VMCI Socket Performance

VMware vSphere 4.0

The VMCI (Virtual Machine Communication Interface) device was first introduced in VMware® Workstation 6.5 and VMware Server 2.0 and is now fully supported in VMware vSphere™ 4. The VMCI device is made available to application programmers through the VMCI Sockets library. The VMCI device allows fast, efficient communication between VMs running on the same host, without using the guest networking stack.

This is our first paper on VMCI Socket performance. It presents VM-VM performance results using VMCI Sockets and compares these results to the VM-VM performance achieved using regular TCP/IP sockets. This paper is mainly intended for application developers. The main points of this paper are:

- VMCI bypasses the guest or VMkernel networking stack to allow for fast, efficient communication among VMs on the same host.
- VMCI offers a high-performance alternative to regular TCP/IP sockets. For most networking configurations, VMCI throughput exceeds TCP/IP throughput.
- The gains in networking performance are well worth the effort of porting an application to use VMCI Sockets.

Introduction

ESX/ESXi 4.0 incorporates the new VMCI device that facilitates high speed communication between virtual machines on the same host as well as between a virtual machine and the hypervisor. The VMCI device is independent of the guest networking stack. Since there is no protocol stack involved, in certain cases, the communication rates provided by this device approach the memory bandwidth of the system.

The drivers for the VMCI device are included in VMware Tools. Both Windows and Linux operating systems are supported. In order to effectively use the VMCI device, VMware provides the VMCI Sockets API. With the help of the library, developers can port their applications to use the VMCI Socket interface, instead of regular TCP/IP sockets, with minimal code modifications. For details on how to port your application or benchmark to VMCI, refer to the *VMCI Sockets Programming Guide* (see "Resources, [1]").

For applications which involve a considerable amount of inter-VM communication on the same host, VMCI provides an efficient, high performance alternative to TCP/IP sockets. Application developers develop their applications using the given API or can port their existing applications to VMCI with little effort. VMCI Sockets can outperform regular sockets in a number of cases. In this paper we compare the performance of 2 VMs communicating over VMCI with the performance of the same VMs communicating via TCP/IP while using a vmxnet3 virtual device. The results indicate that if a given application can be ported to use VMCI Sockets, it can experience a major boost in networking throughput.

For the purpose of this paper, the terms *vmxnet3 performance* and *TCP/IP performance* are used interchangeably.

Performance Methodology

We used a modified version of the network benchmarking tool netperf 2.4.2 for all the tests. Netperf measures uni-directional performance for TCP and UDP traffic. Netperf has a client-server model which is made up of the following:

- **Netperf**, which acts as the data sender
- **Netserver**, which acts as the data receiver

For measuring networking performance under different configurations, netperf allows you to specify parameters, such as socket size and message size, for the tests. For our VMCI Socket experiments, we ported netperf to use VMCI Sockets instead of regular TCP/IP sockets. The modified netperf takes one additional parameter—the context ID[1] of the VM on which the netserver was running. Because the control channel of the ported netperf still uses regular TCP/IP sockets, the VMs need to be configured with a virtual NIC and should be able to ping each other.

The details of the experimental configuration are presented in Table 1. All VMs used for the tests used a uni-processor HAL or kernel and were configured with 1 CPU. All VMs had 512MB of RAM and used a vmxnet3 virtual device for communication. No kind of manual pinning was used.

Table 1. Experimental setup

	Hardware	Software
Server	HP DL 580 G5 4 quad-core Intel Xeon X7350 @ 2.93 GHz 16 GB RAM	ESX 4.0
2 Linux VMs	Red Hat Enterprise Linux 5 Update 1 AS 64-bit	1 vCPU 512MB RAM Device: vmxnet3
2 Windows VMs	Windows Server 2003 SP2 Enterprise Edition 64-bit	

For all VM-VM TCP/IP tests, the VMs were connected to the same vSwitch. Since VMCI Sockets are independent of the TCP/IP sockets, the VMs need not be connected to the same vSwitch to be able to communicate using VMCI. However, since our version of netperf needs both TCP/IP and VMCI connectivity, we did connect the VMs to the same vSwitch for the VMCI experiments.

