WINDOWS SERVER 2012
ECHOSTREAMS FLACHESAN2

Dan Lovinger
Windows File Server Performance
April 2013

1 INTRODUCTION

This article presents a summary of the performance of the Windows Server 2012 File Server's Remote Direct Memory Access capabilities over SMB 3.0, as observed on a production prototype of the EchoStreams FlacheSAN2[1] appliance with the Mellanox FDR 56Gb/s InfiniBand interconnect solution.

Direct Memory Access (DMA) for storage & network devices is a basic feature of modern systems. Regardless of the transports data crosses to and from a target device – PCIe, SAS, SATA – once I/O is issued to hardware, there is no processor intervention until the data movement is complete. What distinguishes networked operations from local storage devices is that in most[2] cases, bulk data moved over networks arrives inside of multiple protocol layers (Ethernet, IP, TCP, HTTP, etc.). Reassembling the contained data normally requires intervention at a cost of CPU processing time and memory copies.

RDMA protocols separate control and data operations. Similar to storage using attach chains combining PCI Express and SATA, SAS, or Fibre Channel, an RDMA protocol's bulk data transfer is fully implemented in hardware.

With RDMA the processor cost of bulk data access on remote file systems has the potential to approach the range of local storage. This provides a novel build option for deploying high speed and highly efficient consolidated storage solutions.

Results include:

- 512KiB I/Os between a single SMB client and a SMB single server
  - Maximum total bandwidth of **16.4 gigabytes per second** (GB/s)
  - At maximum total bandwidth, a per adapter average of 5.5 gigabytes per second (GB/s)
  - 0.31 CPU cycles per byte (c/B)
- 8 KiB I/Os between a single SMB client and a SMB single server
  - **376,000** I/Os per second (IOPs)
  - 6.4 CPU cycles per byte (c/B)
- Consistently low latency across scenarios and workloads

IOPs were limited by server CPU. This is expected to be increased in the production FlacheSAN2 system.

Note: 1 KiB = 1024 bytes, 1 GB = 10^9 bytes. Please see Section 4 (page 4) for details on the conventions used in this article.

[1] Product information: http://www.echostreams.com/flachesan2.html
[2] An example of a protocol that can be fully offloaded is iSCSI. Software implementations are common, however.

2 CONTENTS

1 Introduction 1
3 Background 3
3.1 Windows RDMA 3
3.2 Tested System Configuration 3
4 Methodology 4
4.1 Measuring CPU Efficiency 5
4.2 Measuring Latency 5
4.2.1 Distributions 5
4.2.2 Wire Latency 6
5 Single I/O Latency 7
6 Small I/O Scaling 9
6.1 Scaled Small Reads 9
6.2 Scaled Small Mixed I/O 11
7 Large I/O Scaling 12
8 Conclusions 14
9 Appendix 16
9.1 Configuration 16
9.2 Figures 16

3 BACKGROUND

3.1 WINDOWS RDMA

Three RDMA implementations are supported in Windows Server 2012:

- iWARP
- RoCE
- InfiniBand

iWARP, RDMA over TCP/IP, and RoCE, RDMA over Converged Ethernet, ride on existing Ethernet networks at up to 40Gb. InfiniBand is available at 56Gb (FDR, 54Gb data rate), speed matched to PCIe 3.0, and will be used in the configuration discussed in this paper.

SMB Direct implements an RDMA transport for SMB 3.0 on Windows Server 2012. The underlying RDMA support is exposed by providers implementing to the Network Direct Kernel Programming Interface (NDKPI). An RDMA provider offers the infrastructure for an upper-level transport to perform DMA operations, including:

- management of memory registration
- sending and receiving small messages (including the transport protocol)
- transfer of registered memory

SMB Direct is also a simple protocol (MS-SMBD[3]) carried on the RDMA send/receive path. Extensions to SMB 3.0 transfer registered memory descriptors, which are then given back to the RNIC to perform direct transfers of file data.

3.2 TESTED SYSTEM CONFIGURATION

The systems used for this analysis were provided by EchoStreams and Mellanox. Please see Figure 1, below.

The EchoStreams FlacheSAN2 is an appliance which combines SAS HBAs, enterprise SSDs and high speed networking to build a high performance file server using Windows Server 2012 and Storage Spaces. The build was as follows:

- Networking: three Mellanox ConnectX-3 FDR InfiniBand HCAs
- Storage
    - 5x LSI PCIe Gen 3.0 SAS HBA

[3]Protocol documentation for MS-SMBD can be found at the MSDN Open Specifications site: http://msdn.microsoft.com/en-us/library/hh536346(v=prot.13).aspx

    - 8x Intel 520 SSDs per controller
    - Total: five groups of eight, for 40 total SSDs
- CPU: 2x Intel Xeon E5-2650 (8c16t 2.00Ghz)
- DRAM: 32GB

The FlacheSAN2 has capacity for another HBA and SSD group (48 total SSD), but this sixth HBA was not in the production configuration for this prototype system. As a result only five groups were used.

Each HBA and SSD group was used to build a Storage Space, on each of which a mirrored virtual drive was created (4 column, 2 copy), each of which was exposed as an SMB Share. Five total Storage Spaces, Virtual Drives, and SMB Shares were created.

The client was a white-box system with a matching three Mellanox ConnectX-3 FDR InfiniBand HCAs using 2x Intel Xeon E5-2680 (8c16t 2.7GHz, HT Disabled).

For more details on the FlachSAN2, please reference EchoStream's product page:

http://www.echostreams.com/flachesan2.html

For more details on the system configurations, please see Section 9.1 (page 16).

