10Gbps Networking Performance

VMware® ESX 3.5 Update 1

With increasing number of CPU cores in today's computers and with high consolidation ratios combined with the high bandwidth requirements of today's applications, the total I/O load on a server is substantial. Single 1Gbps network cards are unable to support the demands of these applications, and multiple NICs are often impractical because the number of ports utilized on the host and on the network switches. Now 10Gbps Ethernet adapters offer a solution that provides much higher bandwidth while using fewer ports on the hosts and switches.

Support for 10Gbps networking devices was introduced in ESX 3.5. Features such as TCP segmentation offload (TSO), jumbo frames, and multiple receive queues were introduced to efficiently use 10Gbps devices. ESX 3.5 Update 1 added support for PCI-E–based Intel Oplin 10Gbps Ethernet adapters to ESX.

In this paper, we present the following results:

- A single one-vCPU virtual machine can drive 8Gbps of traffic on the send path and 4Gbps traffic on the receive path when using standard MTU (1500 byte) frames.
- Using jumbo frames, a single virtual machine can saturate a 10Gbps link on the transmit path and can receive network traffic at rates up to 5.7Gbps.
- Both transmit and receive performance scale very well with increasing load on the ESX host.
- Even on hosts with high consolidation ratios, all virtual machines get a fair share of the available receive and transmit bandwidth.

This paper covers the following topics:

Performance Enhancements in ESX 3.5

A number of enhancements have been integrated into the networking code of ESX 3.5, including support for jumbo frames and TSO for Enhanced VMXNET devices. Jumbo frames are large Ethernet frames, typically 9000 bytes, which are larger than the standard Ethernet MTU (maximum transfer unit) of 1500 bytes. Guest operating systems using jumbo frames need fewer packets to transfer large amounts of data and can achieve higher throughputs and lower CPU utilization than guests using packets with the standard MTU size.

TSO is a feature, widely supported by today's network cards, that allows the CPU-intensive task of segmenting large TCP packets (up to 64KB) to be offloaded to the hardware. Because the guest operating system can now send packets larger than the standard MTU to ESX, the processing overheads on the transmit path are greatly reduced.

ESX 3.5 also introduced support for NetQueue. The NetQueue API allows a NIC driver to have multiple queues for processing incoming packets on different physical CPU cores. Hence, on a machine with MSI-X support, the device can use multiple physical CPU cores for processing packets destined for different virtual machines on the host. This boosts the networking performance on the receive path (which traditionally has higher costs than the transmit path) when multiple virtual machines are running on the same host.

Benchmarking Methodology

We used the network benchmarking tool netperf 2.4.2 for all the tests. Netperf measures unidirectional network performance for TCP and UDP traffic. It includes support to measure TCP and UDP throughput, using bulk transfers, and end-to-end latencies. Netperf has a client-server model and comprises the following:

- Netperf client, which acts as a data sender
- Netserver process, which acts as a receiver

To check for network performance under different configurations, netperf allows you to specify parameters, such as the socket size and the message size, for the tests. For more details on netperf, see the Netperf Manual. For a link to the manual, see "Resources" on page 9.

Table 1. Benchmark Test Configuration

Hardware		
ESX	HP DL 580 G5	■ 4 quad-core Xeon X7350 at 2.93GHz ■ 16GB RAM ■ Intel 82598 XF 10Gbps dual-port adapter
Client machine	HP DL 380 G5	■ 2 quad-core Xeon X5355 at 2.66GHz ■ 16GB RAM ■ Intel 82598 XF 10Gbps dual-port adapter
Software		
ESX	ESX 3.5 Update 1	
Client machine	Red Hat Enterprise Linux 5 Update 1 AS 64-bit kernel: 2.6.18-53.el5	
Virtual machines	■ Red Hat Enterprise Linux 5 Update 1 AS 64-bit kernel: 2.6.18-53.el5 ■ Windows Server 2003 SP2 Enterprise Edition 64-bit	■ 1 virtual CPU ■ 512MB RAM ■ Virtual device: Enhanced VMXNET

For tests using both single and multiple virtual machines, we used five simultaneous netperf sessions per virtual machine. Multiple netperf sessions are needed, because a single session is unable to saturate a 10Gbps link even when ample CPU cycles are available.

