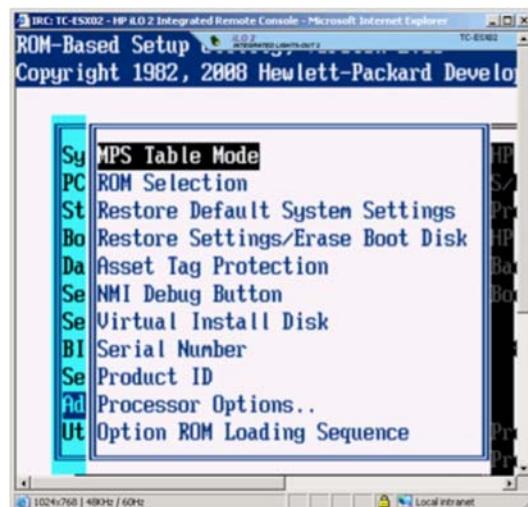
### BIOS Configuration

As indicated in the VMware document “Performance Best Practices for VMware vSphere 4.0,” you should make sure that the BIOS is configured correctly to support the following settings:

- Intel VT-x or AMD AMD-V should be used.
- All CPU sockets and all cores should be enabled.
- Hyperthreading should be enabled.

Figure 1 shows the BIOS setup on an HP DL380 G5, on the Advanced Configuration tab. Choose Processor Options and verify that the features noted here are correctly enabled.

**Figure 1.** BIOS Setup



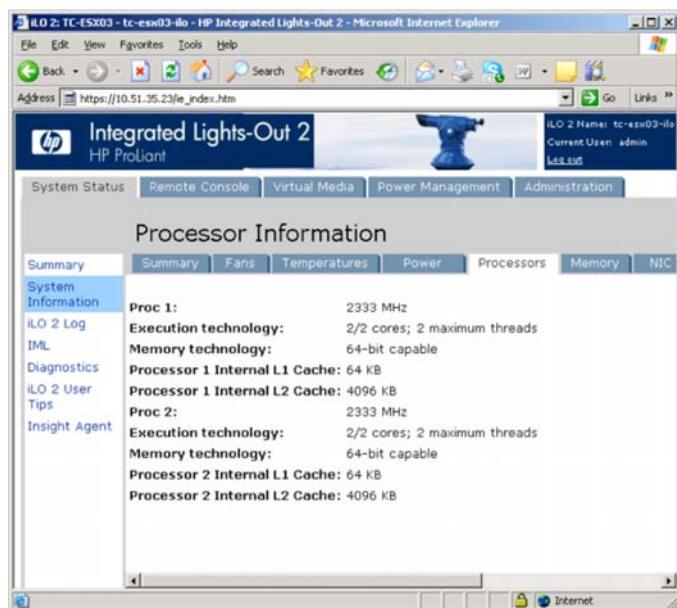
You should also verify the interrupt request (IRQ) mapping under both the USB configuration and the PCI configuration (see the next section).

### System Information

Before proceeding with the tests, collect the information about your system hardware. Figure 2 shows the system information for an HP DL380 G5 with two sockets and two cores per socket.

The processor used in this test is an Intel Xeon 5140, which is a dual-core, 65-nanometer (nm) processor that supports Intel Virtualization Technology (VT). You can obtain this information by simply looking at the console screen of the virtualized server.

**Figure 2.** Gathering Server Information via iLO



### Minimum Hardware Requirements

To achieve close to 9-Gbps performance, you need at least a dual-core, dual-socket platform with 8 Gbps of RAM and a speed of approximately 2.66 GHz. This minimum configuration heavily relies on the offload capabilities of the 10-Gbps card.

For the server to be able to send and receive traffic at close to 10 Gbps, you should use at least a dual-socket, quad-core configuration.

### Minimum Software Requirements

The performance tuning described in this guide is based on VMware ESX 4 Build 164009 with Microsoft Windows Server 2008 with Service Pack 2 (SP2) as a guest operating system.

Microsoft Windows 2008 SP2 offers significant performance advantages compared to Microsoft Windows 2003 because of several improvements in the following areas:

- TCP stack implementation with autotuning capabilities
- Receive-side scaling capabilities when used in conjunction with the VMXNET3 paravirtualized adapter on VMware ESX4 (RSS is not enabled by default; you must make sure that VMXNET3 is chosen and that the guest OS is configured to use this capability)

### Measuring the Disk I/O Capabilities of the Server

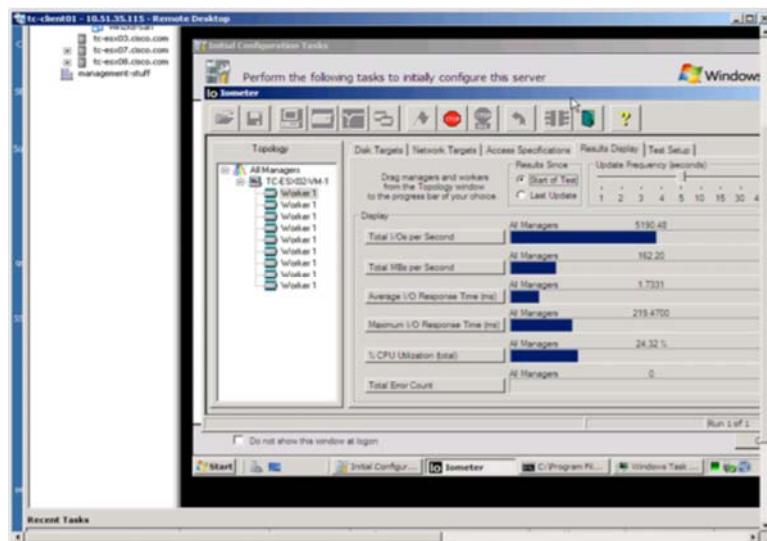
Before running any test that involves I/O, you need to know how much data a given storage configuration is capable of handling. You can run a test with a tool such as lometer.

To determine the maximum I/O, you should run an lometer test with multiple workers (the number of workers is empirical; above a certain number of workers, the aggregate performance does not increase).

Figure 3 shows the I/O performance capabilities of the storage of a virtual machine that is based on Fibre Channel and runs at about 162 Mbps. If this storage uses the Server Message Block (SMB) Protocol, this is also the theoretical maximum achievable performance.

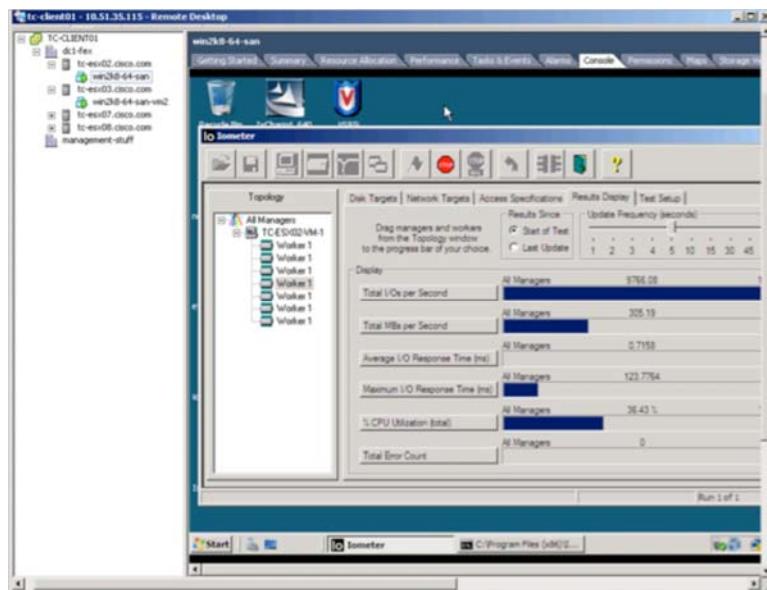
The test illustrated here is focused on 100 percent read operations.

**Figure 3.** Measuring the Performance of Fibre Channel Connected Disks via lometer



When this virtual machine uses a local attached disk, the performance is approximately 300 Mbps (Figure 4).

**Figure 4.** Measuring the Performance of Directly Attached Storage via lometer



## Network-Performance Concepts

This section discusses networking related concepts used in determining performance.

### TCP Window-Size Tuning

The single most important aspect of tuning a server for 10-Gbps performance is making sure that the TCP advertised window is sufficient to achieve the needed performance.

RFC 793, which defines TCP, describes a 16-bit receive-window field that tells the sender how many bytes are available in the receive buffer. The receive-window size is advertised with each acknowledgment (ACK) returned to the sender. The receive-window field defines the maximum number of bytes that the sender can send without waiting for an acknowledgment. Because the receive-window field is only 16 bits, the maximum amount of buffer space that can be advertised is only 64 KB.

RFC 1072 added a TCP option called window scale that extends the receive-window field to 30 bits, for a maximum buffer size of 1 GB. This option is clarified and redefined in RFC 1323, the specification with which all implementations comply. The window-scale option provides a 14-bit left-shift value that defines the number of bitwise places that the 16-bit value, which is advertised in the window field, should be moved leftward.

The receive buffer size limits the amount of data that the transmitter can send without overrunning the receive buffer. If the TCP buffer is too small to hold enough packets to fill a high-bandwidth or high-latency network link, transmission of packets periodically stops until the receiver acknowledges some of the packets in the buffer. The bandwidth is only partially utilized.

The advertised TCP window needs to be calculated based on the round-trip time (RTT) as follows:

$$\text{window size in bytes} = (\text{bandwidth} * \text{RTT delay})/8$$

The maximum segment size (MSS) is 1460, so the window must be an even multiple of 1460 bytes.

In a virtualized environment, TCP tuning needs to be performed within the guest operating system.

In a 10-Gbps environment, even small RTT delays can result in a BDP greater than the default 64 KB due to the importance of the bandwidth value.

Here is an example with servers directly connected using a cut-through switch:

- The cut-through switch has 3.2 microseconds of latency (a store-and-forward switch would typically deliver a latency closer to 20 microseconds)
- Kernel-to-kernel latency is approximately 6.7 microseconds (depending on the guest OS and the hypervisor)
- TCP socket-to-TCP socket latency is approximately 12 to 29 microseconds (depending on the guest TCP stack and the network adapter)

This example yields a RTT of  $2 * 29$  microseconds (depending on the OS stack and the network adapter).

At 29 microseconds, the BDP for 10-Gbps throughput is approximately  $10 \text{ Gbps} * 2 * 29 \text{ microseconds}/8 =$  approximately 72.5 KB, which requires the TCP window-scale option.

One of the initial steps in testing should be to measure the socket-to-socket application latency with some tests (such as a ping) to accurately measure the latency from virtual machine to virtual machine.

In a back-to-back low-latency configuration, a reasonable BDP value ranges from 64 to 384 KB, or even 512 KB.

### Achievable Throughput Convergence Time

Different TCP stack implementations have different ramp-up approaches, and some achieve the maximum possible throughput faster than others. In general, this means that for a throughput test, you need to factor in a ramp-up period before you can observe the full throughput.

### TCP Send and Receive Buffers and Socket Buffers

The window advertised by TCP is the minimum needed based on the TCP window and the socket buffer size. The buffer size is specified by the application when requesting the connection. When an application requests a socket, it should specify a larger buffer size to achieve optimal throughput.

For most applications, the socket buffer size is hardwired, although some applications offer the option of specifying the buffer size, and others rely on registry settings. The main parameters that determine the socket buffer size are:

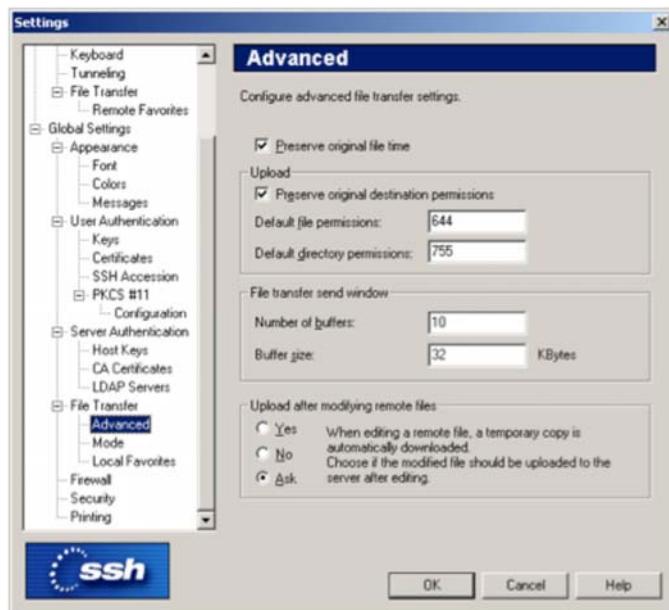
- **SO\_RCVBUF**
- **SO\_SNDBUF**

For example, rsync lets you specify the socket buffer as follows:

```
rsync -av --sockopts=SO_SNDBUF=2000000,SO_RCVBUF=2000000
```

Figure 5 shows another example of buffer sizing options provided by an application (a Secure Shell [SSH] client in this case).

