# VMware vSphere 4 Performance with Extreme I/O Workloads

VMware vSphere 4.0

VMware vSphere™ includes a number of enhancements that enable it to deliver very high I/O performance. Tests discussed in this paper demonstrate that vSphere can easily support even an extreme demand for I/O throughput made possible by new products like the enterprise flash drives (EFDs) offered by EMC. In our tests we were able to achieve

- 350,000 I/O operations per second from a single vSphere host with just three virtual machines running on it
- 120,000 I/O operations per second from a single virtual machine
- I/O latencies just under 2ms
- 12 percent improvement in throughput with PVSCSI at 18 percent less CPU cost compared to the LSI virtual adapter

Results from these tests show that vSphere performance enhancements enable virtualization of even the most I/O-intensive applications.

VMware vSphere is the industry's first cloud operating system, transforming IT infrastructure into a private cloud. It has enterprise-class storage architecture for virtualization and incorporates a number of new high performance features to help deliver very high I/O throughput. It includes a new paravirtualized storage device called VMware Paravirtualized SCSI (PVSCSI), a new core-offload I/O system to utilize processors on multicore systems, processor scheduler enhancements to improve the efficiency of interrupt delivery and associated processing, and advanced I/O concurrency updates, all of which significantly optimize storage throughput.

The EMC CLARiiON CX4 class of storage systems is the next generation in the CLARiiON series of storage products offered by EMC. CX4 is the first array to support enterprise flash drives in the midrange storage market. The CX 4-960 is a high-end storage array in the CX4 series that can support up to 960 drives (FC, SATA, EFDs, or a combination of these drives).

EFDs can deliver up to 30 times the performance of Fibre Channel hard drives. EFDs are of the same form factor as regular Fibre Channel drives and can be managed readily by existing CLARiiON management tools.

## I/O Workload

We chose an I/O workload that simulated common transaction-oriented applications for our tests. We defined an I/O pattern with an 8KB block size and 100 percent randomness, and with read-write mixes of 90 percent reads and 100 percent reads.

Databases such as Oracle and SQL Server support applications that exhibit this type of I/O behavior, in which read access dominates the I/O load.

## Test Setup

We used the following hardware and software in our tests.

### Hardware and Software Configuration

Server:

- Dell R900
- Quad-socket, six-core, Intel Xeon x7460 at 2.66GHz
- 128GB of physical memory
- Six dual-port QLogic QLE2462 (4GBps) HBAs

Storage:

- Three EMC CX 4-960 arrays [1]
- A total of 30 enterprise flash drives

Virtual infrastructure:

- VMware vSphere 4

Virtual machines:

- Two virtual processors
- 4GB of memory
- Windows Server 2008 EE 64-bit SP1

I/O stress tool:

- Iometer [2]

### Settings

The following table shows the software settings we used during testing. For more information, see "Appendix: Software Settings" on page 7.