To allow Linux to use large socket sizes, the Linux VM was configured as follows. No configuration changes were made to the Windows VM.

```
echo 262144 > /proc/sys/net/core/wmem_max
echo 262144 > /proc/sys/net/core/rmem_max
echo “4096 87380 4194304” > /proc/sys/net/ipv4/tcp_wmem
echo “4096 87380 4194304” > /proc/sys/net/ipv4/tcp_rmem
```

The single session results are the average of three one-minute runs. For multi-session runs, 5 simultaneous netperf and netserver sessions were started in the source and destination VMs respectively. To ensure that the skew in start times of the individual sessions did not affect the final results, each iteration of the multi-session tests lasted 2 minutes.

[1] The context ID is a unique identifier for a given VMCI device. When using VMCI Sockets, the context ID serves the same role as an IP address for TCP/IP sockets.

Experimental Results

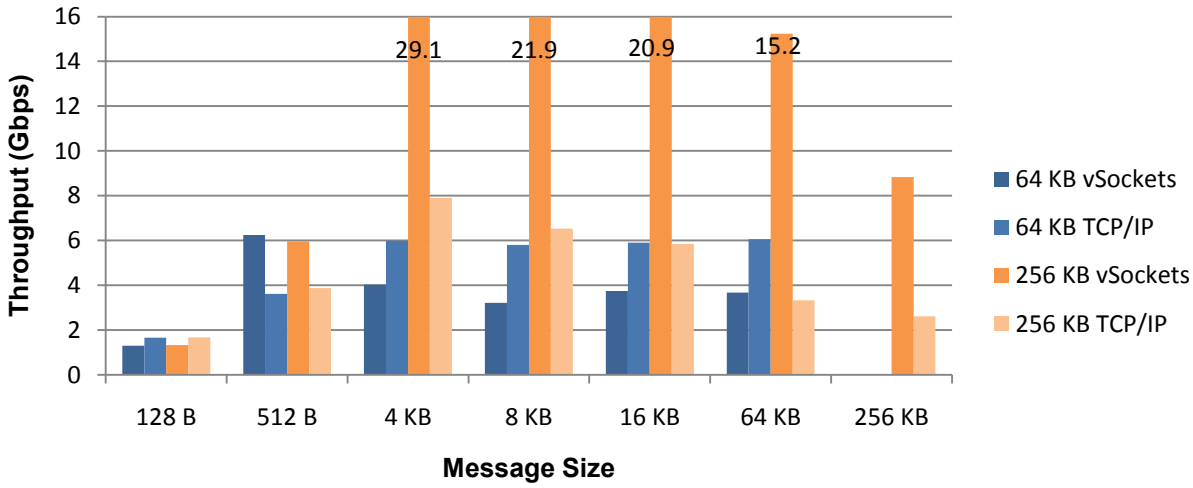
The following sections describe the outcome of our tests.

Single-Session Performance

Linux VM to Linux VM

Figure 1 presents the results of single session tests in which one Linux VM was sending data to a second Linux VM. The bars for the 256KB socket size are clipped in the graph since the throughput for VMCI Sockets was much higher than the throughput for regular sockets.

Figure 1. Linux single-session performance—256KB vSockets greatly outscore 256KB TCP/IP for message sizes between 4KB and 64KB



For 64KB socket sizes (shown in blue), regular TCP/IP socket performance is better than VMCI Socket performance across all message sizes except 512 byte messages. For 256KB socket sizes (shown in orange), VMCI Socket performance is far-superior to TCP/IP socket performance. Using VMCI, we observed up to 29Gbps throughput for 4KB socket sizes, which is more than 3 times the maximum throughput obtained using TCP/IP sockets. Message sizes between 4KB and 64KB give the best performance for VMCI Sockets.