**Figure 5.** Buffer Tuning for an SSH Client



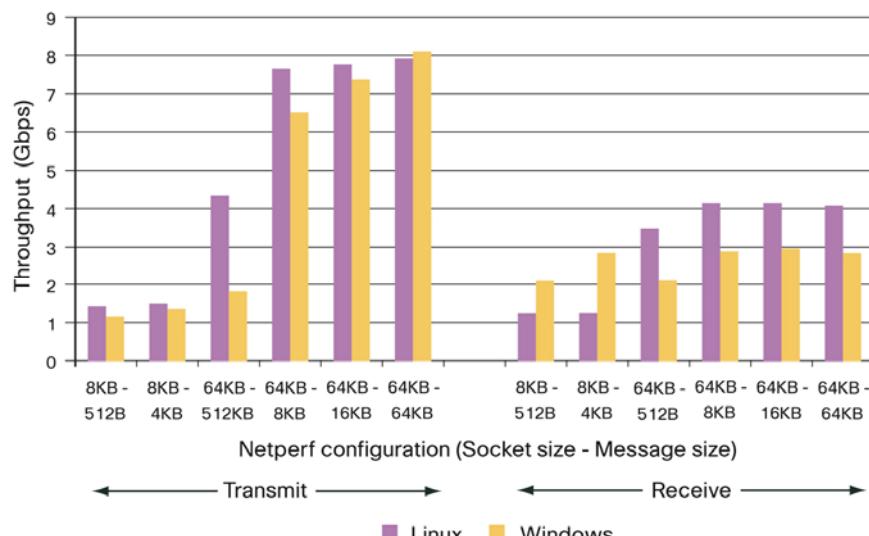
The TCP congestion window is calculated based on the minimum of the maximum TCP receive window and the buffer space that the socket requests.

The socket buffer size should be calculated in the same way as the TCP receive window is calculated: that is, based on the BDP.

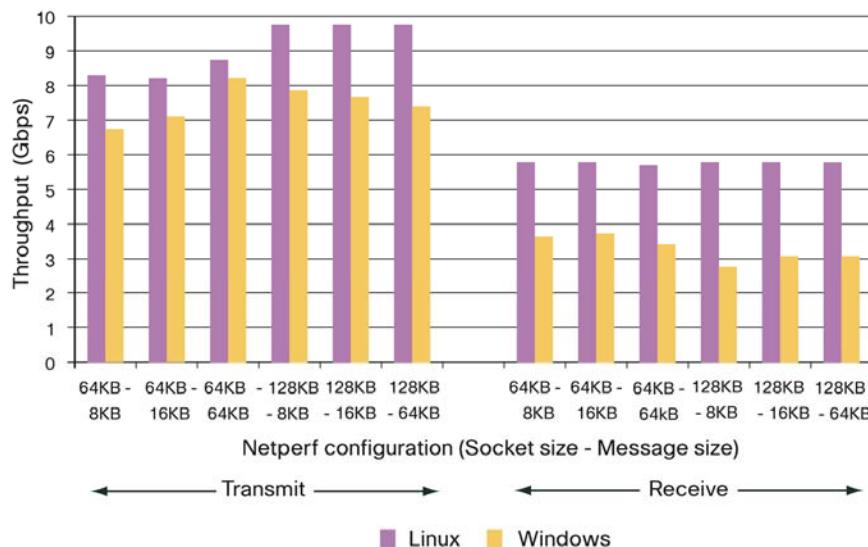
In a back-to-back low-latency configuration, a reasonable BDP value ranges from 64 to 384 KB, or even 512 KB.

The relationship between socket buffer and achievable performance is demonstrated by the test results shown in Figures 6 and 7, from the VMware document [https://www.vmware.com/pdf/VI3.5\\_Performance.pdf](https://www.vmware.com/pdf/VI3.5_Performance.pdf).

**Figure 6.** Impact of Buffer Size on the Application Performance



**Figure 7.** Impact of Jumbo Frames on Application Performance with Varying Socket Buffer Sizes



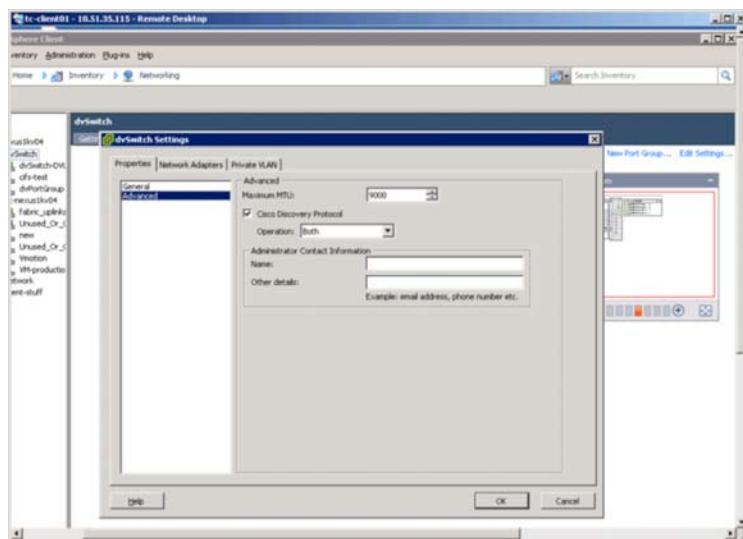
### Jumbo Frames

Jumbo frames (9-KB packets) greatly enhance the maximum achievable performance. Jumbo frames are enabled at the VMware vNetwork Distributed Switch (vDS) or virtual switch (vSwitch level). For information about how to enable jumbo frames, see:

- [http://pubs.vmware.com/vsp40/server\\_config/c\\_advanced\\_networking.html#1\\_7\\_13\\_10\\_10\\_1](http://pubs.vmware.com/vsp40/server_config/c_advanced_networking.html#1_7_13_10_10_1)
- [http://pubs.vmware.com/vsp40/server\\_config/c\\_enabling\\_jumbo\\_frames.html#1\\_7\\_13\\_10\\_10\\_1](http://pubs.vmware.com/vsp40/server_config/c_enabling_jumbo_frames.html#1_7_13_10_10_1)

Figure 8 shows how to enable jumbo frames on a VMware vDS.

**Figure 8.** Enabling Jumbo Frames on VMware vDS



You also need to make sure that the guest OS network adapter is configured to use jumbo frames.

For VMware VMkernel, the jumbo frames configuration is performed using the service console CLI as described here:

- Enter the **esxcfg-vmknic -l** command to display a list of VMware VMkernel interfaces
- Enter the **esxcfg-vmknic -a -i <ip address> -n <netmask> -m <MTU> <port group name>** command to create a VMware VMkernel connection with jumbo frame support

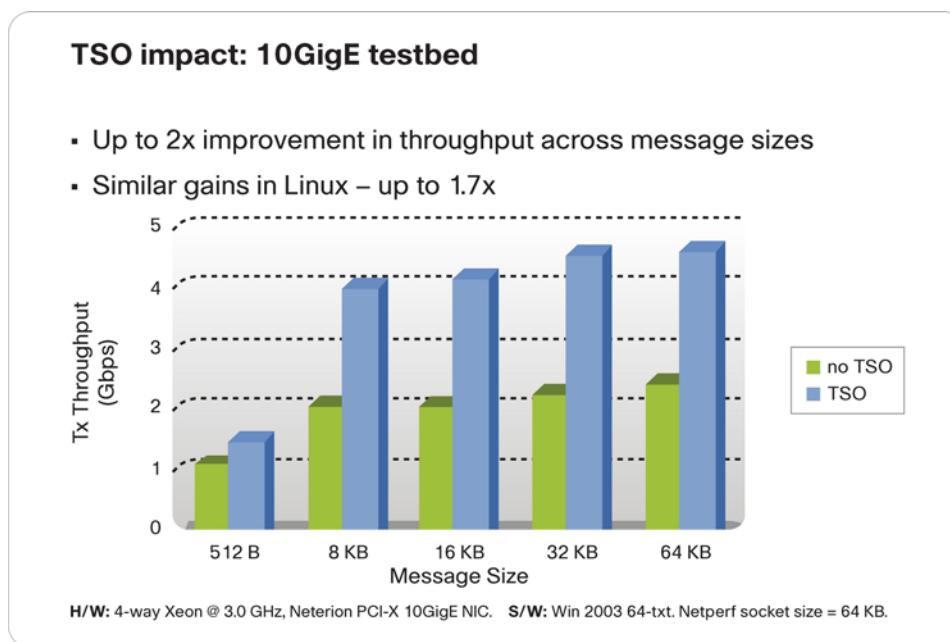
## Read and Write Block Size

The block size is the amount of data copied from the application memory space to the kernel memory space. With large block sizes, performance increases, because fewer interrupts are generated by the system `write()` call. Larger read and write blocks imply fewer CPU cycles to move data from the server to the wire.

To achieve reasonable levels of performance, an application should use at least a block size of 8 KB, as described in the document “10Gbps Networking Performance” at [http://www.vmware.com/pdf/10GigE\\_performance.pdf](http://www.vmware.com/pdf/10GigE_performance.pdf).

Message size has a significant effect on the achievable performance, as shown in the results (which also show the effect of transmit segment offload) in Figure 9.

**Figure 9.** Impact of the Message Size on Application Performance



## Transmit Performance Compared to Receive Performance

Most server setups in general offer less performance in receive mode (traffic received by the server) than in transmit mode (traffic sent by the server), so in typical performance tests with two machines with identical hardware, the sender can easily overwhelm the receiver, causing it to drop frames. One reason for this difference is that most hardware offloads help with the sending performance more than they do with the receive performance. Newer adapters and OS improvements are addressing this phenomenon, so the performance disparity may decrease.

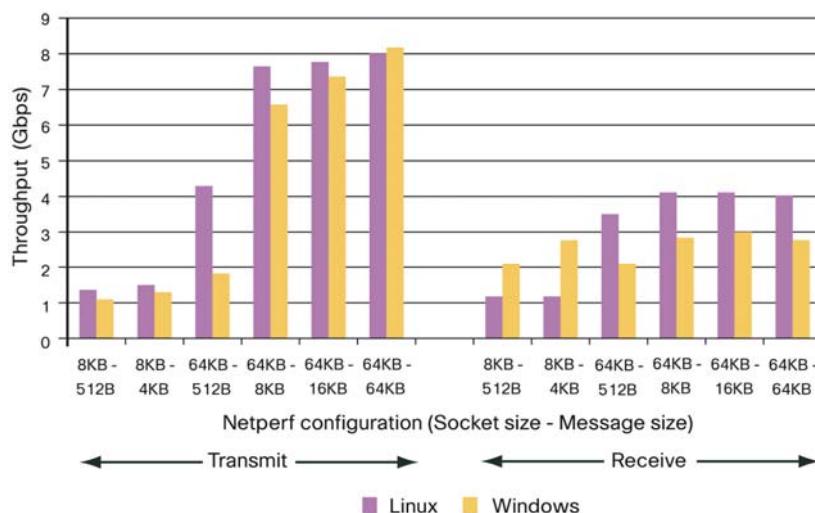
To make sure that the sender and receiver do not throttle performance because of drops, you should make sure that selective acknowledgment (SACK) is enabled in the adapter and operating system.

When operating with TCP offload disabled, the TCP/IP stack in Microsoft Windows automatically implements SACK.

When operating with TCP offload enabled and the TCP/IP stack implemented in hardware on the network adapter card, you may need to verify that SACK is enabled at the network adapter layer.

Similarly, if TCP offloads are implemented in hardware, you may want to make sure that the TCP stack is still providing RFC 1323 windows scaling.

Figure 10 provides an example of the disparity in performance between transmit and receive performance, using test results from VMware.

**Figure 10.** Transmit Versus Receive Performance

In a given hardware configuration, the hardware could yield the following performance: a virtual machine could transmit at 8 Gbps but receive at only 4 Gbps in Linux and 3 Gbps in Microsoft Windows. Technologies such as large-receive offload (LRO) may change these performance numbers.

Because of the difference between receive and transmit performance, when measuring throughput performance, you typically should run multiple receive clients for a single sending server.

### Typical Offload Technologies

The network adapters offer a number of features to optimize performance:

- **TCP checksum offload:** The TCP checksum offload option enables the network adapter to compute the TCP checksum on transmit and receive operations, which saves the CPU from having to compute the checksum. The performance benefits of checksum offload vary by packet size. Small packets achieve little or no savings with this option, while large packets achieve greater savings. Savings for a maximum transmission unit (MTU) of 1500 is typically about 5 percent reduction in CPU utilization, and for an MTU of 9000 (jumbo frames), the savings is approximately 15 percent reduction in CPU utilization.
- **Receive-side scaling queues (disabled, 2, 4, and 8):** RSS facilitates distribution of traffic to the available cores in the system by separating the traffic into multiple queues (as many as the number of cores with which you want to process network traffic). Microsoft Windows also implements RSS. This number should be configured to match the number of cores available. When using VMware, RSS can be used within the guest OS to distribute the traffic across multiple vCPUs.
- **Large-send offload:** When this option is enabled, the OS can pass large message sizes to the network adapter, and the network adapter will slice them up based on the MSS. This feature relieves the CPU from having to deal with TCP segmentation. The TCP large-send offload (LSO) option allows the TCP layer to build a TCP message up to 64 KB long and send it in one call down the stack through IP and the Ethernet device driver.

In a Microsoft Windows system, these options can be activated or deactivated by using the driver configuration. Select the local area connection and then choose Properties > Configure and the Advanced configuration tab.

### VMware vSphere 4 Tuning

VMware vSphere 4 includes several enhancements that can be used for 10-Gbps attached servers:

- Each virtual machine can have up to 8 vCPUs and 255 GB of memory

- SCSI and NIC adapters are faster
- The paravirtualized SCSI adapter provides better storage I/O rates
- The VMXNET3 NIC supports several offloads, including multiple queues with associated receive-side scaling in Microsoft Windows 2008 as well as message-signaled interrupt extended (MSI-X) interrupt delivery
- Each virtual machine performs up to 200,000 I/O operations per second (IOPS)

For more information about VMware vSphere 4 networking, see:

- [http://www.vmware.com/files/pdf/VMW\\_09Q1\\_WP\\_vSphereNetworking\\_P8\\_R1.pdf](http://www.vmware.com/files/pdf/VMW_09Q1_WP_vSphereNetworking_P8_R1.pdf)
- [http://www.vmware.com/files/pdf/vsphere\\_performance\\_wp.pdf](http://www.vmware.com/files/pdf/vsphere_performance_wp.pdf)

Comprehensive VMware vSphere 4 tuning varies depending on the guest OS. This document assumes that the guest OS is Microsoft Windows 2008. For the requirements for Microsoft Windows 2008, see <http://www.microsoft.com/windowsserver2008/en/us/system-requirements.aspx>.