**Table 1.** Software settings used for the extreme I/O workload experiments

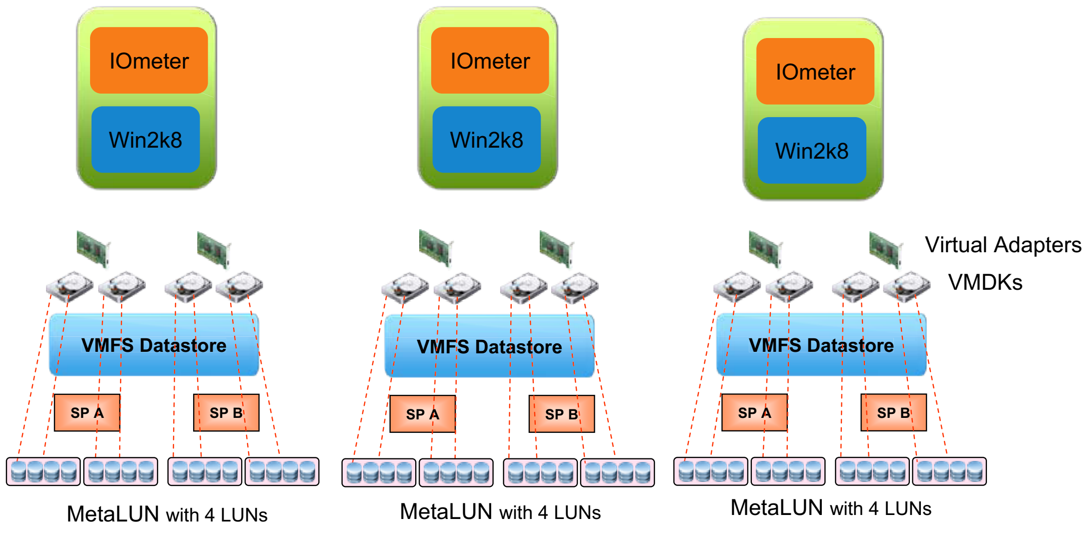
| Configuration Parameter | Settings |
|---|---|
| Multi-pathing policy | Fixed (VMware) |
| Read and write cache on storage arrays | Disabled |
| Device queue depth | 256 |
| Maximum number of outstanding requests per VM | 256 |
| MSI interrupt handling scheme | Enabled |
| Interrupt coalescing | Static; time interval between interrupts: 200 microseconds |
| VM-to-CPU affinity | Enabled |

### Storage Layout

Storage for the tests included three CX4-960 arrays with a total of 30 EFDs. We connected the vSphere host directly to each of these storage arrays via four 4Gbps Fibre Channel links as shown in Figure 1. We grouped the EFDs in each storage array in a RAID 1/0 configuration and created 16 LUNs on the RAID group. We created four metaLUNs on top of the 16 LUNs. Each metaLUN was 50GB in size. We assigned two metaLUNs per storage processor and exposed each metaLUN to the host.

We created a separate VMFS datastore on each storage array by spanning the datastore across all four metaLUNs on that array. In each datastore, we created four virtual disks and assigned them to a single virtual machine using two PVSCSI adapters.

**Figure 1.** Test setup



The storage infrastructure we used (unless otherwise noted) included:

- 30 EFDs spread across three EMC CX 4-960 arrays
- 16 LUNs per array
- Four metaLUNs (two per storage processor) per array
- One VMFS datastore of 200GB per array, spanning all 4 metaLUNs
- Four virtual disks, approximately 50GB in size, per datastore
- Three virtual machines (one per datastore)
- Two PVSCSI virtual adapters per virtual machine
- Four 4Gbps links dedicated to each virtual machine and one array for a total of 12 links between the host and the arrays

We assigned each virtual machine a separate virtual disk where we installed the guest operating system. We created this virtual disk on a separate VMFS datastore, which was created on separate disks.

## Iometer Setup

We used a separate client machine to run Iometer. We installed Dynamo in each virtual machine and configured it to communicate with Iometer in the client machine. This helped us to manage the workload in each virtual machine from a single console. We created two worker threads in each virtual machine. We configured each worker thread to stress two virtual disks. Each worker thread generated 64 outstanding I/Os per disk.

## Tests and Results

We conducted various tests to demonstrate the I/O capabilities of the vSphere storage stack. We discuss the tests and their results in the following sections.

## Performance with a Single Virtual Machine and a Single VMFS

To establish a baseline for the performance observations in subsequent tests, we created four virtual disks in a single VMFS datastore and assigned them to a single virtual machine with two virtual CPUs and 4GB of memory. We used these virtual disks as test disks for Iometer worker threads. We configured Iometer to issue 8K blocks that were 100 percent random and 100 percent read. Table 2 shows the performance we observed from this virtual machine.