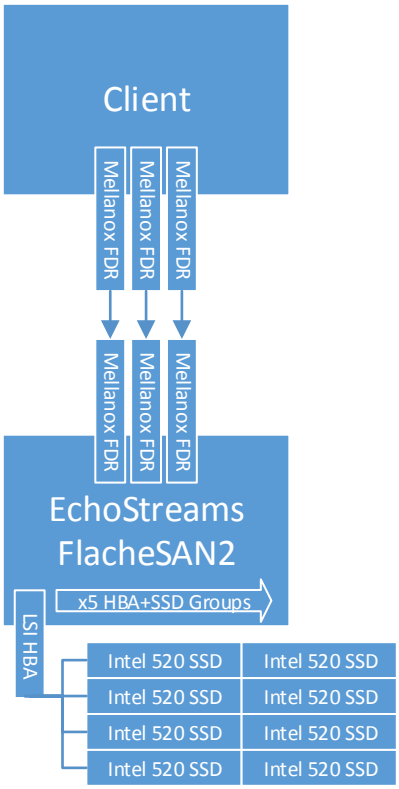

FIGURE 1 – SYSTEMS AND CONNECTIONS

Near-production builds of the FlacheSAN2 do have the sixth SSD group active (for six HBAs and 48 total SSD), and have bumped the CPU frequency to 2.40 GHz. Since the FlacheSAN2 CPU was a limiting factor for small I/Os (see Section 6, page 9), it is expected the final device will show a significant performance increase relative to that seen in this article.

4 METHODOLOGY

All results were captured using the RTM build of Windows Server 2012 (Build 9200) running with out-of-box defaults.

The SQLIO load generator was used to drive load from the client, including an option to affinitize its load threads to a specific processor in the dual processor socketed client. All multi-threaded results discussed were captured by running two instances of SQLIO, each of which ran half of the total threads and assigned its threads to its assigned processor: the first instance ran on processor one, the second on processor two. Each virtual drive exposed two 100GiB files, one of which was used for load driven from processor one and one for load from processor two.

This affinity configuration is a partial emulation of an enterprise application designed with awareness of how to manage the costs of non-uniform memory access[4] (NUMA) costs in multi-socket systems. As an example, a virtual hard disk (VHD) would only be accessed by a specific virtual machine, which in common cases would be assigned to CPU cores in a specific socket.

[4] http://en.wikipedia.org/wiki/Non-Uniform_Memory_Access

Note: The network convention of KB/MB/GB referring to power-of-10 quantities will be used: thousand (KB), million (MB) and billion (GB). Power-of-2 units will use the IEC 60027-2[5] convention of KiB/MiB/GiB where appropriate.

1000 B = 1 KB

1024 B = 1 KiB

4.1 Measuring CPU Efficiency

To measure CPU efficiency of storage, a conventional metric is CPU cycles per byte of user data transferred. Cycles per byte in this report is calculated using the following formula.

EQUATION 1 – CYCLES PER BYTE

$$c/B = \frac{\%Priviliged\ CPU\ Utilization \times Core\ Clock\ Frequency \times \#Cores}{Bandwidth\ in\ Bytes}$$

Using privileged CPU utilization focuses on the system kernel costs of operations, excluding time used by the load generator itself and incidental system service activity. This is available from the Windows ***Processor / % Privileged Time*** performance counter. For consistency, all CPU utilization in this article will refer to % Privileged Time.

In order to measure cycles per byte, it is necessary to eliminate (to the degree possible) variation in processor frequency so that it can be treated as a fixed value in the calculation. This requires some changes to common processor and performance features:

- disabling HyperThreading
- disabling Turbo Boost & SpeedStep
- disabling Virtualization
- disabling deep C-States in BIOS
- enabling the Windows High Performance power plan

These changes in configuration can affect the absolute performance results, for instance by disabling Turbo Boost, and can also increase power utilization by disabling the ability to down-clock during idle periods. All results discussed here were captured with the client system configured to measure cycles per byte which, along with the small overhead of tools used to measure system performance, makes the results in this article conservative estimates of absolute performance.

4.2 Measuring Latency

4.2.1 Distributions

The analysis of latency will use the distribution (histogram) of I/O latencies as opposed to average I/O latency, which might mask the variability of a poorly performing configuration. Distributions allow us to see the total response behavior of the system and validate that the observed performance is consistent and sustained.

A cumulative distribution is a further way of visualizing the distribution. Where the histogram shows how many samples occur in each interval, the cumulative distribution shows the percentile total up to that point. The 50th

[5] Prefixes for binary multiples: http://physics.nist.gov/cuu/Units/binary.html

percentile is the median – 50 percent of the samples are to either side. It is normal to use the 90th percentile for latency as a measure of common behavior.