According to VMware, performance improvement for VMware ESX 4 in comparison to VMware ESX 3.5 is as follows:

- With 1 virtual machine, you get 14 percent more network performance
- With 8 virtual machines, you get 59 percent more network performance
- With 16 virtual machines, you get 86 percent more network performance

Among the new advanced options that VMware vSphere 4 offers are some that are specifically targeted at improving receive performance:

- NetQueue
- Large-receive offload (configurable with **Net.VmkInxLROEnabled**)

### **Physical NICs (vmnics) Capabilities and Troubleshooting**

Before configuring networking options on the guest OS or the VMware VMkernel interface, a good practice is to retrieve the information about the capabilities of the physical adapters (vmnics) on the VMware ESX server (Figure 12).

You can enter the command ethtool at the service console CLI to retrieve this information. Table 2 shows the syntax.

**Table 2.** ethtool Command Syntax

Command	Description
<b>ethtool -c --show-coalesce DEVNAME</b>	Show coalesce options
<b>ethtool -C --coalesce DEVNAME</b>	Set coalesce options
<b>ethtool -k --show-offload DEVNAME</b>	Get protocol offload information
<b>ethtool -K --offload DEVNAME</b>	Set protocol offload
<b>ethtool -i --driver DEVNAME</b>	Show driver information
<b>ethtool -S --statistics DEVNAME</b>	Show adapter statistics

You can also use the command **esxcfg-nics -l** and then **esxcfg-info -n <vmnic>**.

**Figure 11.** Output of the Command esxcfg-nics -l

```

tc-esx02 - tc-esx02-console - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
----Driver..... ixgbe
----Network Hint.....
----MTU..... 9000
----Configured Speed..... 0
----Actual Speed..... 0
----Configured Duplex..... 2
----Actual Duplex..... 2
----Link Up..... false
----Wake on LAN supported ..... false
----Wake on LAN enabled ..... false
----CDP Network..... false
\==Hardware Capabilities :
    ----Vlan Tag..... true
    ----Vlan Untag..... true
    ----Checksum Offload..... true
    ----Tcp Segmentation Offload..... true
    ----Scatter Gather Tx..... true
\==Supported Speeds and Duplex :
    ----Supported Mode..... 10000/full
\==+PCI Device :
    ----Bus..... 0x19
    ----Slot..... 0x00
    ----Function..... 0x01
Connected to tc-esx02      SSH2 - aes128-cbc - hmac-md5 - none | 60x24

```

### Distinguishing VMware VMkernel Interface Performance from vNIC Performance

When optimizing networking performance for VMware ESX deployments, you should be sure to separate the performance optimizations into two categories:

- Performance of the VMware VMkernel TCP/IP stack
- Performance of the guest OS virtual machines

While VMware VMkernel and Guest OSs may share the same physical adapter, the TCP stack of the VMware VMkernel is fully under VMware ESX hypervisor control, while the TCP stack of the Guest OS is under the Guest OS control, but it may benefit from the vNIC adapter optimizations.

#### VMware VMkernel and Service Console TCP/IP Stack

VMware vSphere 4 also improves the TCP/IP stack implementation (referred to as **tcpip2**) that is used by the VMware VMkernel interface and by the service console:

- This new TCP/IP stack is based on FreeBSD 6.1
- The new stack has improved locking and threading capabilities compared to the current TCP/IP stack
- The new stack enables better use of multiple CPU cores for improved VMware ESX scalability

These improvements benefit the performance of applications such as VMware VMotion, iSCSI, and Network File System (NFS). The VMware VMkernel interface by default supports TCP segmentation offload (TSO).

In VMware vSphere 4, TSO is enabled by default in the VMware VMkernel interface (**vmknic**). To verify that TSO is enabled for the VMware VMkernel interface, you can enter the command **esxcfg-vmknic -l**.

#### VMXNET Generation 3 and TSO

The VMXNET generation-3 (VMXNET3) adapter is a new adapter available in VMware vSphere 4. This adapter is a paravirtualized device that supports MSI and MSI-X, RSS (with Microsoft Windows 2008), TSO, TCP checksum offload, and large transmit and receive ring sizes, which can be configured from within the guest OS.

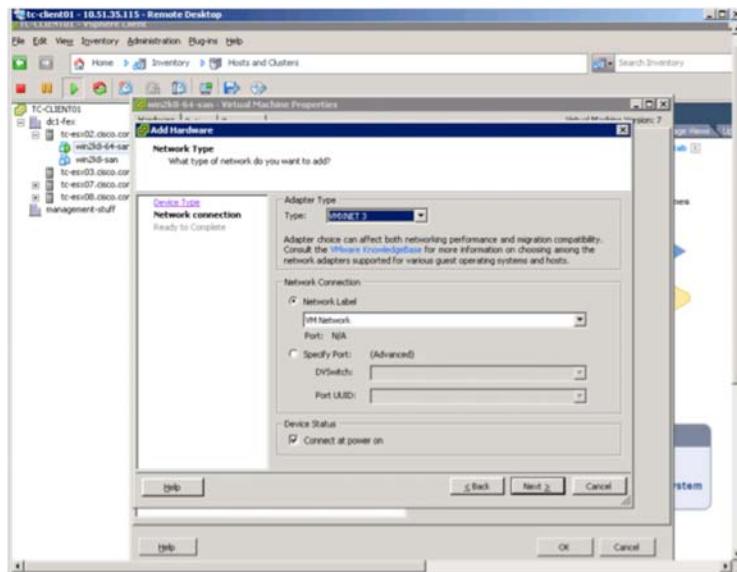
At the time of this writing, VMXNET3 is supported on the following operating systems:

- 32- and 64-bit versions of Microsoft Windows XP and later

- 32- and 64-bit versions of Red Hat Enterprise Linux 5.0 and later
- 32- and 64-bit versions of SUSE Linux Enterprise Server 10 and later
- 32- and 64-bit versions of Asianux 3.0 and later
- 32- and 64-bit versions of Debian 4.0 and Ubuntu 7.04 and later
- 32- and 64-bit versions of Sun Solaris 10 U4 and later

From the guest OS, to use TSO, you need to select either the enhanced vmnet network adapter or, better, the VMXNET3 adapter as depicted in Figure 12.

**Figure 12.** VMXNET3 as Seen from vCenter

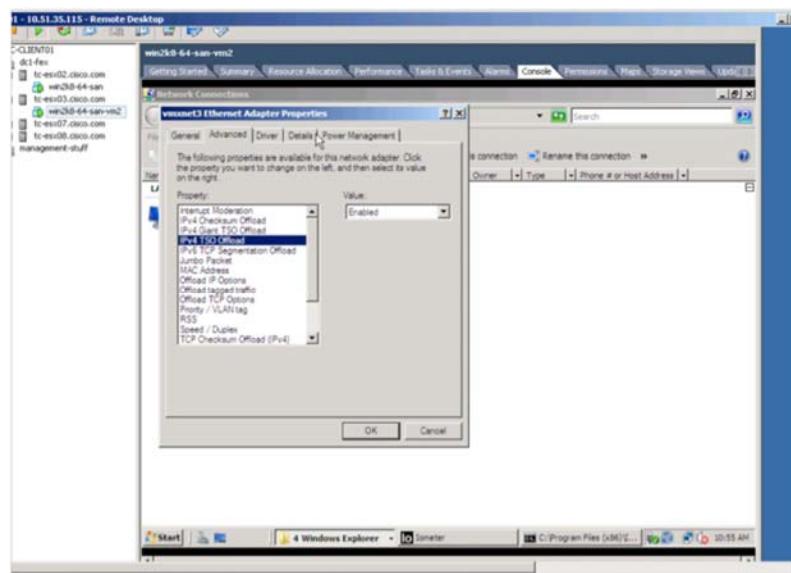


Within the guest OS, you need then to configure the network adapter to support TSO.

TCP chimney is not supported in VMXNET3, only TCP segmentation offload.

In Microsoft Windows 2008, you can perform this configuration by using the netsh CLI command or the network adapter properties as shown in Figure 13.

**Figure 13.** Configuring Network Adapters Properties from the Guest OS



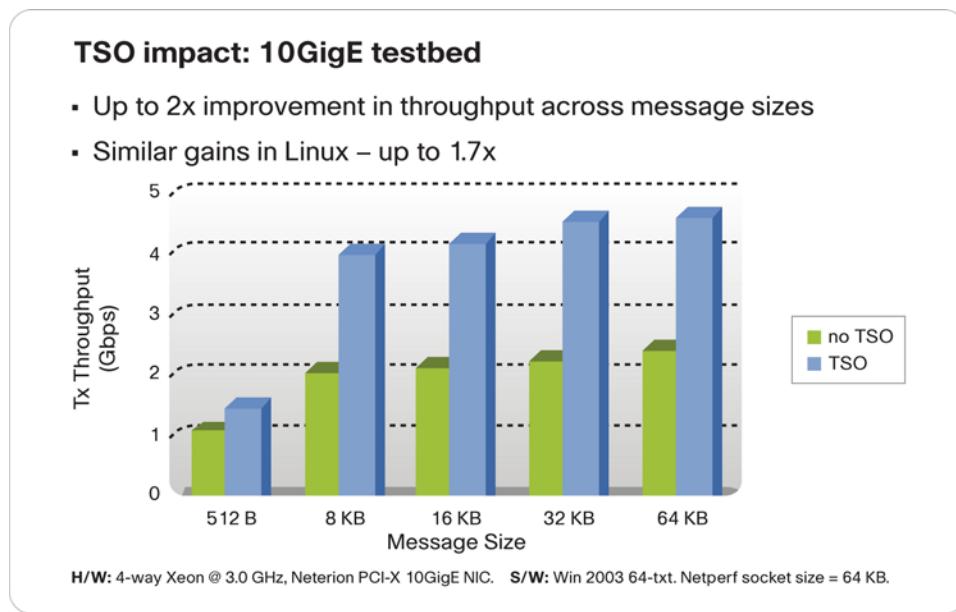
In Linux, you can enable or disable the segmentation offload capabilities with the **ethtool** command. The **ethtool -k** command (notice the lowercase “k”) enables you to see which offload capabilities are enabled. Here is an example of **ethtool -k eth0** output:

```
rx-checksumming: off
tx-checksumming: off
scatter-gather: off
tcp segmentation offload: off
```

To enable segmentation offload, you can enter the command **ethtool -K eth0 tso on**.

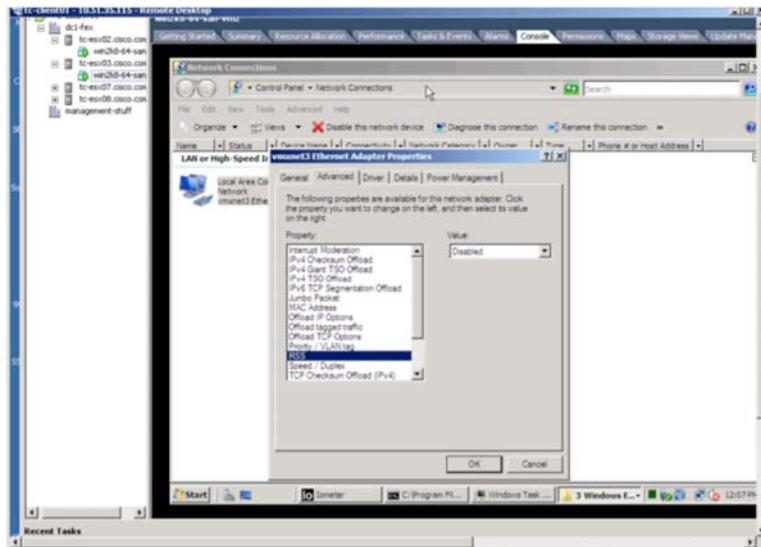
Figure 14, from VMware, shows the performance improvements when using TSO.

**Figure 14.** Performance Improvements with TSO



The VMXNET3 adapter also supports RSS with Microsoft Windows 2008. In this case, in the Microsoft Windows 2008 guest OS, you have to explicitly enable RSS in the network adapter properties as shown in Figure 15 or using the **netsh** CLI command: **netsh int tcp set global rss=enabled**.

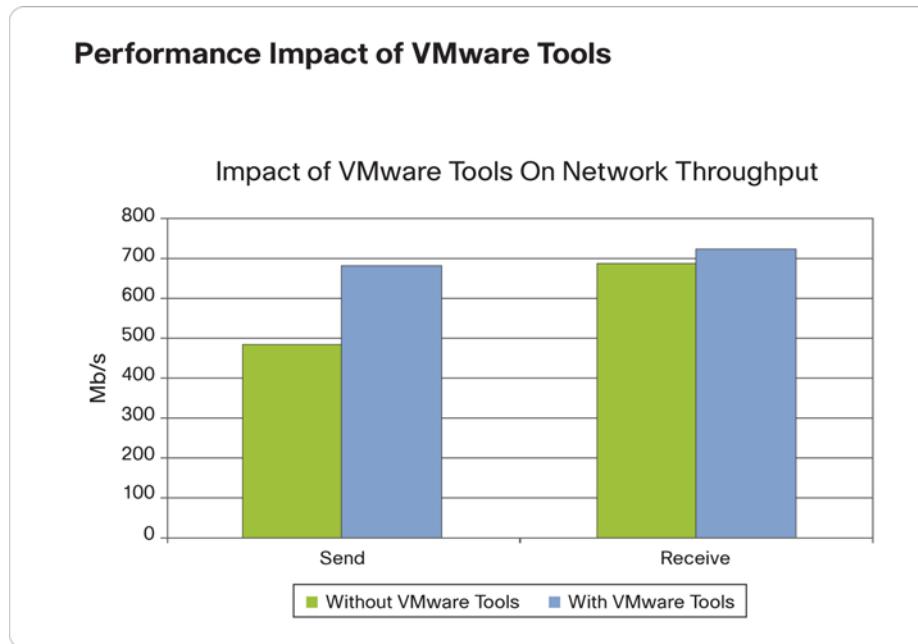
**Figure 15.** Configuring RSS from the Guest OS



### VMware Tools

If the virtual machine is not set to upgrade VMware Tools at each power-on, you must upgrade VMware Tools manually as described in the administration guide. Figure 16 shows a VMware Tools screen.