**Table 2.** Performance of a single virtual machine

| | IOPS | Latency (ms) |
|---|---|---|
| 1 virtual machine | 119094 | 1.9 |

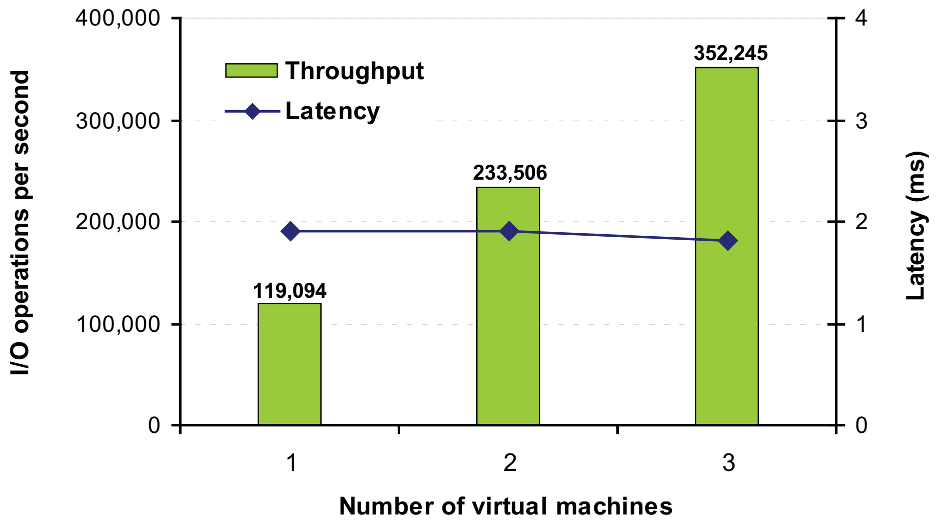
A single virtual machine supported just under 120,000 I/O operations per second (IOPS). Latency of each I/O was just under 2 milliseconds. This virtual machine's performance formed the baseline for remaining tests.

## Scaling I/O Performance on vSphere

To explore the scalability of the vSphere storage stack, we used a building block approach. A single virtual machine with four test virtual disks (explained in "Performance with a Single Virtual Machine and a Single VMFS" on page 4) formed the basic building block. We created these virtual disks in a single VMFS datastore, which was in turn created on a CX4-960 array. We connected the storage array to the vSphere host through four 4Gbps Fibre Channel links. See "Test Setup" on page 2 for more information.

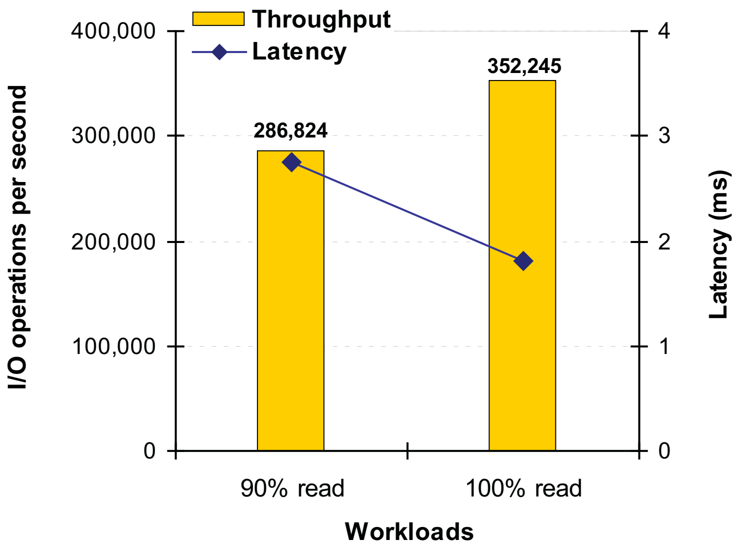
Figure 2 shows the aggregate I/O performance from three virtual machines. The number of I/O operations per second scaled linearly while I/O latency remained constant at less than 2ms. The three virtual machines delivered slightly more than 350,000 I/O operations per second. Due to time constraints, we did not undertake additional testing that could have leveraged the available storage bandwidth to achieve much higher throughput results.

**Figure 2.** Scaling I/O performance using vSphere



In the second part of this test, we changed the Iometer workload profile to introduce some write requests. The I/O profile was 100 percent random, 90 percent read with an 8K block size. Figure 3 compares the performances of 90 percent read and 100 percent read workloads.