The details of the experimental configuration are presented in Table 1. All virtual machines used for the experiments used a uniprocessor kernel or HAL and were configured with one virtual CPU. All virtual machines were configured with 512MB of RAM and used the Enhanced VMXNET virtual device. To understand the performance difference between e1000 and vmxnet virtual devices, see "Performance Comparison of Virtual Network Devices." For a link, see "Resources" on page 9.

TSO was enabled by default in the virtual machines as well as on the client machine. To enable jumbo frames, the MTU was set to 9000 in the following places:

- In the client, using `ifconfig eth1 mtu 9000`
- On the ESX host, using `esxcfg-vswitch -m 9000 vSwitch1`
- In the Windows virtual machine, by setting **Device Manager** > **Network adapters** > **VMware PCI Ethernet Adapter** > **Properties** > **Advanced** > **MTU** to 9000
- In the Linux virtual machine, using `ifconfig eth1 mtu 9000`

NOTE It is important that there is no mismatch of MTUs along a given link.

For tests using a single virtual machine, for minimal variance in the experiments, both the virtual machine and physical NIC interrupt vectors were pinned to two adjoining cores on the same CPU. We pinned the virtual machine to physical CPU 3 using VI Client (**VM** > **Edit Settings** > **Resources** > **Advanced CPU** > **Scheduling affinity**). We pinned the Oplin interrupt vector (`0xb1` in our case) to the adjoining core by running the following command in ESX:

```
echo "move 0xb1 2" > /proc/vmware/intr-tracker
```

For the tests using multiple virtual machines, instead of creating 15 clones of the original virtual machines, we used nonpersistent disks. We booted the virtual machines in nonpersistent mode by adding the following line to the `.vmx` file for each virtual machine:

```
scsi0:0.mode = independent-nonpersistent
```

In this way, we could boot multiple virtual machines from the same virtual disk. All changes to such a virtual machine are lost after the virtual machine powers down.

Because 16 virtual machines sending a large number of small packets can easily overflow the default queues on the host, for tests using multiple virtual machines, we increased the **Pending Tx Packet** queue size from the default value of 200 to 500 using VI Client (**Configuration** > **Advanced Settings** > **Net** > **Net.MaxNetifTxQueueLen**).

For the scaling tests using multiple virtual machines, we enabled NetQueue using VI Client (**Configuration** > **Advanced Settings** > **VMkernel** > **VMkernel.Boot.netNetqueueEnabled**).

We configured the Intel Oplin driver (`ixgbe`) to be loaded with MSI-X based interrupt and 16 receive queues on the test interface and a single queue on the second interface of the dual-port NIC by using the following command:

```
esxcfg-module -s "InterruptType=2,2 VMDQ=16,1" ixgbe
```

For the changes to come into effect, we needed to reboot the ESX host. For more details on enabling NetQueue, see "Enabling Support for NetQueue on the Intel 82598 10 Gigabit Ethernet Controller." For a link, see "Resources" on page 9.

Figure 1. Test Setup for Send Test from Virtual Machine to Native Environment

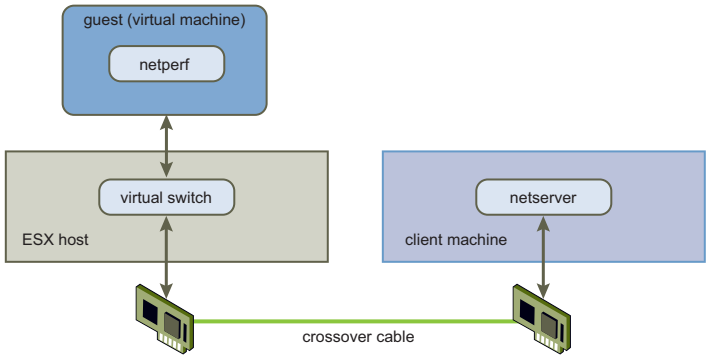


Figure 1 shows the test configuration for the send tests in which a virtual machine is sending data. For the receive tests, we swapped the netperf and netserver processes. We ran two sets of throughput tests—first without jumbo frames, and then with jumbo frames enabled. For the tests using a single virtual machine, all reported numbers are the average of three one-minute runs. For the tests using multiple virtual machines, the run lengths were increased to two minutes to compensate for the skew in starting experiments simultaneously in as many as 16 virtual machines.