**Figure 16.** Performance Improvements with VMware Tools

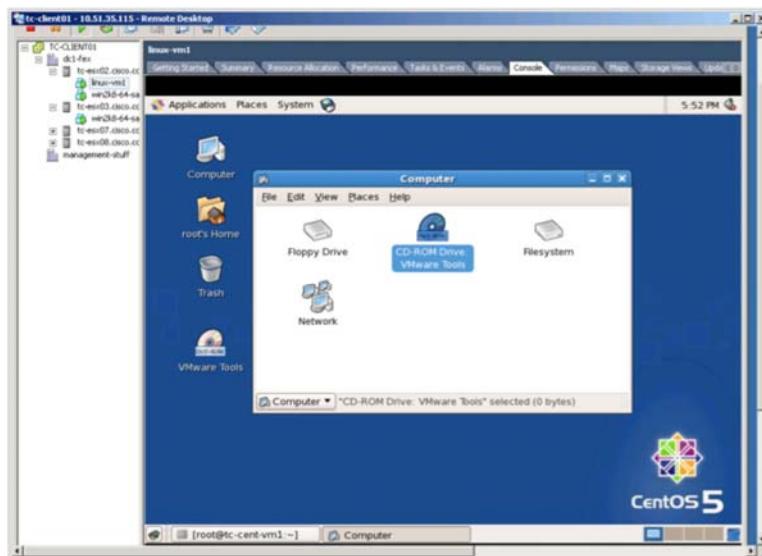


In the case of a Linux guest OS, the procedure for installing VMware Tools is as follows:

- Make sure that the virtual machine is configured with a CD-ROM drive (this is necessary to be able to install the VMware Tools as they appear as a CD at the next step; also see Figure 17).

- From VMware vCenter, right-click Guest - Install VMware Tools; a CD appears as mounted on the guest OS.
- From /usr/sbin/, run ./vmware-config-tools.pl.

**Figure 17.** Installing VMware Tools in Linux



Alternatively, you can use:

```
[root@localhost ~]# cd /media/cdrom
[root@localhost ~]# cp VMwareTools-3.0.x-nnnnn.tar.gz /tmp
[root@localhost ~]# cd /tmp
[root@localhost ~]# tar zxvf VMwareTools-3.0.x-nnnnn.tar.gz
```

Change the directory to **vmware-tools-distrib** and enter the following command:

```
[root@localhost ~]# ./vmware-install.pl
```

You should then verify that VMXNET was loaded:

```
[root@tc-vm2-esx03 vsftpd]# ethtool -i eth0
driver: vmxnet3
version: 1.0.0.32-NAPI
firmware-version: N/A
bus-info: 0000:03:00.0
```

If you cannot find VMXNET, you can check in **/etc/sysconfig/network-scripts**.

You may have to restart the network services for the module to be loaded:

```
/etc/init.d/network stop
rmmod pcnet32
rmmod vmxnet
modprobe vmxnet
/etc/init.d/network start
```

To run VMware Tools, enter **/usr/bin/vmware-toolbox**.

## Intel VMDq and NetQueue

### Intel VMDq

Network adapters offer several optimization features for virtualized environments. In the case of Intel, the Intel VMDq technology helps queue traffic to the appropriate virtual machines.

According to the “Intel VMDq Technology” document, when packets arrive at a network adapter enabled and configured for the Intel VMDq enabled and configured network adapter, a Layer 2 classifier and sorter in the network controller sorts and determines the correct destination queue using MAC addresses and VLAN tags. The sorter then places the packet in the receive queue assigned to that virtual machine. The virtual machine monitor (VMM) switch, or hypervisor, routes the packets to the respective virtual machine, which means that the VMM does not have to sort the data itself.

For more information about the Intel VMDQ technology, please refer to:

- [http://www.intel.com/technology/platform-technology/virtualization/VMDq\\_whitepaper.pdf](http://www.intel.com/technology/platform-technology/virtualization/VMDq_whitepaper.pdf)
- [http://www.intel.com/network/connectivity/vtc\\_vmdq.htm](http://www.intel.com/network/connectivity/vtc_vmdq.htm)

To verify that Intel VMDq is working, you can check the statistics for the individual queues of a given interface. First you can see which ports have the driver loaded:

**esxcfg-nics -l**

Then you can query the statistics on these ports using **ethtool**.

If Intel VMDq is enabled, you should see statistics for multiple receive queues, shown as **rx\_queue\_0** through **rx\_queue\_<number>**.

For example, with two queues (and NetQueue enabled), you would see the traffic counters increasing on both queues (you can check the counters with **ethtool -S vmnic<number>**):

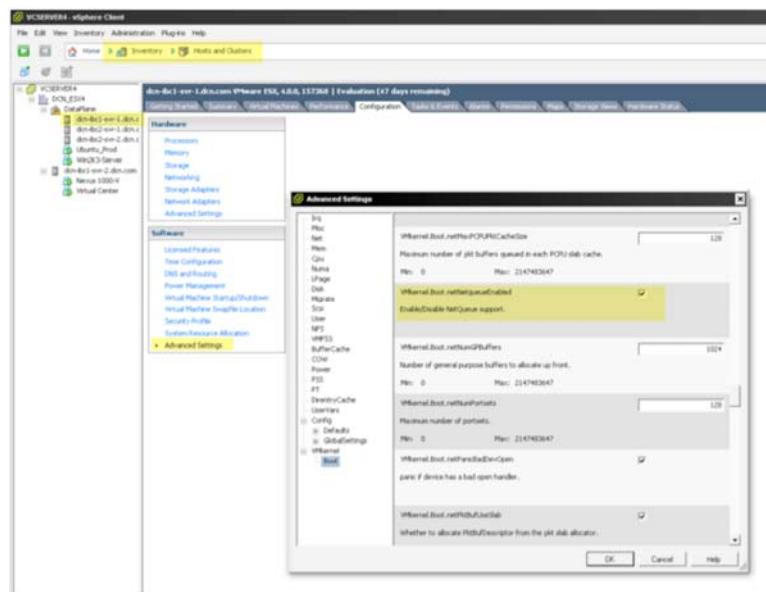
```
rx_queue_0_packets: 603921
rx_queue_0_bytes: 282349079
rx_queue_1_packets: 27998249
rx_queue_1_bytes: 41679757876
```

### NetQueue

VMware ESX 3.5 introduced support for NetQueue. NetQueue allows a NIC to have multiple queues for processing packets that are destined to different CPU cores. NetQueue processing increases the performance on the receive path with multiple virtual machines running on the same host.

To find the NetQueue configuration, choose Configuration > Advanced Settings and select VMkernel.Boot.netNetqueueEnabled and reboot the system (Figure 18).

**Figure 18.** Enabling Netqueue



You can also configure NetQueue by adding the following line to the `/etc/vmware/esx.conf` configuration:

```
/vmkernel/netNetqueueEnabled = "TRUE"
```

If NetQueue is disabled (or if Intel VMDq is not supported), you will see that all traffic goes to a single queue shown in the following example (output was obtained with `ethtool -S vmnic<number>`):

```
tx_queue_0_packets: 2659429
tx_queue_0_bytes: 323395294
tx_queue_1_packets: 0
tx_queue_1_bytes: 0
tx_queue_2_packets: 0
tx_queue_2_bytes: 0
tx_queue_3_packets: 0
tx_queue_3_bytes: 0
rx_queue_0_packets: 72344070
rx_queue_0_bytes: 107288816400
rx_queue_1_packets: 0
rx_queue_1_bytes: 0
rx_queue_2_packets: 0
rx_queue_2_bytes: 0
rx_queue_3_packets: 0
rx_queue_3_bytes: 0
```

Complete Configuration Example for Intel 82598 with Intel VMDq

Make sure NetQueue is enabled; as described in the previous section, the process is as follows:

- Choose Configuration > Advanced Settings > VMkernel
- Select the check box for VMware VMkernel.Boot.netNetqueueEnabled

On the VMware ESX console, identify which vmnics are present by entering this command:

```
esxcfg-nics -l
```

The output will be similar to the following:

```
vmnic0 04:00.00 bnx2 Up 1000Mbps Full 1500 Broadcom Corporation Broadcom NetXtreme II BCM5708  
1000Base-T
```

```
vmnic1 08:00.00 bnx2 Down 0Mbps Half 1500 Broadcom Corporation Broadcom NetXtreme II BCM5708  
1000Base-T
```

```
vmnic2 0b:00.00 ixgbe Up 10000MbpsFull 9000 Intel Corporation 82598EB 10 Gigabit AF Dual Port Network  
Connection
```

```
vmnic3 0b:00.01 ixgbe Up 10000MbpsFull 9000 Intel Corporation 82598EB 10 Gigabit AF Dual Port Network  
Connection
```

```
vmnic4 0d:00.00 ixgbe Up 10000MbpsFull 1500 Intel Corporation 82598EB 10 Gigabit AF Dual Port Network  
Connection
```

```
vmnic5 0d:00.01 ixgbe Up 10000MbpsFull 1500 Intel Corporation 82598EB 10 Gigabit AF Dual Port Network  
Connection
```

The following configuration steps show how to enable four NetQueue instances for an Intel 82598 chip set with two ports.

First unload the driver with the command **vmkload\_mod -u ixgbe**.

Then reload the driver with the correct setup in multiple-queue mode with the command **vmkload\_mod ixgbe  
VMDQ=X,X InterruptType=2,2**.

Here, X is the number of queues (for example, 8 for eight virtual machines, and **InterruptType=2** indicates MSI-X).

X and 2 are repeated once for each physical port (a typical configuration has two 10 Gigabit Ethernet ports, which is why the values are repeated twice).

An alternate configuration uses the **esxcfg-module** command. For example, here the relevant interfaces are the ones indicated as **ixgbe**. To enable the **ixgbe** module for the Intel 82598 chipset, enter:

```
esxcfg-module -e ixgbe
```

```
esxcfg-module -s "InterruptType=2,2 VMDQ=4,4" ixgbe
```

For more information, see <http://kb.vmware.com/kb/1004278>.

To verify the settings, you can use the following command:

```
esxcfg-module -g ixgbe
```

The output should be similar to the following example:

```
ixgbe enabled = 1 options = 'InterruptType=2,2 VMDQ=4,4'
```

This configuration also appears in the esx.conf file:

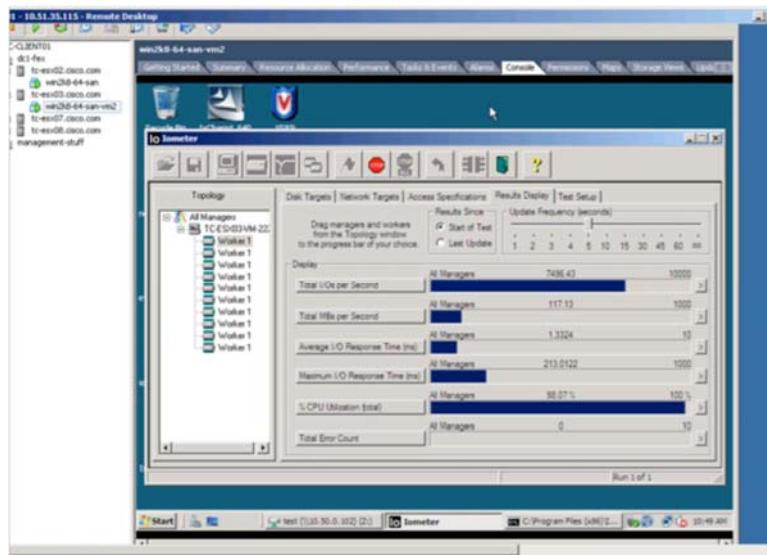
```
/vmkernel/module/ixgbe.o/options = "InterruptType=2,2 VMDQ=2,2"
```

```
/vmkernel/netNetqueueEnabled = "TRUE"
```

### Removing the CPU Bottleneck

When running virtual machine-to-virtual machine tests, the bottleneck in the test may not be the TCP stack operations, the application, the network, or the disk. It may instead be the CPU as shown in Figure 19.

**Figure 19.** Using lometer and Saturating the CPU



The conclusions from this test vary.

### Large-Receive Offload

The LRO feature works similarly to TSO but in the receive direction. VMware vSphere 4 supports LRO, which coalesces TCP packets from the same connection to reduce CPU utilization. Using LRO with VMware ESX provides 40 percent improvement in both throughput and CPU costs.

According to <http://communities.vmware.com/docs/DOC-10892>, this feature is not yet suitable for the guest OS. This document says that the **VmkLnkLROEnabled** command enables large packets for recent Linux guests with VMXNET 2 and 3 and is most likely to benefit hosts with a small number of virtual machines and a few sessions each, where each session has a heavy receive load (more than 1 Mbps). The document states: “This is an experimental feature and has not been tested extensively.”