**Figure 3.** Comparison of performance with 100 percent read and 90 percent read

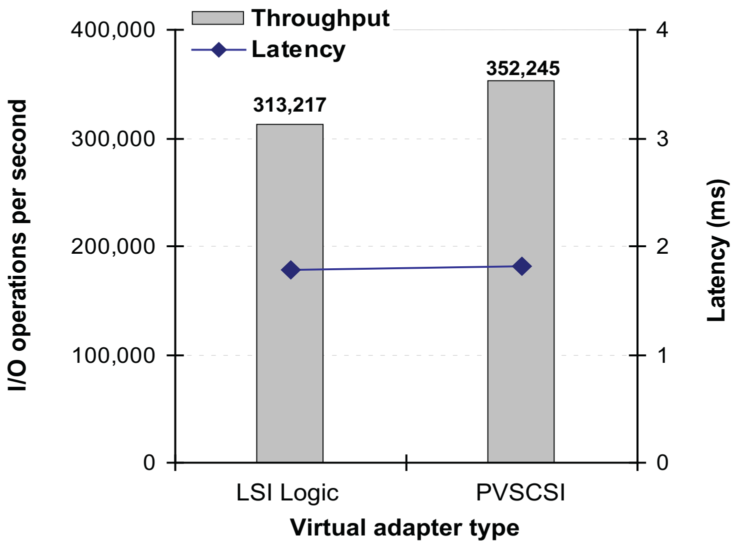


With 10 percent writes in the I/O mix, the aggregate I/O performance dropped to 286,000 I/O operations per second with I/O latencies of approximately 2.75ms. This variance in performance is due to the fact that EFDs provide higher read performance compared to write performance. [3]

## Performance Comparison of Virtual SCSI Adapters in vSphere

Figure 4 shows the performance of LSI and PVSCSI adapters. We used three virtual machines, each with four virtual disks, in the tests to compare the performance of the two adapters.

**Figure 4.** Performance comparison of PVSCSI and LSI virtual adapters



PVSCSI adapters gave 12 percent better absolute performance while the CPU cost of an I/O was 18 percent less compared to LSI.

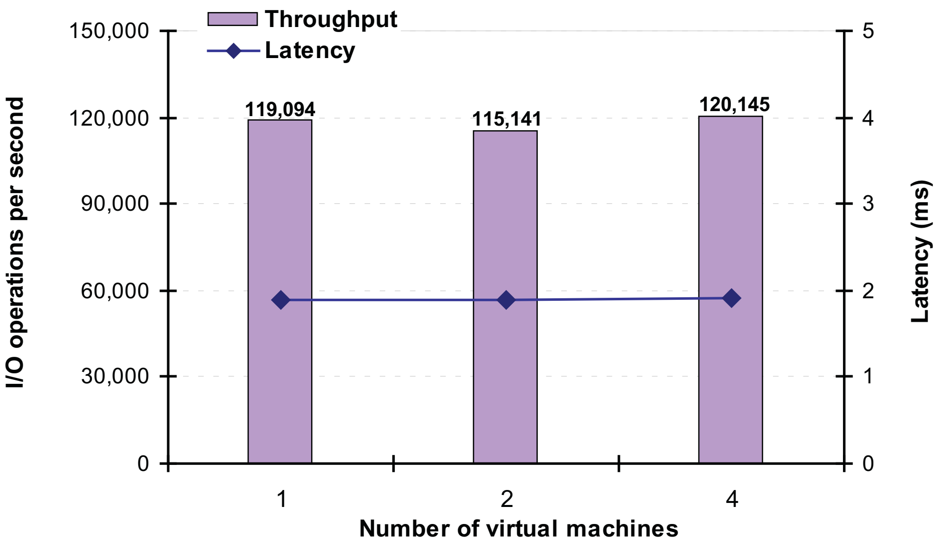
## Comparison of Dedicated and Shared VMFS

VMFS allows virtual machines to share storage resources by abstracting the underlying storage architecture. A VMFS datastore can be dedicated or shared across multiple virtual machines. There is no performance impact of sharing a VMFS datastore across multiple I/O intensive virtual machines.