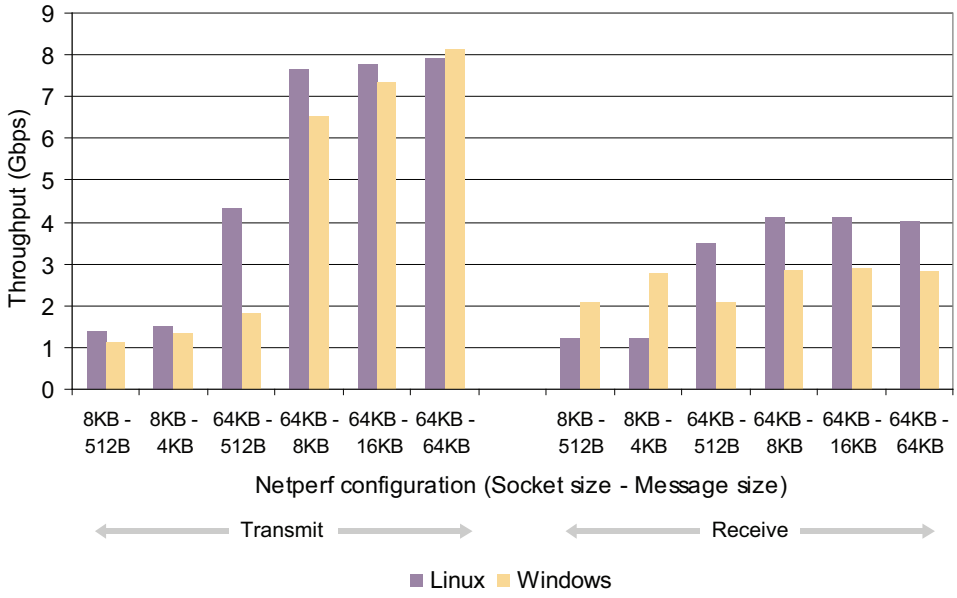
This paper does not discuss the performance when two virtual machines are running on the same ESX host. The network performance between two virtual machines on the same host is independent of physical NIC speeds and is discussed in detail in "Networking Performance in VMware ESX Server 3.5." For a link, see "Resources" on page 9.

Standard MTU Performance

Figure 2 shows the TCP throughput observed for Linux and Windows one-vCPU virtual machines transmitting and receiving without using Jumbo Frames. With either operating system, a single uniprocessor virtual machine can drive approximately 8Gbps of traffic. Using a large socket size and a large message size helps transmit performance significantly. On the receive path, Linux virtual machines can receive 4Gbps of traffic, whereas Windows virtual machines are limited to 3Gbps of traffic. This difference in performance can be attributed to the differences in the TCP/IP stacks of the two operating systems. The receive throughput is usually lower than the transmit throughput for the same network configuration, because the processing overheads on the receive path are higher than those on the transmit path.

Thus, with current hardware, a one-vCPU virtual machine can drive 8Gbps of traffic on the transmit path and 4Gbps on the receive path.

Figure 2. Single Virtual Machine Performance with 1500-byte Packets



Jumbo Frame Performance

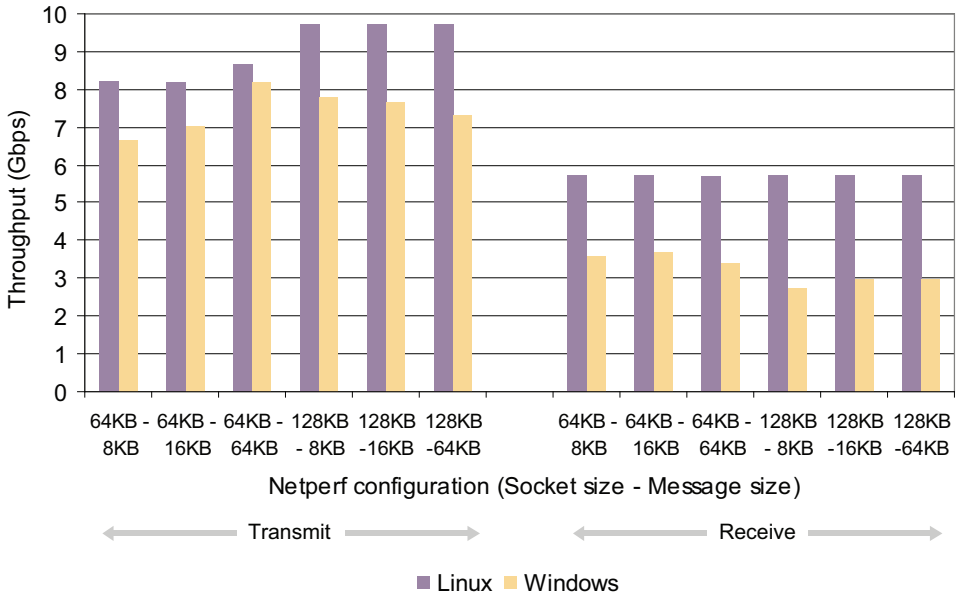
Though the transmit throughput using the standard MTU (1500 bytes) packets comes close to the native throughput, a single uniprocessor virtual machine is CPU limited and cannot saturate a 10Gbps link. Similarly, receive throughput is limited to 4Gbps. Jumbo frames can help reduce some processing overheads because large Ethernet frames result in few packets.