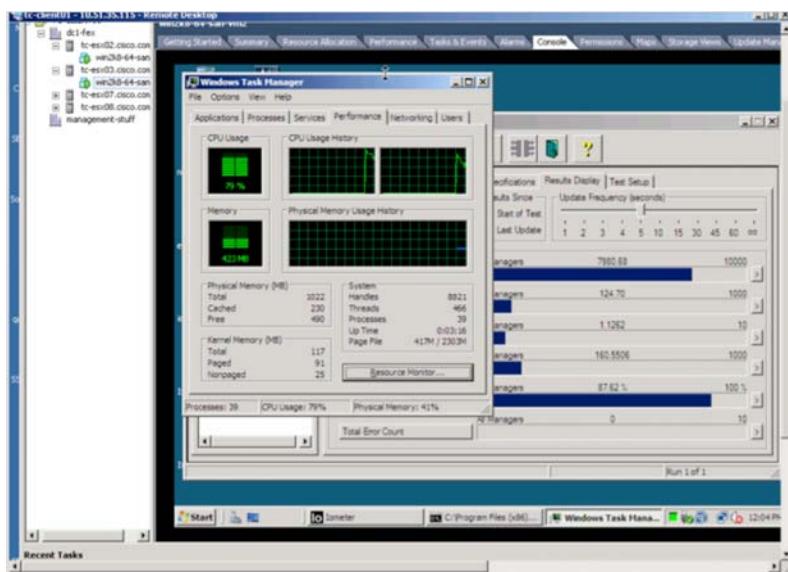
However, for the VMware VMkernel interface, this feature may provide some benefits:

“**TcpipDefLROMaxLength (16000, 1, 65535)** Maximum length for the LRO aggregated packet for vmkernel connections. Increasing this reduces the number of acknowledgments, which improves efficiency but may increase latency.”

### Increasing vCPU and Using RSS

When using VMXNET3, you can use RSS. To do so requires configuring multiple vCPUs for the guest OS and enabling the RSS feature from the adapter properties on the guest OS. In a 2 vCPU Guest, if RSS has been properly configured, you should see both vCPU utilized as depicted in Figure 20

**Figure 20.** If RSS is in Place Both vCPU Are Utilized

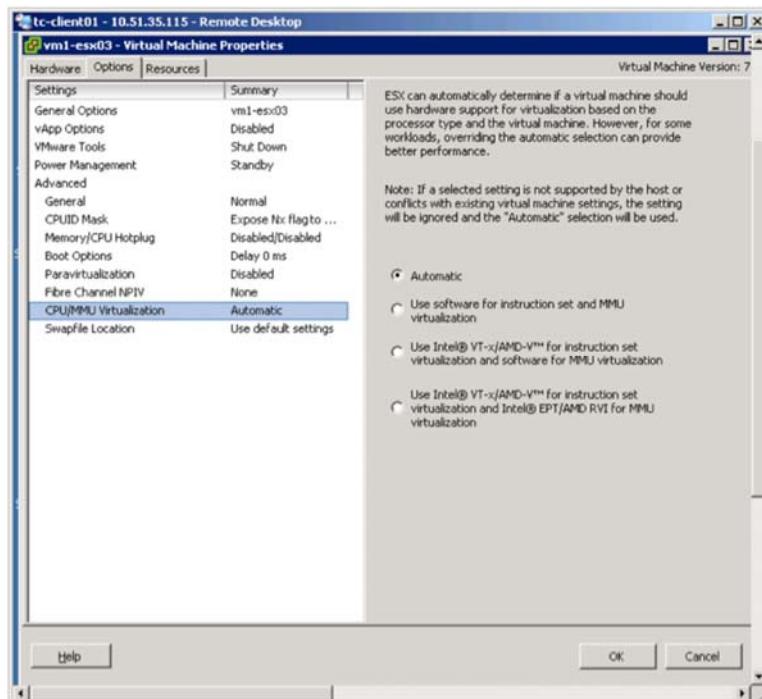


## Memory Management

Memory management configuration is an important factor in performance tuning as described in this document: <http://communities.vmware.com/docs/DOC-9882>.

If you want to force VMware vSphere 4 to use the processor's hardware memory management features, you can choose the Options tab and select CPU/MMU Virtualization.

**Figure 21.** Memory Management Configuration



## Large Memory Pages

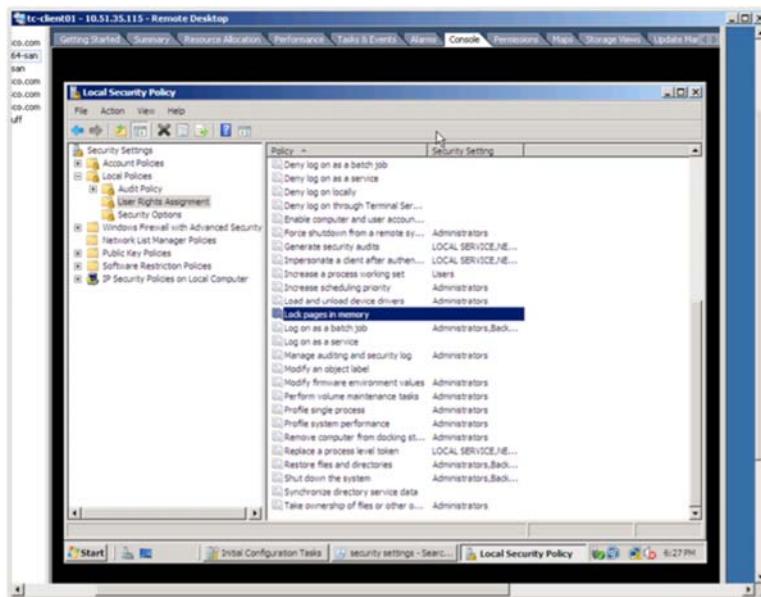
Large memory pages can make a significant difference in performance. For this reason, you should enable this feature (which requires configuration at the guest OS level).

"In addition to the usual 4-KB memory pages, VMware ESX makes 2-MB memory pages available (commonly referred to as large pages). By default, VMware ESX assigns these 2-MB machine memory pages to guest operating systems that request them, giving the guest OS the full advantage of using large pages. The use of large pages reduces memory-management overhead and can therefore increase hypervisor performance". (For more information, see <http://communities.vmware.com/docs/DOC-6912>.)

To configure large memory pages in Microsoft Windows 2008, follow these steps:

1. Assign the "Lock pages in memory" privilege to any user who runs your application. This includes administrators (Figure 22).
2. Choose Control Panel > Administrative Tools > Local Security Policy.
3. Choose Local Policies > User Rights Assignment.
4. Double-click "Lock pages in memory" and then add users and groups (Figure 22).
5. Reboot the machine.

**Figure 22.** Configuring Large Memory Pages in a Guest OS Running Microsoft Windows 2008



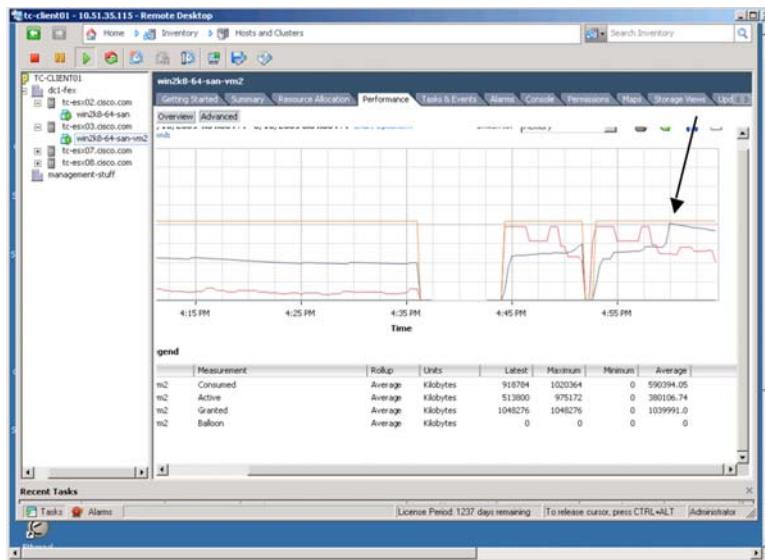
Concurrently with enabling large memory pages in the OS, set VMware ESX to handle large memory pages. Set the **Mem.AllocGuestLargePage** option to 1 to enable the use of guest large pages with host large pages. This setting reduces the number of transmit-load balancing misses and improves performance in server workloads that use guest large pages. (Set 0 to disable this option; 1 [enabled] is set by default.)

For more information about large memory pages, see <http://communities.vmware.com/docs/DOC-6912>.

#### Removing the Memory Bottleneck

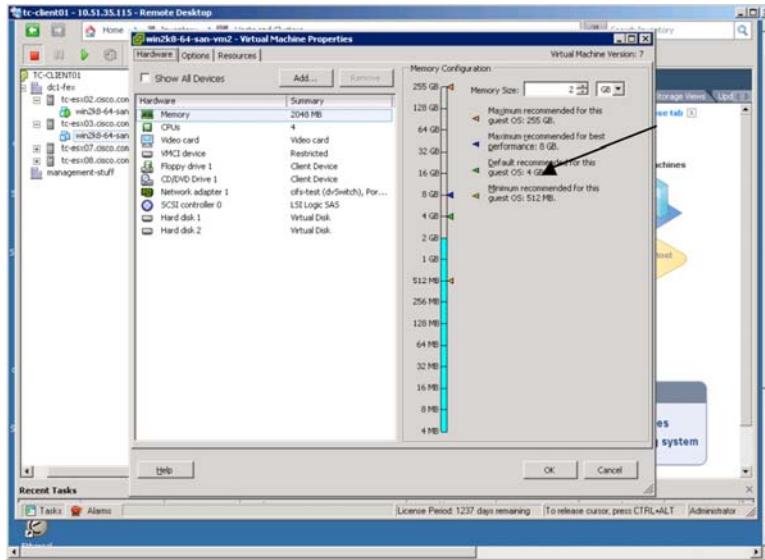
While running performance tests, make sure the performance that you are measuring is not memory bound. To do so, you should monitor the memory utilization on the guest machine as shown in Figure 23. To improve performance, you may need to allocate larger buffers and possibly run multiple instances of the testing tool, each with a large buffer allocation. While this should not account for more than a few megabytes (for example, 10 times 384 KB) of additional memory utilization, this setup is desirable to make sure that the bottleneck in the test is not the memory itself.

**Figure 23.** Monitoring a Virtual Machine Resource Usage Can Reveal a Setup That is Memory Bound



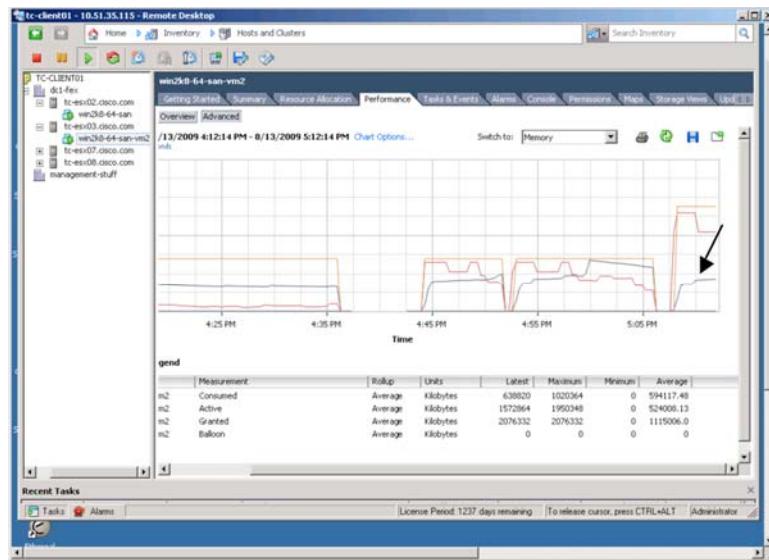
In the guest configuration, you may notice that VMware ESX advises setting the memory to approximately 4 GB as shown in Figure 24.

**Figure 24.** VMWARE vCenter Advises on the Amount of Memory that a Guest OS Needs



After you set the memory, the performance graph indicates that the setup is working better (Figure 25).

**Figure 25.** Monitoring the Virtual Machine Performance After Increasing Memory



## Tuning the Guest OS

One of the initial testing steps should be to measure the socket-to-socket application latency with tests (such as ping) to accurately measure the latency from virtual machine to virtual machine.

### Microsoft Windows 2008

Microsoft Windows 2008 includes a new TCP/IP stack; it supports SMB 2.0 and an improved firewall and other features that are outside the scope of this document. It supports TCP chimney to completely offload the TCP processing to the network adapter (not supported by the VMware ESX VMXNET adapters).

With 10 Gigabit Ethernet adapters on a VMware ESX host, the Microsoft Windows 2008 guest OS should be configured to use the VMXNET3 adapter, which appears as a 10 Gigabit Ethernet vNIC.

Microsoft Windows 2008 provides significant performance improvements over Microsoft Windows 2003, so it is highly recommended as a guest OS for virtual machine networking performance testing.

#### Tuning the TCP Stack

When using Microsoft Windows, you can use the VMXNET3 offloads for TCP segmentation, which should be enabled by default. However, in normal setups TCP chimney offload should not be enabled.

RSS is not enabled by default, so you should enable it at the VMXNET3 level as well as the OS level. To enable RSS at the OS level, you can use the **netsh** CLI command as follows:

```
netsh int tcp set global rss=enabled
```

Microsoft Windows 2008 supports TCP autotuning, which is performed per application (that is, per socket). Autotuning adjusts the TCP window based on RTT.