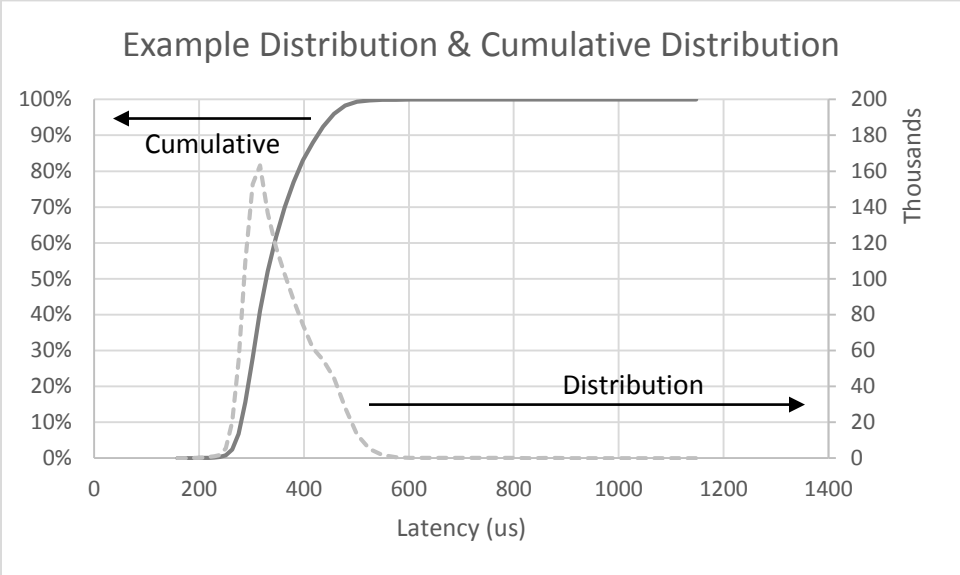

FIGURE 2 – AN EXAMPLE DISTRIBUTION

Figure 2, above, shows a latency distribution which rises quickly at 300 microseconds and then smoothly tails off to 600 with a final few values going up to around 1150us. The 50% mark on the darker, cumulative distribution, is around 350 microseconds. In more complex and potentially less desirable cases, there could be multiple peaks.

All latency data was captured using the Windows Performance Analyzer[6] with file I/O tracing, on both the client and server. This provided individual I/O timing. Each distribution in this paper sampled 1-1.5 million I/Os or, in lower IOPs cases, one minute's worth of I/O. Three minutes of warm up load was applied to the storage prior to taking latency measurements.

4.2.2 Wire Latency

There are two main components of the total latency seen at the client, as diagrammed in Figure 3, below. The Windows Performance Analyzer samples key metrics immediately above the NTFS file system and the SMB Client. On the server side this measures the response time of the file system and storage, and on the client this measures the complete end-to-end response time.

The difference between the two is the time taken in the SMB 3.0 Server, network, and SMB 3.0 Client – the latency of using remote access to the storage, the wire latency.

[6] WPA: http://go.microsoft.com/fwlink/?LinkId=214551

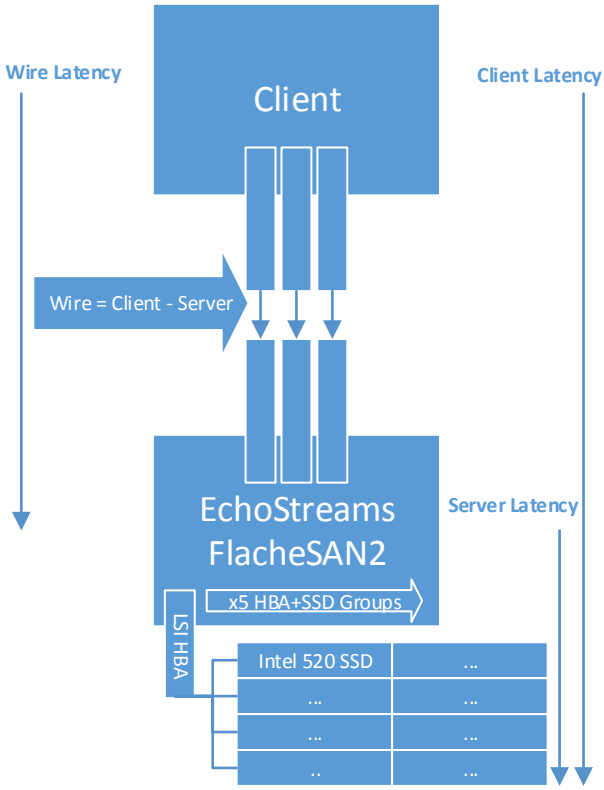

FIGURE 3 – COMPARING CLIENT, WIRE AND SERVER LATENCY

To find wire latency, the distributions of client and server latency will be compared. For instance, if the 90th percentile latency for the client is at 200us and for the server is at 150us, the 90th percentile latency of the remote access (the wire) is 50us.

5 SINGLE I/O LATENCY

Starting with the latency of single I/O operations, the load generator was configured to issue a single random I/O at a time to one share on the FlachCache2.

This section will introduce a way to view latency that will be repeated through the remainder of the article, using the cumulative distributions discussed earlier. The goal will be to show that the system is both performing well *and* performing consistently.

TABLE 1 – SINGLE RANDOM I/O LATENCIES AT CLIENT AND SERVER

Single I/O Latency (us)	90th Percentile Read			90th Percentile Write		
Size (KiB)	Client	Server	Wire	Client	Server	Wire
1	204	176	**29**	153	119	**34**
8	197	159	**38**	113	65	**49**
64	419	366	**52**	366	303	**63**
512	1297	1112	**185**	1355	1143	**212**

Table 1, above, shows the 90th percentile latencies measured at the client and server, and the resulting wire latency estimate. The storage itself dominates, with the notable case of 8KiB writes where the SSD physical characteristics line up very efficiently with the I/O.

To put the 90th percentile in context, Figure 4 (below) shows the 8KiB read latency distribution measured at the server. This shows a total of ~330,000 I/Os clustering in the range of 120-160us with a small tail out to 200us. What we see here is the behavior of the local SSDs.