Figure 3 shows the throughput results for one-vCPU virtual machines using 9000 byte jumbo frames. For jumbo frame tests, we used 128KB sockets instead of 8KB sockets because most operating systems see benefits from jumbo frames when using large socket sizes. As the graphs show, with a 128KB socket, the Linux virtual machine is able to saturate the 10Gbps link, irrespective of the message sizes being used for the test. Windows throughput remains nearly constant for all the network configurations.

On the receive path, the throughput for Linux virtual machines increases by up to 40 percent over the standard MTU case. We have observed that Windows receive performance is adversely affected by socket sizes greater than 64KB, as can be seen in the graph.

Thus, with jumbo frames, a single virtual machine can saturate a 10Gbps link on the transmit path and its receive performance can improve by up to 40 percent in some cases.

Figure 3. Single Virtual Machine Performance with 9000-byte Packets



Performance Scalability with Multiple Virtual Machines

One of the primary advantages of using virtualization is the ability to consolidate multiple physical servers into multiple virtual machines running on a single physical host. These virtual machines share the host CPU and memory as well as the storage and networking devices. We assume that a large number of small virtual machines will share a single physical 10Gbps NIC. To understand the performance in such a scenario, we ran tests with as many as 16 virtual machines sharing the same physical NIC. Ideally, as virtual machines are added to the system, the throughput should increase linearly and then stabilize after a certain point (and not decrease as the load is increased on the system). Figure 4 and Figure 5 show the results from the transmit scaling and receive scaling tests, respectively. The results presented in this section simulate the worst case scenario, in which all virtual machines are simultaneously stressing the network. In real world deployments, the bandwidth requirements of most applications are low, and hence consolidation ratios will be higher.