Autotuning can be set using the CLI at different levels:

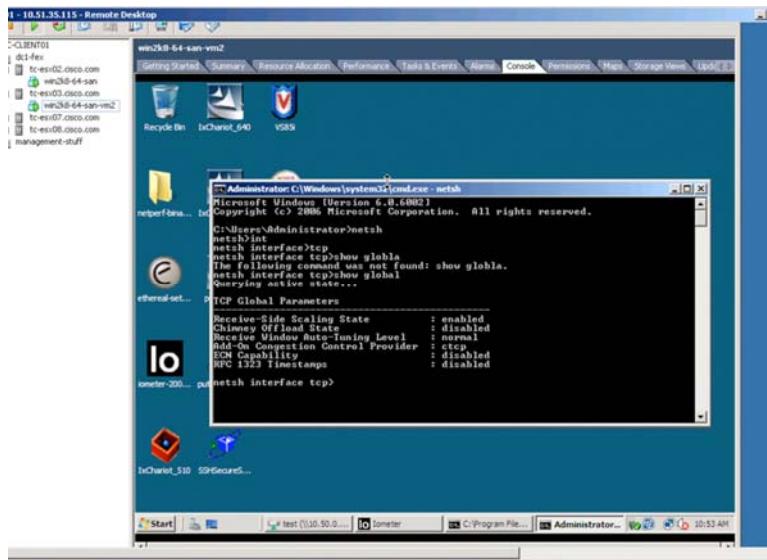
- Disabled: RFC1323 scaling is disabled
- Highly restricted: Scaling is limited to 256 KB
- Restricted: Scaling is limited to 1 MB
- Normal (default): Scaling is limited to 16 MB
- Experimental: Scaling is unlimited (1 GB)

Some applications may alter the TCP tuning automatically by selecting a more restrictive tuning.

Microsoft Windows 2008 also performs compound TCP by default, which increases the send window more aggressively.

Figure 26 shows the default TCP configuration.

**Figure 26.** Default TCP Configuration



In 10 Gigabit environments and running Microsoft Windows 2008 as a guest OS in a VMware vSphere 4 environment, the default settings (normal and no timestamps) are normally sufficient.

Certain applications such as Microsoft Internet Explorer (IE) and Mozilla Firefox are always going to advertise WS=2, so they are not the best choices for performance testing.

The setting for TCP chimney in Microsoft Windows 2008 by default is automatic, which means that TCP is offloaded if the connection is 10 Gigabit Ethernet, the RTT is less than 20 milliseconds (ms), and the connection has exchanged at least 130 KB of data. Since VMXNET3 does not support TCP chimney, you may want to explicitly disable it:

**netsh int tcp set global chimney=disabled**

#### Socket Buffers

Even if the TCP window scaling in Microsoft Windows 2008 does not require any tuning, you should remember that the maximum window size that the server and client advertises also depends on the socket buffer allocation.

As noted by Microsoft (see <http://msdn.microsoft.com/en-us/library/ms819736.aspx>), the default receive window size that TCP advertises in Microsoft Windows Server 2003 depends on the following, in order of precedence:

- Value for the **SO\_RCVBUF** Microsoft Windows Sockets option for the connection; this option specifies the total per-socket buffer space reserved for receive operations
- Per-interface **TcpWindowSize** registry value
- **GlobalMaxTcpWindowSize** registry value

There is no global socket buffer tuning. Each application sets the size of **SO\_RCVBUF** by using the Microsoft Windows Sockets function **setsockopt**.

**SO\_RCVBUF** specifies the total per-socket buffer space reserved for receive operations. This value is unrelated to **SO\_MAX\_MSG\_SIZE** and does not necessarily correspond to the size of the TCP receive window.

**SO\_SNDBUF** specifies the total per-socket buffer space reserved for transmissions. This value is unrelated to **SO\_MAX\_MSG\_SIZE** and does not necessarily correspond to the size of a TCP send window.

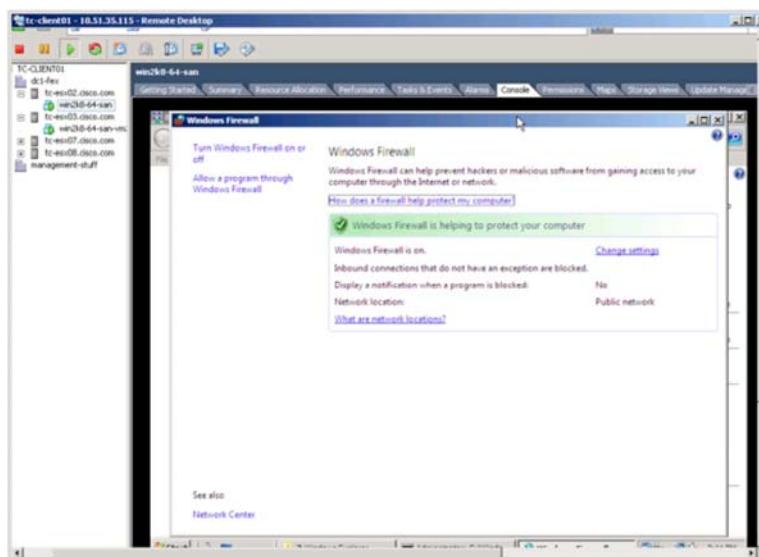
Whether the socket buffer can be tuned and the buffer's value depend on the application, which is why for network performance tests, you should use tools designed for this purpose: that is, tools that allow you to specify the message size and socket size (Netperf is an example of such a tool).

Prior to Microsoft Windows 2008, it was possible to change the default socket size entries at the registry level: **CurrenControlSet/Services/AFD/Params** (see [http://technet.microsoft.com/en-us/library/cc781532\(WS.10\).aspx](http://technet.microsoft.com/en-us/library/cc781532(WS.10).aspx)).

#### Microsoft Windows Firewall

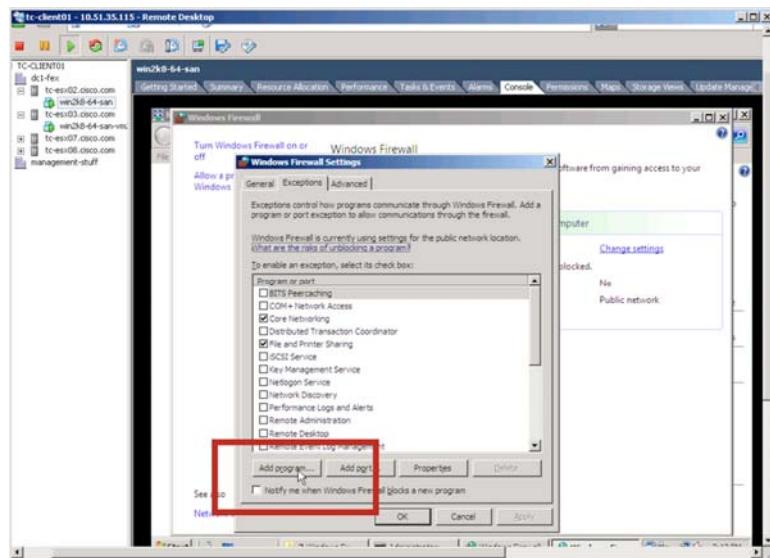
At the time of this writing, disabling Microsoft Windows Firewall to perform performance tests caused other applications not to work. Thus, if you plan to test with a Microsoft Windows 2008 guest OS, make sure you add the program to the list of allowed applications rather than disabling the firewall altogether (Figure 27).

**Figure 27.** Windows Firewall Configuration

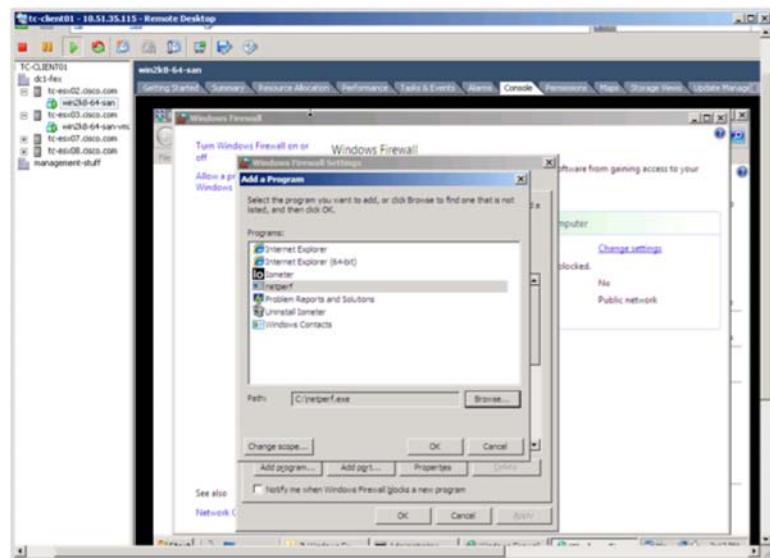


In this example, to allow Netperf to run, click Add Program (Figure 28) and select netperf (Figure 29).

**Figure 28.** Changing the Windows Firewall Configuration to Allow Netperf (Step 1)



**Figure 29.** Changing the Windows Firewall Configuration to Allow Netperf (Step 2)



## Linux

Several webpages offer tips on how to tune the Linux TCP stack.

By default, TCP window scaling is enabled: **tcpwindoscaling** is set to 1 by default under **/proc/sys/net/ipv4**

Security Enhanced Linux (SELinux, <http://www.redhat.com/docs/manuals/enterprise/RHEL-4-Manual/selinux-guide/selg-preface-0011.html>) should be disabled by editing **/etc/selinux/config** and changing the **SELINUX** line to **SELINUX=disabled**.

Myrinet provides guidelines on how to tune Linux for 10 Gigabit Ethernet performance (<http://www.myri.com/serve/cache/636.html>):

**net.core.rmem\_max = 16777216**

```
net.core.wmem_max = 16777216
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
net.core.netdev_max_backlog = 250000
```

These settings can be configured in the **/etc/sysctl.conf** file, or you can configure them in the **/proc/sys/net/core** or **/proc/sys/net/ipv4** file system via `sysctl -p /etc/sysctl.conf`

### Tuning the Client Side

The preceding sections assume that the tuning is performed on the guest OS. This is typically the server side of the test. The client side needs to be tuned as well.

Given the disparity between the transmit and receive performance of the same hardware, more devices with the same hardware need to be used as clients.

Also, if the client does not run Microsoft Windows 2008 or Linux, it needs to be manually tuned for the TCP window-scale option using the Scalable Networking Pack and making the associated registry modifications.

## Measuring Network Performance in Virtual Machines

### Preliminary Steps

This list summarizes the steps described in the previous sections of this document. Make sure these steps have been completed before proceeding with performance tests of virtual machine-to-virtual machine performance:

- Make sure that you have readily available the information about the memory and processors of the VMware ESX host.
- Verify that the 10 Gigabit Ethernet cards are placed in the correct PCIe slots.
- Verify that there is no interference from IRQ sharing with other devices.
- Verify the BIOS configuration.
- Select VMXNET3 as the network adapter for the guest OS on the server side (and client too if the client is also a virtual machine).
- Install VMWare Tools in the guest OS.
- Consider adding more than one vCPU and using RSS in Microsoft Windows 2008.
- Preferably, use Microsoft Windows 2008 rather than Microsoft Windows 2003 as the guest OS.
- With Microsoft Windows 2008 as the guest OS, the normal settings are normally sufficient, but do not forget that even if the TCP window-scaling option is enabled, a small socket buffer will prevent the window from opening enough to achieve optimal performance.
- If you are using Linux, consider tuning it for 10 Gigabit Ethernet performance.
- Make sure that TSO is enabled and is working. To verify that TSO is working, you may want to use Wireshark. If the packet size going out of the vNIC is larger than the standard MTU, TSO is likely working.
- If you are running a test with multiple virtual machines on a single VMware ESX host, consider using Intel VMDq and NetQueue if the network adapter supports this.
- Make sure each guest OS has enough memory.
- Consider enabling jumbo frames on the vDS and vSwitch and in the guest OS for better performance.
- Calculate the BDP for the test Microsoft Windows 2008 does not need any tuning at the TCP level, but the socket buffer needs to be sized accordingly.

- Make sure that both the client and server are capable of TCP window scaling and, if necessary, tune them.
- When using testing tools, make sure to properly size the socket buffer of both the client and server based on the calculations of the BDP.

One of the initial testing steps should be to measure the socket-to-socket application latency with some tests (such as ping) to accurately measure the latency from virtual machine to virtual machine to verify that the regular TCP tuning on the guest OS is sufficient for the targeted performance from an individual virtual machine (which normally is the case).

VMware benchmarking guidelines can be found at <http://www.vmware.com/resources/techresources/1061>.

### Reference Performance

Before running the tests, review some performance reports from VMware to see what you could potentially achieve. A good starting point is the following blog: <http://blogs.vmware.com/performance/>.

According to the document at [http://www.vmware.com/pdf/10GigE\\_performance.pdf](http://www.vmware.com/pdf/10GigE_performance.pdf), a reference system consisting of 16 cores could achieve the following performance:

- A single vCPU virtual machine can forward 8 Gbps of traffic on the transmit path and 4 Gbps traffic on the receive path when using standard MTU (1500-byte) frames
- Using jumbo frames, a single virtual machine can saturate a 10-Gbps link on the transmit path and can receive network traffic at rates up to 5.7 Gbps

### Performance with a Legacy Four Cores Machine

The following performance numbers are not intended to represent the best hardware configuration, they are just an example of which performance level can be achieved with a four cores system and highlight the differences between TX and RX behavior and how single-virtual machine-to-single-virtual machine performance behaves in such an environment.

In the test bed running multiple virtual machines per VMware ESX host on an HP DL380 G5 with 8 GB and a two-socket, dual-core processor, the receive performance of this system peaks at approximately 4.5 Gbps. In the transmit direction, the system running four virtual machines can generate close to 9 Gbps.