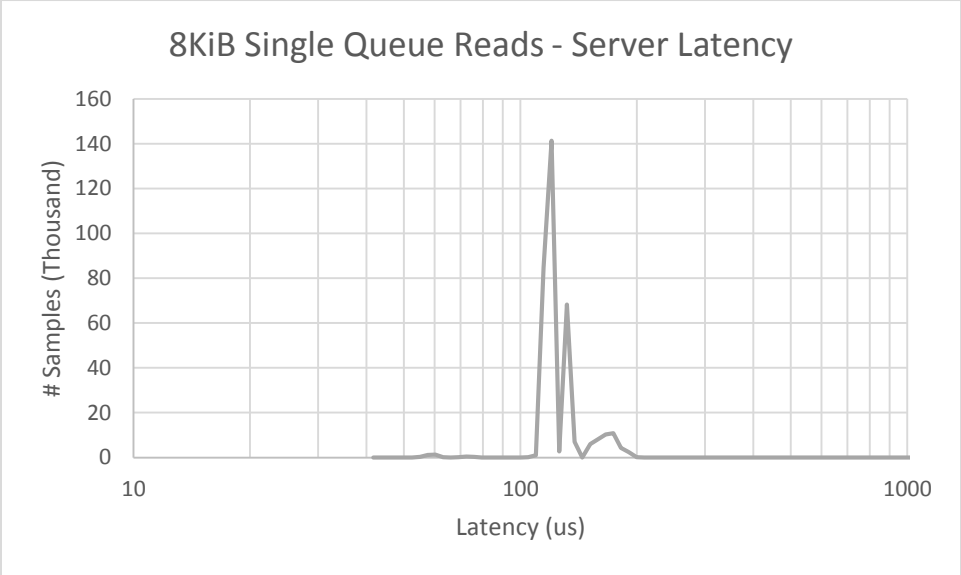

FIGURE 4 – 8KIB I/O CLUSTERED IN THE LOWER-MID 100US RANGE

Moving to Figure 5 (below), the server latency is shown alongside the measurements at the client, using cumulative distributions. Here the 90th percentile latency from Table 1 (above) can be seen in context, in the upper part of the curves.

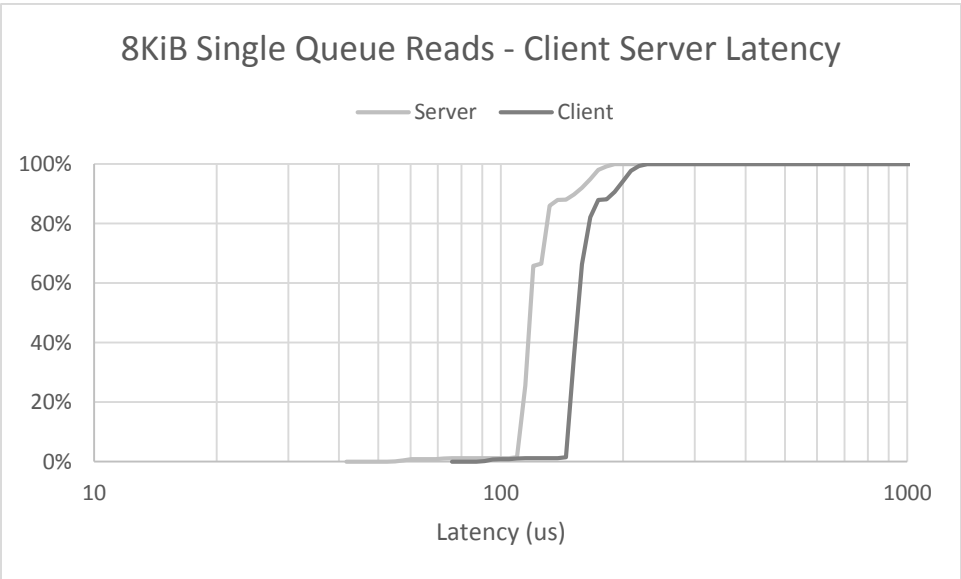

FIGURE 5 – THE SHIFT IN LATENCY FROM CLIENT TO SERVER APPEARS CONSTANT

Seeing the two side by side, there is a strong suggestion of a nearly uniform separation, which is the wire latency discussed earlier.

Finally, Figure 6 (below) shows the wire latency from the 10th to 95th percentiles for all of the I/O sizes measured.

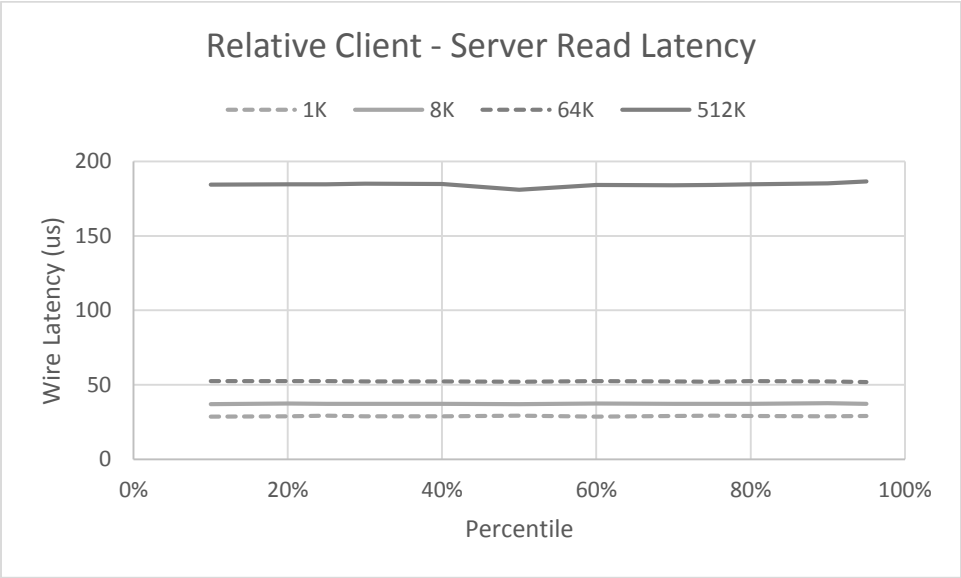

FIGURE 6 – CONSTANT WIRE LATENCY

As is apparent, the wire latency is essentially constant across all I/O sizes measured. This tight range of performance is an important starting point to build the value of the file server solution, showing that at initial loads the latency contribution of the network transfer is both modest and constant.