Figure 4. Multiple Virtual Machine Transmit Performance

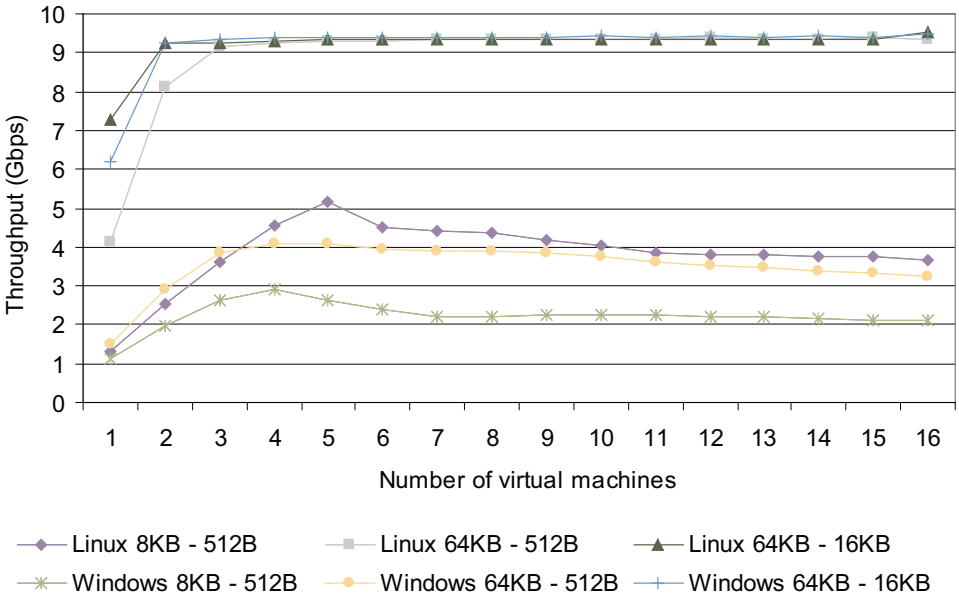


Figure 4 shows how the transmit performance scales with an increasing number of virtual machines. For both Linux and Windows, two virtual machines are enough to saturate a 10Gbps link. In the previous sections, we showed that a single virtual machine can drive up to 8Gbps of traffic. The large socket size cases show that VMware Infrastructure 3.5 scales well and that networking throughput increases until it reaches the saturation point (9.4Gbps for standard MTU) and remains constant despite increasing load on the system. In tests with small socket sizes and small message sizes, throughput increases linearly up to a few virtual machines and then stabilizes at a lower value. In such configurations, throughput is usually limited by the TCP/IP stack or by the processing power of the single client, which is receiving all traffic from multiple virtual machines. To confirm this, we ran Red Hat Enterprise Linux 5 natively on the same hardware (with 16 physical cores) and found that it can drive only approximately 6Gbps of traffic when using an 8KB socket size and 512 byte message size.

Figure 5. Multiple Virtual Machine Receive Performance

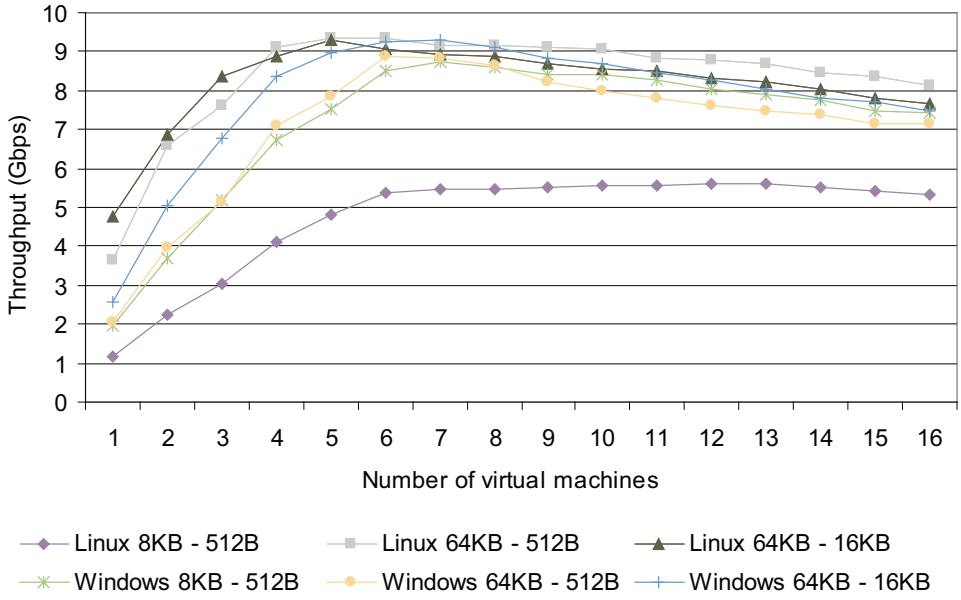
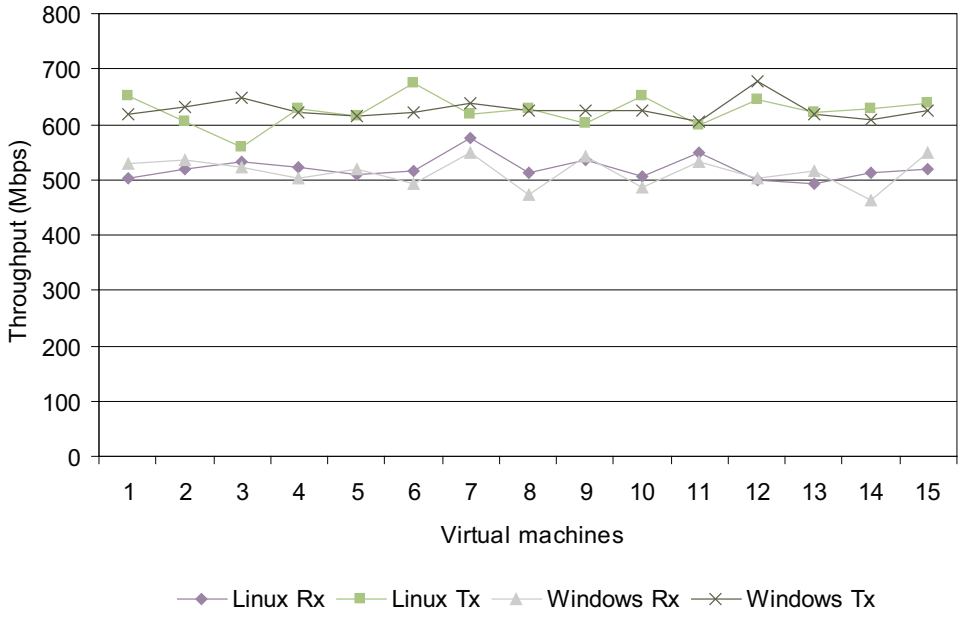