However, virtual machine-to-virtual machine performance across two VMware ESX hosts (HP DL380 G5, each with 8 GB of RAM and two-socket, dual-core processors) could achieve up to approximately 3.5 Gbps in the following configuration:

- Microsoft Windows 2008 guest OS configured
- One vCPU in the transmit direction and two vCPUs in the receive direction with RSS
- 4 GB of RAM per virtual machine
- Intel 82598 chip set
- Netperf configured for a message size of 64 KB and a socket size of 384 KB
- TSO enabled

Such a test normally requires multiple Netperf sessions.

In the best-case scenarios with the hardware listed here (HP DL-380 with four cores), tests involving disk I/O would yield much lower performance numbers for virtual machine-to-virtual machine tests (which, unless caching is present, will not exceed the disk I/O performance):

- The FTP client on Microsoft Windows 2008 lets you specify the socket buffer, and you can achieve up to approximately 500 Mbps in best cases

- FTP between a Microsoft Windows 2008 client and Linux FTP server with jumbo frames can achieve approximately 928 Mbps
- With SMB2, you may achieve a maximum of approximately 400 to 600 Mbps
- With NFS and the User Datagram Protocol (UDP), you can achieve up to 941 Mbps

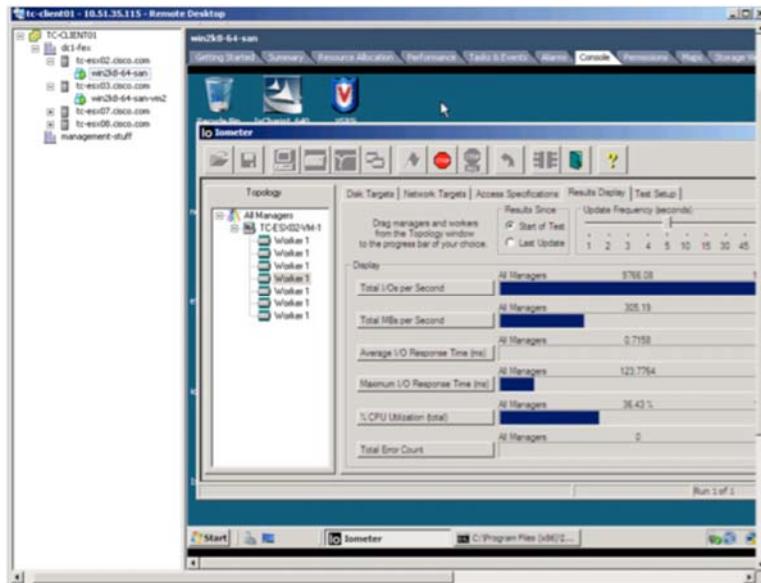
These preliminary numbers reveal the following:

- If the purpose of the test is to measure the maximum virtual machine networking performance, the best way to test is by using a memory-based tool such as Netperf
- Such a tool also is suitable because it allows you to specify the socket buffer size
- A test involving disk I/O with a regular application such as FTP, SMB, or NFS is throttled by the disk I/O itself as well as the application-specific socket buffer size
- Before running any additional tests, you should have an idea of the maximum disk I/O that the VMware ESX host system can handle
- Unless the number of clients is greater than the number of servers, the test results will provide the lowest common denominator between the transmit and receive performance of a given system
- If the goal is to prove that a virtualized server can take advantage of 10 Gigabit Ethernet network performance, the right way to test this is by running multiple virtual machines, not by measuring the performance of one virtual machine to one virtual machine

### Measure Disk I/O

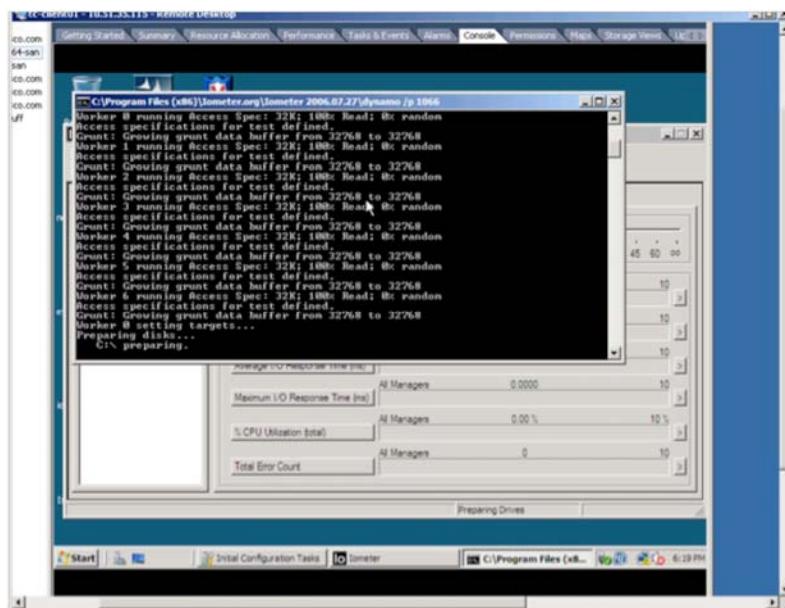
Figure 30 shows a measurement of 100 percent read operations for a local disk. The result is a maximum of 300 MB/s (which is 2.4 Gbps). This measurement sets the maximum achievable performance for any tests that use disk I/O.

**Figure 30.** Baseline the Disk I/O Performance



With Iometer, you can benchmark the local disk performance as well as the performance of remote disks, whether they are SMB2, NFS, or Fibre Channel disks. In the first phase of the test, Iometer creates patterns on the disk that is going to be tested. These patterns are then read during the test (Figure 31).

**Figure 31.** Iometer Preparing the Disk for the Test



Iometer gives you a list of disks. When a disk has not been primed yet, it displays a tiny red slash (/). This red/indicates that the first time that you launch the test, Iometer will write a large file on the disk, which will be used later in the test.

Iometer lets you specify the ratio of read operations to write operations as well as the message size to be used for the test. For a maximum-throughput test, the read-to-write ratio should be 100 percent read operations and zero write operations.

The real-time window provides real-time information about the IOPS as well as the throughput that is being measured.

To achieve the highest performance, you have to create multiple workers. In Figure 30, seven workers achieved the maximum 300-MB/s throughput. Adding workers did not improve the final results.

### Monitoring Performance

The reference publications for the tools that can be used to monitor the performance are:

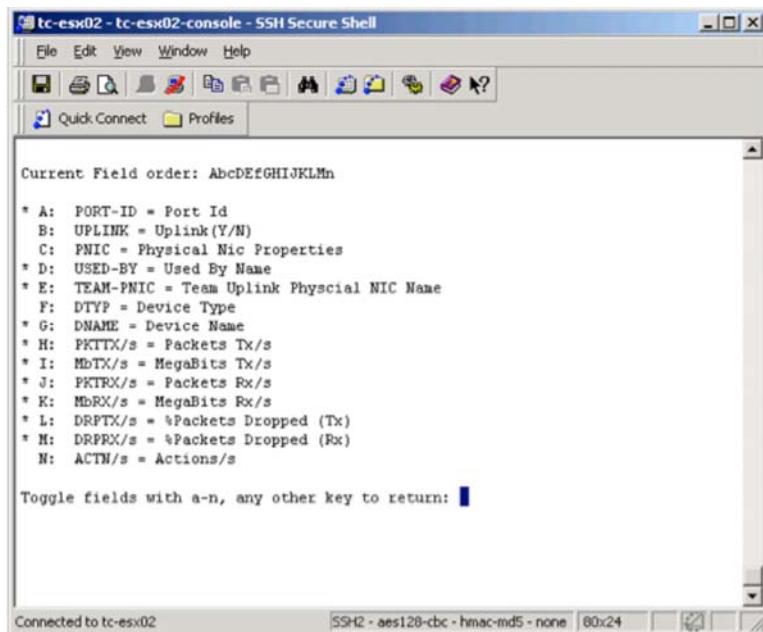
- [http://www.vmware.com/pdf/vi3\\_301\\_201\\_resource\\_mgmt.pdf](http://www.vmware.com/pdf/vi3_301_201_resource_mgmt.pdf)
- <http://www.vmware.com/resources/techresources/1061>

The following tools offer these capabilities:

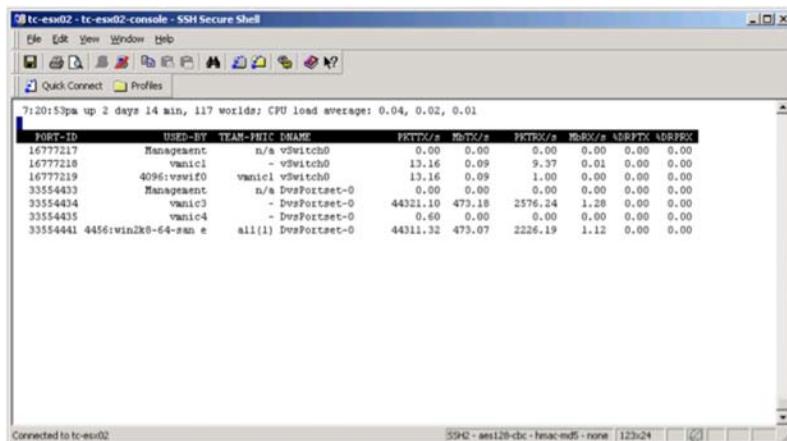
- **VMware esxtop:** Allows you to monitor CPU and core utilization from the VMware ESX console
- **perfmon:** Allows you to monitor performance from within the guest OS
- **vsish:** Allows you to monitor port statistics

### VMware esxtop

You launch the VMware esxtop tool from the service console CLI. It displays the options shown in Figure 32 (the \* indicates what is displayed at run time).

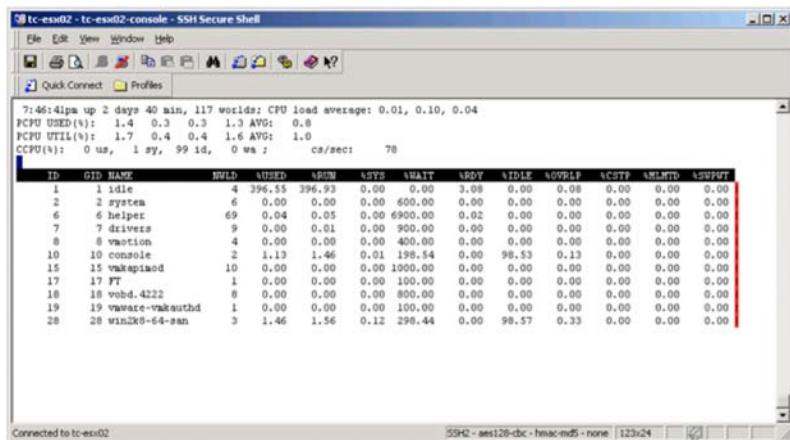
**Figure 32.** Using esxtop

The VMware esxtop allows you to monitor CPU utilization percentage, traffic on the individual vmnics, and traffic to the vNICs as shown in Figure 33 (this view is toggled by pressing the N key)

**Figure 33.** Network Performance Monitoring with esxtop

This example shows one virtual machine running on the VMware ESX host, win2k8-64-san. There also are several vmnics, and the ones that are relevant for the test are associated with the vDS: vmnic3 and vmnic4. You can also see that most of the traffic is in the transmit direction from this VMware ESX host on vmnic3.

To monitor the core utilization, you can toggle the display to the screen shown in Figure 34, where PCPU USED shows the four cores and their utilization.

**Figure 34.** esxtop View of the Cores Utilization

### NIC Statistics

NIC statistics can also be monitored from the service console, by using the **ethtool** command. Enter **esxcfg-nics -l**.

You can then query the statistics using **ethtool**:

**ethtool -S vmnic<number>**

### Increasing the Throughput

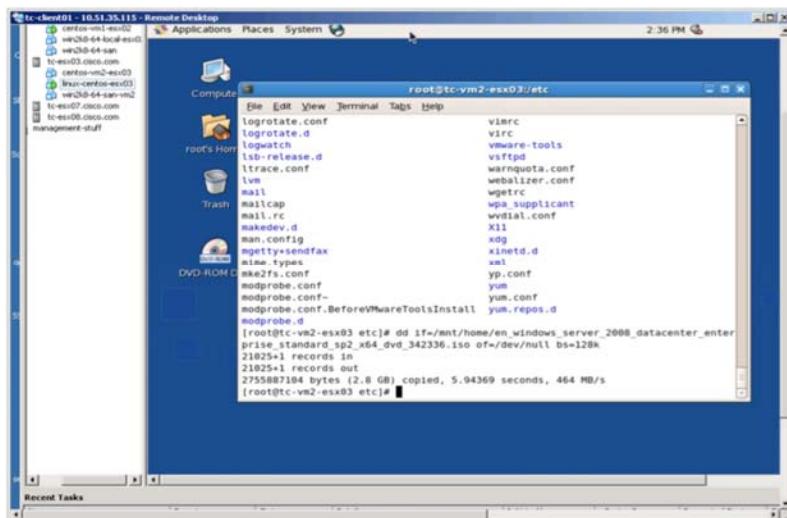
Testing from Memory Compared to Testing from Disk

Make sure that the applications allow tuning the socket buffer. Make sure to use files that are large enough for the TCP stack to properly autotune and achieve the maximum performance. Make sure to measure the TCP BDP for proper socket buffer allocation.

In the setup here for virtual machine-to-virtual machine tests using Netperf, the best results were achieved with a message size of 64 KB and a socket size of 384 KB.

Tests involving disk access (even if just for read operations) could not achieve better than 1 Gbps of throughput with NFS and a real copy operation from one virtual machine to another virtual machine.

With the dd application, a copy operation to the bit bucket (**/dev/null**) achieved greater than 1-Gbps performance as shown in Figure 35. This test is in between a pure memory test and a full disk I/O test.

**Figure 35.** Using dd

Alternatively, you could run tests with dd from a ramdisk to the bit bucket.