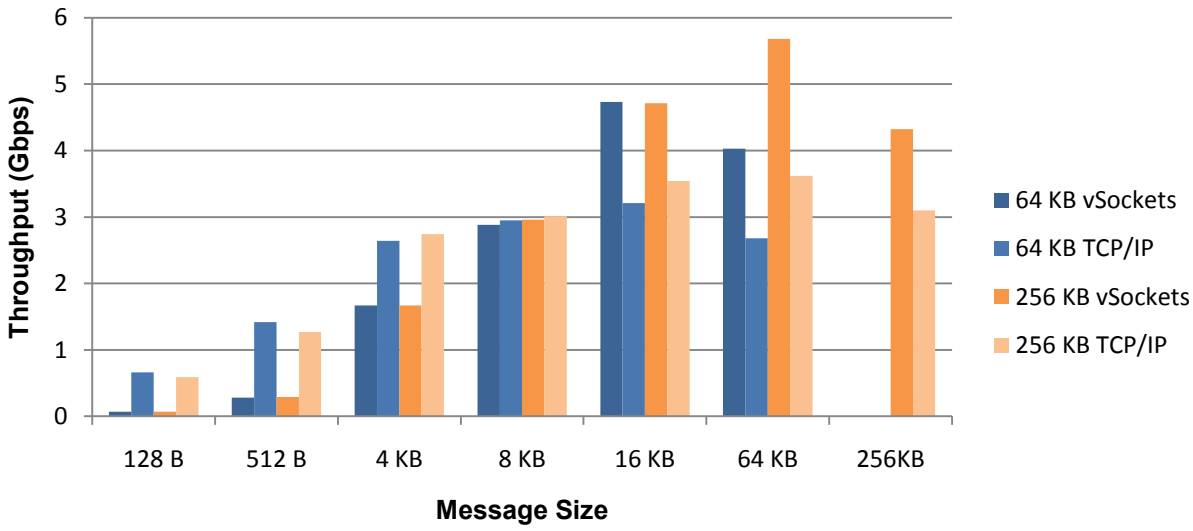
Windows VM to Windows VM

Figure 2 presents single-session results for a Windows Server 2003 VM. The trend for Windows is quite different from the trend for Linux.

First, the absolute throughput observed on Windows is much lower compared to Linux. The maximum throughput observed on Windows is just under 6Gbps, whereas on Linux the maximum throughput observed was 29Gbps.

Second, the relative performance of VMCI device and vmxnet3 depends on the message size used for the tests. For small messages (8KB and under), TCP/IP throughput is higher than the VMCI Socket throughput. However, for large message sizes, the VMCI device is better than vmxnet3.

Figure 2. Windows single-session performance—vSockets wins for large message sizes



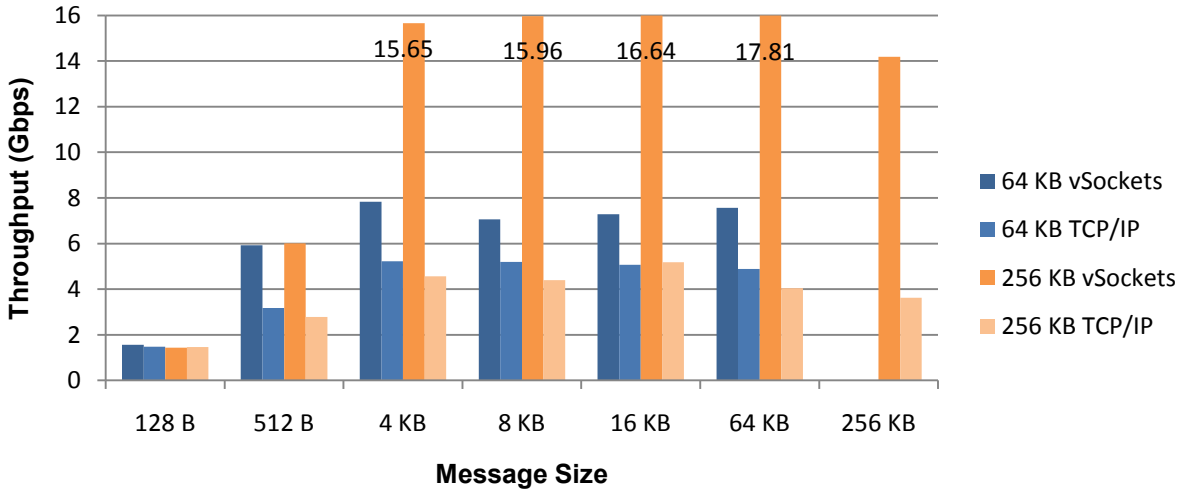
Multi-Session Performance

In this section we discuss multi-session performance to determine how applications which use multiple simultaneous streams would perform using VMCI Sockets. We present results when 5 simultaneous netperf sessions, with the same parameters but different destination ports, were used in the VMs.

Linux VM to Linux VM

The multi-session results for Linux VMs are shown in figure 3.