To test the performance on dedicated and shared storage, we ran a test using four virtual disks on a single datastore. In each iteration, we distributed the virtual disks among a different number of virtual machines and stressed the storage using Iometer running in those virtual machines. All the other aspects of the configurations were identical. The iterations included one, two, and four virtual machines, each stressing four, two, and one virtual disks, respectively. In the case of the single virtual machine, we dedicated the datastore to that one virtual machine. We used the I/O performance of the case using a single virtual machine as the baseline.

Figure 5 compares the I/O performance in the three test cases. The results indicate that there is no difference in I/O performance whether a VMFS datastore is dedicated to a single virtual machine or shared among multiple virtual machines.

**Figure 5.** Performance comparison of dedicated and shared VMFS



## Conclusion

VMware vSphere includes a number of enhancements that enable it to deliver very high I/O performance. The tests detailed in this study demonstrate that vSphere can easily support 350,000 I/O operations per second with just three virtual machines running on a single host. A single virtual machine handled 120,000 I/O operations per second. I/O latency was 1.9ms. The new paravirtualized SCSI adapter supported by vSphere offered 12 percent improvement in throughput at 18 percent less CPU cost compared to the LSI virtual adapter. Sharing a VMFS datastore across multiple virtual machines had no performance impact. These results demonstrate that vSphere can easily sustain even an extreme demand for I/O throughput.

## Appendix: Software Settings

This section explains the settings shown in Table 1, "Software settings used for the extreme I/O workload experiments," on page 2. These settings were required as we approached very high I/O access rates. At lower rates, these settings have a much smaller or negligible effect on performance.

### Multi-Pathing Policy

To maintain a constant connection between a vSphere host and its storage, vSphere supports multiple I/O paths between a host and its storage. Multipathing is a technique that allows more than one physical path to be used for data transfer between the host and the external storage device. Multipathing in vSphere is managed by VMware Native MultiPathing Plug-in (NMP). There are two types of NMP subplugins: Storage Array Type Plugins (SATPs), and Path Selection Plugins (PSPs). SATP handles path failover and is storage array specific. PSP handles physical path selection to a storage device.

For the experiments described in this paper, we required a very high I/O bandwidth to sustain an extremely high number of I/O operations per second. We had to use all the physical I/O paths between the vSphere host and the storage arrays to obtain an I/O bandwidth that could support 350,000 I/O operations per second. We manually assigned a separate physical I/O path to each logical device (metaLUN). We used Fixed PSP for selecting a physical path for every logical device.

For more details, see "Understanding MultiPathing and Failover" in the *Fibre Channel SAN Configuration Guide*. [4]

### Disabling Read and Write Cache

EFDs are extremely fast compared to traditional FC or SATA hard drives. For workloads that have random access patterns, when read cache is enabled for the LUNs residing on EFDs, the read cache lookup for each read request adds significant overhead compared to that of traditional drives. It is faster to read the block directly from EFD.

A saturated write cache causes a forced flush situation where the dirty buffers have to be written to the disks. When this happens frequently, I/O requests incur extra latency as they have to wait to obtain a free buffer in the cache. In these situations, it is better to write the block directly to EFDs rather than to write the cache of the storage system.

We turned off both read and write cache in all the storage arrays following EMC best practices for EFDs.

### Device Queue Depth

EFDs support a very high number of I/O requests per second. To queue the extreme number of I/O requests that were generated by the virtual machines, we had to increase the device queue length from the default value of 32 to 256 in vSphere. To change the queue depth for a Qlogic HBA, refer to Chapter 6, "Managing ESX/ESXi Systems That Use SAN Storage" of the *Fibre Channel SAN Configuration Guide*. [5]

### Maximum Number of Outstanding Requests per Virtual Machine

When two or more virtual machines access the same LUN, the default number of outstanding requests each virtual machine can issue to the LUN is limited to 32. We changed the limit to 256. For changing this setting, refer to KB 1268 [6].