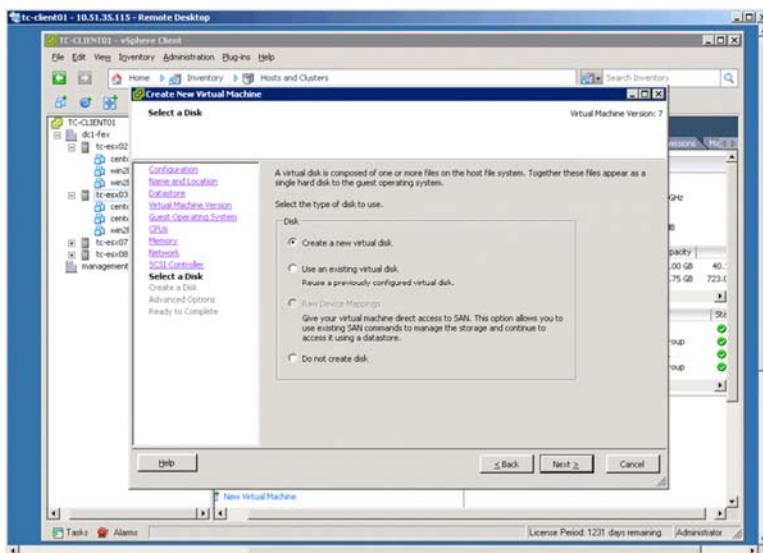
#### Running Multiple Sessions or Workers

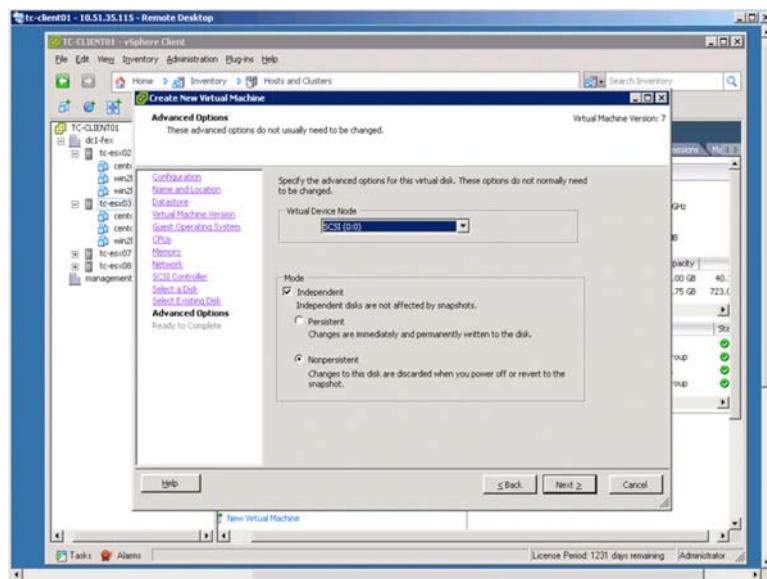
When running performance tests, you should run multiple threads or workers of the testing tool to randomize the distribution of Layer 4 ports and IP addresses. This approach in turns helps distribute the traffic to all the available cores in the system since every flow (**src\_ipaddr**, **src\_port**, **dest\_ipaddr**, and **dest\_port**) is assigned to a different vCPU.

#### Using Nonpersistent Disks and Running Multiple Virtual Machines Concurrently

If you need to run a test with multiple virtual machines, instead of creating multiple clones you can simply use nonpersistent disks. You can boot virtual machines in nonpersistent mode by adding this line to the .vmx file of each virtual machine: **scsi0:0.mode = independent-nonpersistent**.

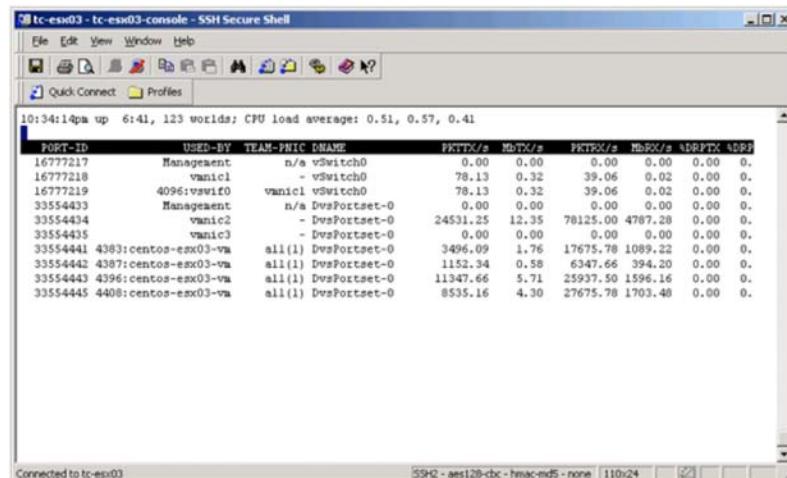
You can create a new virtual machine that uses an existing disk as shown in Figures 36 and 37.

**Figure 36.** Create a New Machine That Uses an Existing Disk

**Figure 37.** Nonpersistent Disk

This technique allows you to boot multiple virtual machines from the same disk.

With the HP DL380 G5 with four cores that was previously described, running multiple virtual machines on the receive side allows the VMware ESX server to achieve the maximum receive performance as Figure 38 shows; the traffic entering vmnic2 approaches 4.7 Gbps, distributed across the four CentOs virtual machines.

**Figure 38.** Network Performance with Multiple VMs Running from a Non-Persistent Disk

### Testing with VMware VMmark

The use of VMware VMmark is beyond the scope of this document. VMware VMmark is not specifically targeted at measuring the networking performance of a virtual machine. It runs workloads that include Java, email, and web browsing. For more information, see <http://communities.vmware.com/community/vmtn/general/performance/vmmark>.

To check published VMware VMmark test results, so to <http://www.vmware.com/products/vmmark/results.html>.

### Testing with Netperf

Netperf is a commonly used network performance measurement tool. The Netperf manual is available at <http://www.netperf.org/netperf/training/Netperf.html>.

The source code can be compiled with the Microsoft Windows Driver Kit (previously known as the Microsoft Windows Driver Development Kit). Several organizations also offer precompiled binaries; for example, see <http://itg.chem.indiana.edu/inc/wiki/software/251.html>.

A common configuration setup consists of launching the server component on one machine, netserver, which puts the machine in listening mode on port 12865:

```
netperf -H <netserver IP address> -p <netserver L4 port> -- -m <local send message size> -M <remote receive message size> -s <local send/receive socket size> -S <remote send/receive socket size>
```

If you need to generate multiple streams, you should open multiple **netserver -p <l4 port>** instances and then launch multiple Netperf sessions from the client.

It is normal to launch multiple Netperf sessions to measure the maximum throughput:

**-I <duration, default is 10s>**

With a 10 Gigabit Ethernet adapter and low-latency interconnection between servers, the best results have been achieved with **-m 64KB** and **-M 64KB** and with **-s 384KB** and **-S 384KB**.

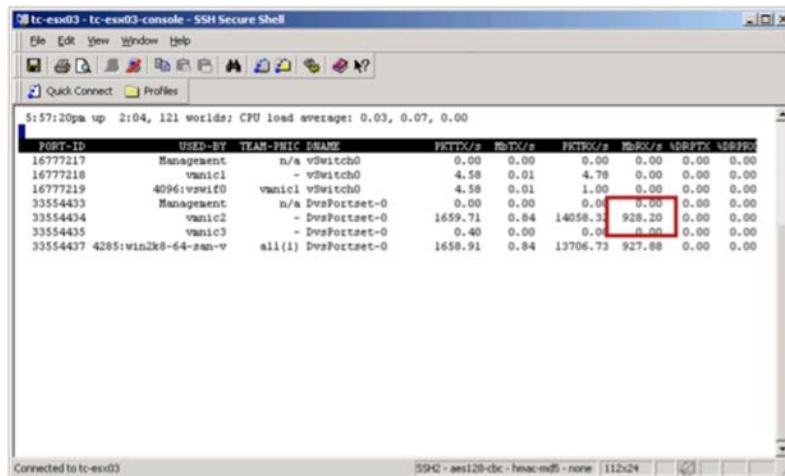
### Testing with FTP

The FTP client in Microsoft Windows 2008 and Vista allows you to set the send and receive sizes with the options **-r: 65535** and **-x: 65535**.

Thus, a good combination for testing is a Linux FTP server properly tuned and a Microsoft Windows 2008 FTP client. The FTP client can then advertise a window large enough for a high-throughput FTP transfer.

The example in Figure 45 shows a virtual machine-to-virtual machine FTP transfer with one virtual machine running CentOs as the server and one virtual machine running Microsoft Windows 2008. The virtual machine-to-virtual machine performance is close to 1 Gbps.

**Figure 39.** FTP Performance from a Virtual Machine



### Testing with SMB

SMB is not the best tool for demonstrating the performance capabilities of a 10 Gigabit Ethernet server because of several factors, including the socket buffer size and potential bandwidth throttling.

If you need to use Common Internet File System (CIFS), make sure to use SMB 2.0, which has better performance than SMB 1.0. SMB 2.0 allows large send requests to be split into multiple parallel threads. SMB 2.0 also removes a

lot of the chattiness that characterized SMB 1.0, which helps improve performance results, and it supports dynamic crediting, so that a client can have multiple outstanding requests.

Instructions about how to disable SMB 1.0 for Microsoft Windows 2008 can be found at <http://blogs.technet.com/askperf/archive/2008/05/30/two-minute-drill-overview-of-smb-2-0.aspx>.

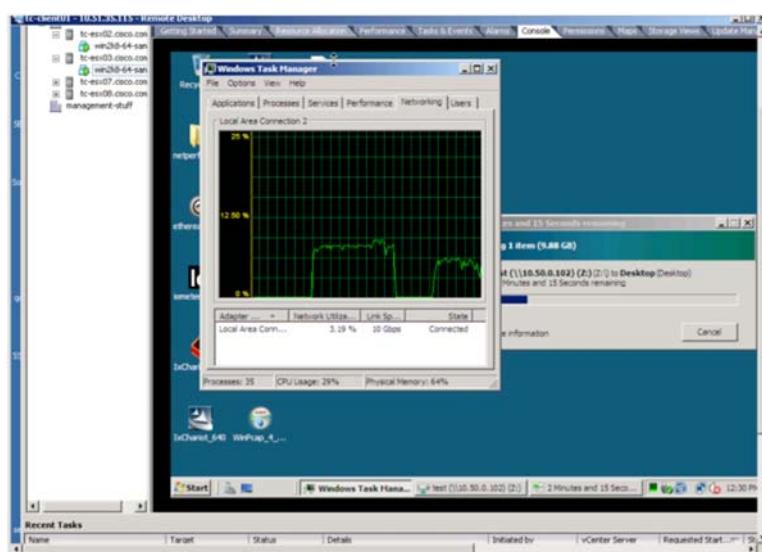
To disable SMB 1.0 on a Microsoft Windows Vista or Windows Server 2008 system that is acting as the server system (hosting the network resources), a registry modification is required. Navigate to the **HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters** key. If there is no **REG\_DWORD** value named **Smb1**, then create one. Set the value to 0 to disable SMB 1.0 or to 1 to enable SMB 1.0.

To disable bandwidth throttling, follow the instructions described at [http://social.msdn.microsoft.com/Forums/en-US/os\\_fileservices/thread/832d395b-6e6f-4658-8dbb-120138a4cd7c](http://social.msdn.microsoft.com/Forums/en-US/os_fileservices/thread/832d395b-6e6f-4658-8dbb-120138a4cd7c).

The default setting is 0. This setting is available starting with Microsoft Windows Server 2008 SP2. By default, the SMB redirector throttles throughput across high-latency network connections, in some cases to avoid network-related timeouts. Setting this registry value to 1 disables this throttling.

With SMB, the maximum virtual machine-to-virtual machine performance achieved was approximately 400 Mbps as shown in Figure 40.

**Figure 40.** SMB Performance in a Virtual Machine



### Testing with NFS

With NFS, you may want to mount a directory by specifying the read and write sizes to increase performance.

The **mount** command **rsize** and **wsize** options specify the size of the chunks of data that the client and server pass back and forth to each other. If the **rsize** and **wsize** options are not specified, the most common default is 4 KB (4096 bytes).

```
mount -o proto=tcp, rsize=65535, wsize=65535 10.50.0.112:/mnt/share /mnt/home
```

Using NFS, UDP, jumbo frames, and a socket size of 384 KB and copying by blocks of 128 KB, the maximum performance that was achieved was approximately 900 Mbps.

### Testing with DD

The dd application could be used to achieve higher network performance without involving disk access, while still using NFS. In this case, you could mount a directory with **rsize=65535** and **wsize=65535** on the client side and then enter a command like **dd if=/ram/somefile of=/dev/null** to fetch a file from the remote server ramdisk to the bit bucket.

With this procedure, you can achieve performance similar to that achieved with Netperf.

### Testing with Rsync

Rsync provides more options to control the buffer size. For more information, see <http://www.samba.org/rsync/documentation.html>.

Rsync works by starting a rsync daemon and connecting to TCP port 873. A rsync configuration with a large buffer size looks like this:

```
rsync -av --sockopts=SO_SNDBUF=2000000,SO_RCVBUF=2000000 host::module
```



Americas Headquarters  
Cisco Systems, Inc.  
San Jose, CA

Asia Pacific Headquarters  
Cisco Systems (USA) Pte. Ltd.  
Singapore

Europe Headquarters  
Cisco Systems International BV  
Amsterdam, The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at [www.cisco.com/go/offices](http://www.cisco.com/go/offices).

Cisco and the Cisco Logo are trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and other countries. A listing of Cisco's trademarks can be found at [www.cisco.com/go/trademarks](http://www.cisco.com/go/trademarks). Third party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1005R)