Figure 5 shows how performance scales on the receive path with an increasing number of virtual machines. As expected from the single virtual machine results, three virtual machines are enough to saturate the link in the receive case. All lines on the graph show a similar trend—the throughput increases linearly until it reaches line rates and remains constant until the load reaches seven virtual machines. As we increased to more than seven virtual machines, we saw a very gradual continuous decline as the load in increased to 16 virtual machines. The Linux small socket size and small message size case does not perform as well as the others and, as mentioned earlier, this is a result of the fact that Red Hat Enterprise Linux 5 running natively (on the client) cannot push beyond 6Gbps when using small sockets and small message sizes.

Thus, ESX scales well with an increasing number of virtual machines on the system. Our tests show the worst case scenario, in which all virtual machines are running a bandwidth-intensive application. Performance should be better in real world deployments in which it is rare that all virtual machines booted on a system are pushing traffic at the maximum rate at the same time. Virtual machines in typical customer environments usually have much lower throughput requirements and can scale to a much larger numbers of virtual machines.

Figure 6. Throughput Fairness in Test with 15 Virtual Machines



We also investigated fairness among the TCP flows across all virtual machines. Figure 6 shows the average throughput per virtual machine when using a configuration with 64KB sockets and 16KB messages. The relatively flat lines indicate that even when the system is highly loaded, all virtual machines get a fair share of the bandwidth. For example, in the Windows transmit case with 15 virtual machines, in which there are 75 TCP connections (five per virtual machines), the average throughput per virtual machine is 627Mbps and the standard deviation is a mere 17Mbps.

Conclusion

The results presented in the previous sections show that virtual machines running on ESX 3.5 Update 1 can efficiently share and saturate 10Gbps Ethernet links. A single uniprocessor virtual machine can push as much as 8Gbps of traffic with frames that use the standard MTU size and can saturate a 10Gbps link when using jumbo frames. Jumbo frames can also boost receive throughput by up to 40 percent, allowing a single virtual machine to receive traffic at rates up to 5.7Gbps.

Our detailed scaling tests show that ESX scales very well with increasing load on the system and fairly allocates bandwidth to all the booted virtual machines. Two virtual machines can easily saturate a 10Gbps link (the practical limit is 9.3Gbps for packets that use the standard MTU size because of protocol overheads), and the throughput remains constant as we add more virtual machines. Scaling on the receive path is similar, with throughput increasing linearly until we achieve line rate and then gracefully decreasing as system load and resource contention increase.

Thus, ESX 3.5 Update 1 supports the latest generation of 10Gbps NICs with minimal overheads and allows high virtual machine consolidation ratios while being fair to all virtual machines sharing the NICs and maintaining 10Gbps line rates.

Resources

- "Enabling Support for NetQueue on the Intel 82598 10 Gigabit Ethernet Controller" http://kb.vmware.com/kb/1004278
- "Performance Comparison of Virtual Network Devices" www.vmware.com/files/pdf/perf_comparison_virtual_network_devices_wp.pdf
- Netperf manual http://www.netperf.org/netperf/training/Netperf.html
- "Networking Performance: VMware ESX Server 3.5" http://www.vmware.com/files/pdf/ESX_networking_performance.pdf

If you have comments about this documentation, submit your feedback to: docfeedback@vmware.com

VMware, Inc. 3401 Hillview Ave., Palo Alto, CA 94304 www.vmware.com

Revision: 20081104 Item: PS-071-PRD-01-01