6 Small I/O Scaling

This section will continue the latency analysis to show that the consistency seen for single I/Os continues to high loads, first for small I/O operations. A matrix of loads were run to identify specific ranges where scaling a single load parameter took the system just to the point of saturated performance.

Along with simple scaling, this section also aims to demonstrate efficiency in the range of local storage access. This is the underlying motivation for RDMA, and essential for making the case for remote file system storage.

6.1 Scaled Small Reads

Looking first at reads, the load generator was configured to issue four I/Os per thread per share. Across the five shares, each thread issued 20 simultaneous I/Os. The threads were then scaled up to the number of available processing cores on the client, 16. At the upper limit the load generator was driving 320 I/Os simultaneously.

TABLE 2 – SCALING OF READ IOPS IS CONSTRAINED BY SERVER CPU

Read Scale	1KiB					8KiB				
Threads	**IOPs**	**90th (us)**	**c/B**	**%CPU**	**%CPU Srv**	**IOPs**	**90th (us)**	**c/B**	**%CPU**	**%CPU Srv**
1 (20 I/O)	76650	265	43.3	7.9	10.8	64500	310	7.9	9.7	9.8
2 (40 I/O)	144050	320	43.3	14.8	21.7	123600	365	7.0	16.4	20.3
4 (80 I/O)	244250	390	41.4	24.0	48.4	211500	445	6.5	26.3	46.6
8 (160 IO)	360950	560	41.7	35.7	84.5	327050	530	6.4	40.0	82.5
16 (320 IO)	438400	1040	44.9	46.6	99.9	425900	955	7.2	58.2	100.0

Table 2, above, shows the scaling of IOPs, read latency, cycles per byte (CPU efficiency), and the total client and server CPU utilization. The system reaches into mid-400K IOPs before running into limits, discussed below. Beyond that there are several interesting effects.

The first is CPU efficiency. The client gains modest c/B improvements as I/O load increases, due increased batching of work, up to the point where the end-to-end system is at saturation. The 6.4 client c/B of 8KiB reads at high rates immediately before saturation, at 8 threads, is notably close to local storage efficiency. In similar configurations and thread/depth loads, Microsoft has measured c/B at 6.0-6.5 c/B for locally attached Fibre Channel and SAS solutions.

Very small 1KiB I/Os, generally not seen in production workloads, are significantly less efficient per byte. In terms of total CPU utilization, though, they are somewhat more efficient per I/O. This is due to the small I/Os being able to embed the transferred data directly in the SMB protocol, and skip the memory registration used to perform bulk data movement. The efficiency tradeoff of the added processing for bulk data movement begins to switch over by 8KiB.

At saturation, the limit is the server CPU, which was completely utilized. The effect can be seen strongly in how read latency spikes in the last step to 16 threads. This is normal, showing the effect of I/O queuing behind the server CPU and common when reaching for maximum operation rates with simple load generators such as IOMETER or SQLIO. These synthetic generators scale up by queuing constant numbers of I/Os per thread, and tuning them to the point of balanced performance can be an art very specific to their design – as opposed to the considerations more appropriate to end-to-end production workloads. We can estimate, though, that at some point closer to the 8 thread load – 9, 10 threads or a slightly rebalanced I/O depth – these systems would reach that point close to the mid-500us latency.

Figure 7, below, shows the scaling for the 8KiB reads in chart form.

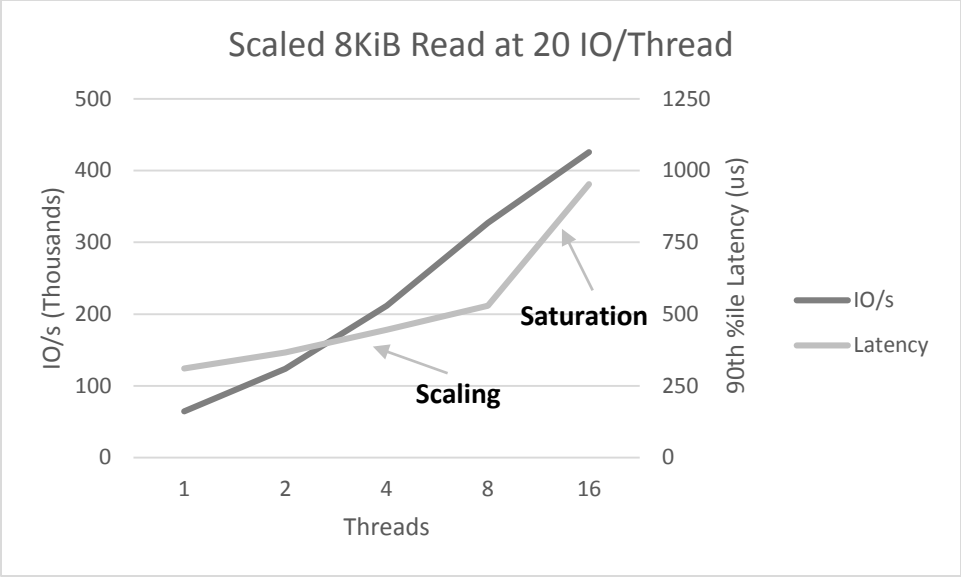

FIGURE 7 – READS SCALE SMOOTHLY TO THE SATURATION POINT

Similar to that seen in Figure 6 (page 9) for single I/Os, Figure 8, below, shows that the relative wire latency is in a narrow band through the range where I/O is scaling up under load. This shows the consistency in the end-to-end performance of the Windows 2012 Server SMB and networking layers.