Figure 3. Linux multi-session performance—256KB vSockets throughput is up to 4 times greater than 256KB TCP/IP throughput



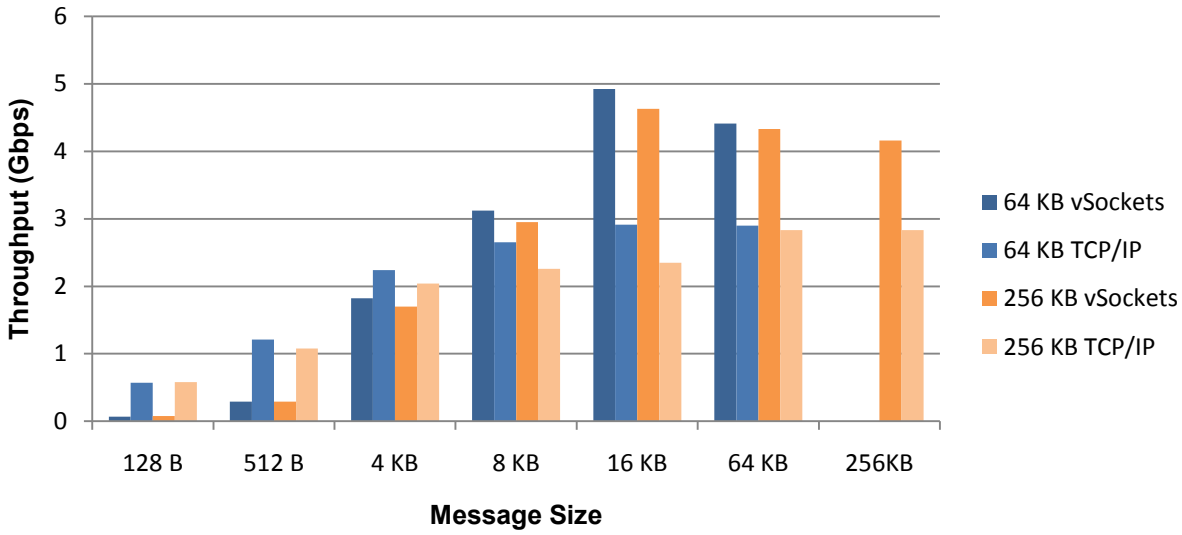
As in the single session case, the performance sweet spot for VMCI Sockets is when the message size is between 4KB and 64KB. VMCI throughput is higher than vmxnet3 throughput in all cases. The difference in performance becomes more prominent as the message size is increased. With 256KB socket buffers (shown in orange), VMCI throughput is up to 4 times more than the vmxnet3 throughput.

The multi-session VMCI throughput is a bit less than single-session throughput. This is due to increased contention among the individual streams.

Windows VM to Windows VM

The multi-session results for Windows are shown in Figure 4.

Figure 4. Windows multi-session performance—vSockets outperforms TCP/IP by up to 100%



For small messages (<8KB), the throughput obtained using TCP/IP sockets is more than the throughput obtained using VMCI Sockets. However, for message sizes of 8KB and greater, VMCI Sockets outperform regular TCP/IP sockets by up to 100%.

Conclusions

In this paper, we presented detailed results of our tests for measuring VMCI Socket performance for inter-VM traffic and compared the results to the performance data for regular sockets. We showed that inter-VM throughput can exceed 29Gbps when VMCI Sockets are used. On Linux VMs, VMCI Sockets outperform regular TCP/IP sockets in nearly all socket size – message size configurations. On Windows VMs, the performance of VMCI Sockets is comparable to vmxnet3 performance with regular sockets being better for small message sizes and VMCI Sockets being superior for large message sizes.

The performance numbers show that the cost of inter-VM communication may be drastically reduced when a given application is ported to use VMCI. It is important to note that the absolute throughput numbers mentioned in this paper are specific to the machine architecture (CPU speeds, cores per socket, and cache configuration) on which we ran the tests. The performance trends should be similar on other architectures, while the absolute gains may vary.

VMCI Sockets thus offer a high performance alternative to TCP/IP sockets for applications which involve a lot of VM to VM traffic on the same host. There is a small porting effort involved in porting an existing application from regular sockets to VMCI Sockets. However, the performance gains of using VMCI Sockets far outweigh the downsides of the additional coding effort involved.

Resources

[1] *VMCI Sockets Programming Guide*. VMware technical publication. Aug 20, 2009.
http://www.vmware.com/pdf/ws65_s2_vmci_sockets.pdf

[2] *Netperf Manual*. Rick Jones, Hewlett-Packard. Accessed Oct 13, 2009.
http://www.netperf.org/svn/netperf2/tags/netperf-2.4.5/doc/netperf.html

About the Author

As a performance engineer at VMware, Bhavjit Walha focuses on ESX networking performance and has written previous papers on this topic. He received his master's degree in computer science from the University of California, San Diego.