### Message Signaled Interrupts

Message Signaled Interrupts (MSI) is an alternate way of generating an interrupt. A device generates an MSI by writing a small amount of data to a special address in memory space. The chipset will deliver the corresponding interrupt to a CPU. Traditionally, a device asserts its interrupt pin for interrupting its host CPU. Pin-based interrupts are often shared among several devices. When an interrupt is generated, the kernel must call each interrupt handler associated with that interrupt leading to reduced performance for the system as a whole. MSIs are never shared, thus offer performance benefits.

In our experimental setup, we had six dual port HBAs in the vSphere host. Some of the HBAs shared the same interrupt vector. In such situations where multiple HBAs drive a very large number of I/O requests per second, interrupt vector sharing might cause performance issues. To avoid performance problems because of sharing, we enabled MSI for Qlogic HBAs.

### To set MSI

1 Verify which QLogic HBA module is currently loaded by entering the following command:

```
vmkload_mod -l |grep qla2xxx
```

2 Run the following commands.

```
vicfg-module -s ql2xenablemsi = 1 qla2300_707
```

The example shows the qla2300_707 module. Use the appropriate module based on the outcome of step 1.

3 Reboot your host.

## Interrupt Coalescing

Interrupt Coalescing (IC) is a feature that allows a device driver to generate a single interrupt for a burst of I/O requests when received in a short period of time, rather than generating interrupts after every I/O request. IC reduces CPU utilization (and CPU cost per I/O) and improves the throughput of I/O intensive workloads. IC can also increase the latency of I/O operations, however, because an interrupt is generated after several I/O completions instead of each I/O completion. Setting this parameter requires a careful study of the latency sensitivity of the application.

In our experiments, we had twelve Fibre Channel ports (6 dual port 4Gbps HBAs) generating very high I/O requests per second. To reduce the CPU overhead of interrupt processing, we set the delay between interrupts to 200 microseconds (the default is 100 microseconds).

### To change the delay between interrupts

1 Verify which QLogic HBA module is currently loaded by entering the following command:

```
vmkload_mod -l |grep qla2xxx
```

2 Run the following commands.

```
vicfg-module -s ql2xintrdelaytimer=2 qla2300_707
```

The example shows the qla2300_707 module. Use the appropriate module based on the outcome of step 1.

3 Reboot your host.

## VM-to-CPU Affinity

The large number of I/O requests generated by the virtual machines caused the I/O throughput to vary during the benchmark run. To minimize the variance in the I/O throughput, we pinned the virtual machines to physical cores.

## References

[1] CLARiiON CX4 Series product page. EMC. http://www.emc.com/products/series/cx4-series.htm

[2] Iometer Web site. http://www.iometer.org

[3] Zeus$^{IOPS}$ Solid State Drive brochure. STEC.
http://www.stec-inc.com/downloads/ssd_brochures/zeusiops_brochure.pdf

[4] [5] *Fibre Channel SAN Configuration Guide* for vSphere 4. VMware.
http://www.vmware.com/pdf/vsphere4/r40/vsp_40_san_cfg.pdf

[6] "Setting the Maximum Outstanding Disk Requests per Virtual Machine," KB 1268. VMware.
http://kb.vmware.com/kb/1268

## About the Author

Chethan Kumar is a senior performance engineer at VMware. In this role, his primary focus is to analyze and help improve the performance of enterprise applications on VMware vSphere. Before VMware, Kumar was a systems engineer at Dell. Kumar received a Master of Science degree in computer science and engineering from the University of Texas at Arlington.

## Acknowledgements

This testing was the result of a joint effort between VMware and EMC. The author would like to thank the midrange partner solutions engineering team at EMC Santa Clara for providing access to the hardware, for the use of their lab, and for their collaboration throughout this project.