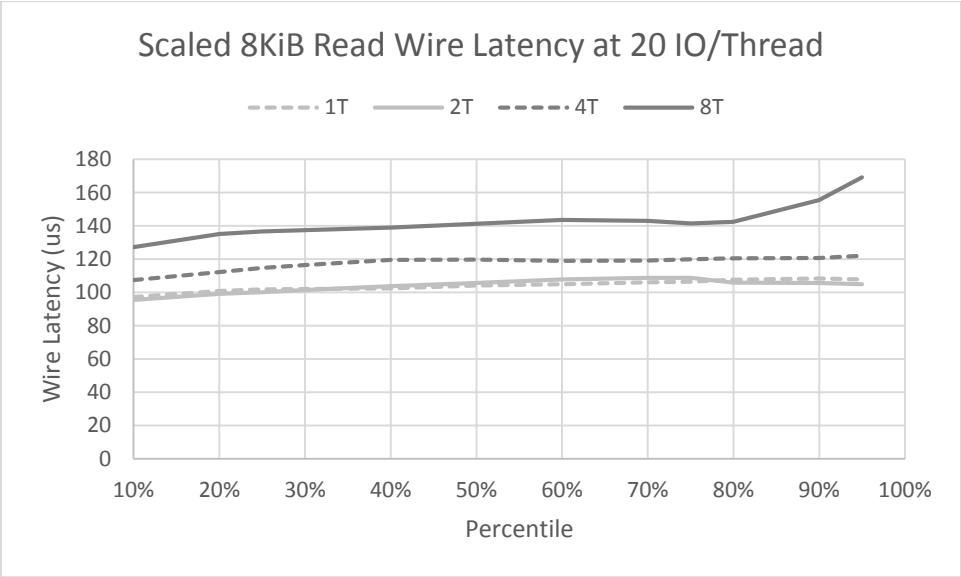

FIGURE 8 – WIRE LATENCIES ARE CONSISTENT AS LOAD SCALES

6.2 SCALED SMALL MIXED I/O

Moving from a read-only workload to a mixed read-write load, the load generator was configured to a 60:40 ratio of random reads and writes. This is a common model for Online Transaction Processing (OLTP) loads for database operations, which balance reads from incoming queries and writes of dirty data back to database files.

TABLE 3 – SCALING OF MIXED I/O'S IS SIMILAR TO READ-ONLY

60R/40W Scale	1KiB					8KiB				
Threads	**IOPs**	**90th R(us)**	**90th W(us)**	**c/B**	**%CPU**	**IOPs**	**90th R(us)**	**90th W(us)**	**c/B**	**%CPU**
1 (20 I/O)	69800	300	350	44.3	7.3	70700	310	270	7.1	9.5
2 (40 I/O)	125900	355	410	45.3	13.5	124950	370	340	7.0	16.6
4 (80 I/O)	206450	435	495	43.4	21.3	210850	450	410	6.9	27.4
8 (160 I/O)	319150	545	635	39.6	30.0	328150	575	510	6.8	42.5
16 (320 I/O)	424850	960	1140	47.1	47.5	375900	1235	1330	7.4	52.7

Similar to the results in the previous section, the server CPU (not shown) saturated in the final step for both I/O sizes. CPU efficiency is also similar through the range where the systems were scaling, and saturation of the systems is clearly indicated by the jump in latency.

Very importantly for performance, though, the systems were also similar at the network layer when performing bidirectional transfers at high rates. Figure 9, below, shows the behavior at 8 threads (and 1.3M mixed IOPs):

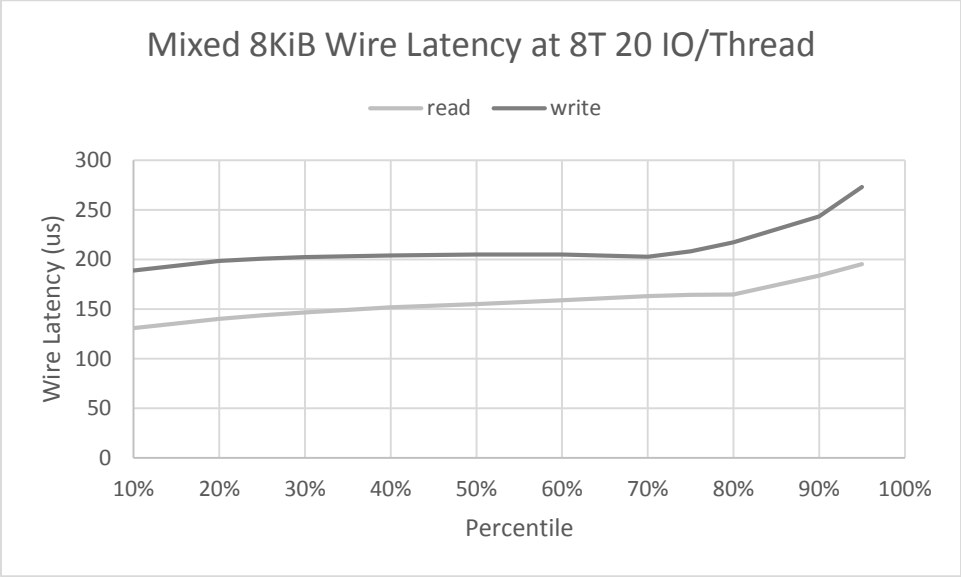

FIGURE 9 – WIRE LATENCIES ARE CONSISTENT FOR MIXED I/O'S AS WELL

7 LARGE I/O SCALING

Moving to large I/Os of 64KiB and 512KiB, the load generator was configured to 100% read and writes loads, scaling across the same thread and depths used for small I/O. In the case of reads, this is a match to classic data mining workloads, where a database or processing application aggregates across very large datasets with complex or unindexable queries. The bandwidth and capacity of the storage subsystem is very important in these scenarios.

Random write workloads encounter the limits of the underlying SSDs, which reach maximums of 5.8GB/s at 512KiB. To avoid repetition, we'll focus on a deeper latency analysis of the much faster read load, just prior to complete saturation.

Table 4 (below) shows the scaling of large reads.

TABLE 4 – LARGE I/O SCALES TO 14.4-16.4GB/S AS SATURATION OCCURS

Large Read Scale	64KiB				512KiB			
Threads	**GBytes/s**	**90th (us)**	**c/B**	**%CPU**	**GBytes/s**	**90th (us)**	**c/B**	**%CPU**
1 (20 I/O)	2.45	550	1.22	6.9	6.64	1630	0.31	4.7
2 (40 I/O)	4.95	630	1.06	12.2	11.34	2570	0.29	7.6
4 (80 I/O)	8.58	780	1.05	20.8	14.41	4970	0.29	9.8
8 (160 I/O)	11.84	1300	1.06	29.0	15.68	9930	0.30	10.9
16 (320 I/O)	13.99	2520	1.09	35.3	16.40	19900	0.31	11.6

The combined effect is to scale to **14.5-16.5GB/s** CPU efficiency is also notable – similar saturation of classic 10GB Ethernet with TCP transports would be significantly above 1 c/B, while local storage is in the range of 0.20 c/B.

Latency increases significantly at the 90th percentile as well, though, which we will look at next. Figure 10 (below) shows the contribution of the network (wire) communication to the total latency for the 8 thread / 160 I/O case, highlighted in Table 4.

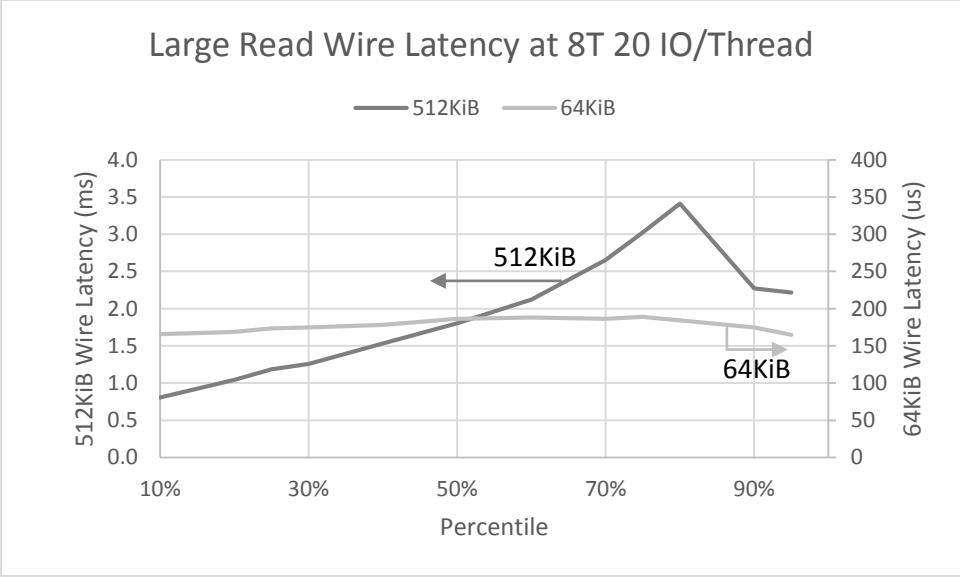

FIGURE 10 – BANDWIDTH LIMITS CREATE A SPREAD OF LATENCIES FOR LARGE I/O'S, MORE PRONOUNCED FOR 512KIB

Note: The relative scales are different – microseconds for 64KiB (right) and milliseconds for 512KiB (left).

In both cases, at 64KiB and 512KiB, a fixed amount of work is being fed into the system. As the size of the response grows, the system begins to run into the bandwidth limits of the network transport. The nodes here have three independent FDR InfiniBand adapters, and Windows Server 2012 schedules the I/O in a round robin fashion among them using SMB Multichannel. For the 512KiB I/Os, the sizes, speeds, and maximum latencies work out as follows:

- 160 I/Os * 512KiB = 80MiB of outstanding I/O

- 3 * 54Gb/s network = 20.25GB/s theoretical maximum bandwidth
- 80MiB / 20.25GB/s = 4.1ms transmit time

The result, in Figure 10 (above), shows the chance of an I/O encountering smaller vs. larger queuing to transmit back to the client. The proportionally smaller I/O at 64KiB can still be seen to experience a similar, though much reduced, effect. This is generally expected for a synthetic workload.

Compared to Figure 8 (page 6), the difference between reaching a multi-adapter bandwidth limit and CPU limit can be seen. Running out of CPU first has a relatively even impact on all I/Os, whereas running into bandwidth limits on very large I/Os can see a broader range of effects.

Figure 11 and Figure 12, below, offer another pivot on the system scaling, showing the rise in total (network + storage) latency as the system runs toward bandwidth limits.

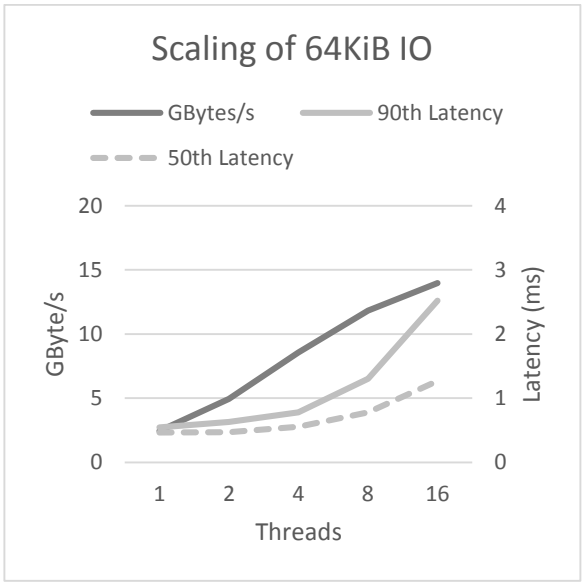

FIGURE 11 – 64K QUEUE BLOCKING

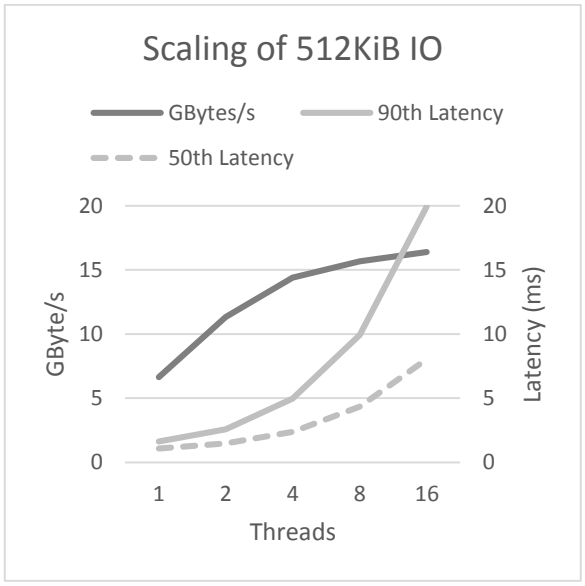

FIGURE 12 – 512K SEES MORE QUEUE LIMITS DUE TO SIZE

In the broader view, it is remarkable to be discussing bandwidth limits in the range of 16GB/s for file-based network communication between two systems. This shows how far Windows Server 2012 has come, and the capabilities of the platform provided by the EchoStreams FlacheSAN2.

8 CONCLUSIONS

This article has demonstrated the performance potential of Windows Server 2012 on a new class of hardware represented by the EchoStreams FlacheSAN2 containing Mellanox FDR InfiniBand connectivity.

- Maximum bandwidth of **16.4GB/s** (~5.5GB/s/adapter) at 0.31 c/B, with 512KiB I/Os (Section 7, page 12)
- IOPs of **376,000** at 6.4 c/B, with 8KiB I/Os (Section 6.1, page 9)
- A near-constant latency profile introduced at the network layer for all cases

In the course of this analysis, we have also introduced some reference points for latency scaling. In particular, queuing theory predicts the effects seen in Section 7 for very large I/Os. Synthetic workloads used in bench analysis are limited in this respect; production systems may see different results due to the specifics of their workloads.

The combination of RDMA and the implementation of SMB 3.0 on Windows Server 2012 result in very good CPU efficiency, close to that seen for locally attached storage. Together with the performance demonstrated, it is reasonable to consider remotely deployed storage on Windows Server 2012 for highly scaled computing environments.

The EchoStreams FlachSAN2 is a notable example of a new class of storage device. As discussed earlier, these results were produced on a pre-production prototype of the device. At production, with increased CPU and an additional production-ready group of 8 SSDs (48 total) (see Section 3.2, page 3), we have good reason to look forward to increased performance for the small I/Os critical to most virtualized environments.

9 APPENDIX

9.1 CONFIGURATION

TABLE 5 – ECHOSTREAMS FLACHESAN2/SERVER

CPU	2x Intel E5-2650 2.0Ghz, 8c16t Total of 16c32t
RAM	32GB, 16GB/socket in 1x4x4GB DDR3-1333
Network	3x Mellanox FDR InfiniBand, PCIe 3.0 x8
Storage	5x LSI 9207-8e, PCIe 3.0 x8 8x Intel 520 SSD per HBA
Storage Configuration	Microsoft Storage Spaces 1x Space per HBA composed of all 8 SSDs 1x Mirrored Virtual Drive (4 Column 2 Copy) per Space 5 total Virtual Drives on 40 SSD

TABLE 6 – CLIENT

CPU	2x Intel E5-2680 2.7Ghz 8c16t HT, Turbo, SpeedStep, deep-C disabled Total of 16c16t
RAM	128GB, 64GB/socket in 2x4x8GB DDR3-1600
Network	3x Mellanox FDR InfiniBand, PCIe 3.0 x8

9.2 FIGURES

Figure 1 – Systems and Connections 4
Figure 2 – An Example Distribution 6
Figure 3 – Comparing Client, Wire and Server Latency 7
Figure 4 – 8KiB I/O clustered in the lower-Mid 100us range....8
Figure 5 – The Shift in Latency from Client to Server Appears Constant....8
Figure 6 – Constant Wire Latency....9
Figure 7 – Reads Scale Smoothly to the Saturation Point 11
Figure 8 – Wire Latencies are Consistent as Load Scales....11
Figure 9 – Wire Latencies are Consistent For Mixed I/O's as Well 12
Figure 10 – Bandwidth Limits Create A Spread of Latencies for Large I/O's, More Pronounced For 512KiB 13
Figure 11 – 64K Queue Blocking 14
Figure 12 – 512K Sees More Queue Limits Due To Size....14