Using CIFS as a Global File System for HPC Clusters

Microsoft Corporation

Published: October 2007

Updated:

Executive Summary

The Common Internet File System (CIFS) can provide a low-cost, high-performance global file system for small and mid-sized high-performance computing (HPC) clusters. This white paper presents data gathered at the University of Toronto demonstrating that a Windows® operating system–based storage solution can provide sustained data transfer rates in excess of 100 MB/sec per server.

Table of Contents

Introduction

This white paper presents data that shows that the Common Internet File System (CIFS), a network protocol that allows file sharing across a local area network (LAN), can provide a low-cost, high-performance global file system for small and mid-sized high-performance computing (HPC) clusters.

The data was gathered at the University of Toronto on a 61-node HPC cluster running Windows® Compute Cluster Server (WCCS) 2003. The goal of the study was to determine if a Windows–based storage solution for an HPC cluster could handle data transfer rates in excess of 600 megabytes per second (MB/sec), performance that is adequate for many applications. The University of Toronto cluster uses multiple CIFS servers as its global file system, with multiple shares and multiple Gigabit Ethernet (GigE) network interface cards (NICs) on each storage server.

For the study, engineers ran a simple Message Passing Interface (MPI) application that handled I/O. The data layout was constrained so that the MPI program placed the data on the separate storage servers and on separate shares on each of those servers. The original design goal used six storage servers, but only four were available at the time of the study.

Engineers first measured the local I/O performance on each of the four available servers. They then measured the aggregate remote I/O throughput for multiple I/O streams from the MPI program to the storage servers.

Remote I/O aggregate bandwidths in excess of 100 MB/sec per server were measured for each of the four storage servers. The aggregate performance appeared to scale linearly, therefore the design goal of an aggregate data transfer rate of 600 MB/sec seems achievable with six storage servers.

The study demonstrates that if data is properly spread across multiple servers and shares, CIFS can yield extremely high data transfer rates at a fraction of the cost of high-end file system solutions. This solution is appropriate for many small and mid-sized clusters.

CIFS Overview

To determine the most appropriate network file system for a cluster, you should carefully consider the types of traffic on the network and the level of performance you need. Ideally, you should use a native file transfer protocol that is integrated into the system; this facilitates data transfer across the network. It is often unnecessary to invest in a high-end, expensive solution. An option for a small to mid-sized cluster is one or more Windows servers and CIFS. Using CIFS, rates greater than 100 MB/sec per storage server have been demonstrated on servers with internal redundant array of inexpensive disks (RAID). More robust server configurations have demonstrated rates of 500 MB/sec per server. (See the Microsoft Windows Storage Server 2003 R2 TDM/BDM White Paper at: www.microsoft.com/windowsserversystem/wss2003/productinformation/wss2003r2-tdmbdm-wp.mspx.)

What Is CIFS?

CIFS is a specification of the Microsoft cross-platform Server Message Block (SMB) protocol, the native file-sharing protocol in the Microsoft® Windows® 95, Windows NT®, and OS/2 operating systems. Millions of computer users utilize CIFS to share files across corporate intranets. CIFS is also widely available on UNIX and other platforms.

CIFS Features

Most traffic travelling across LANs occurs between Windows clients and servers, and most traffic in Windows-based networking environments is CIFS. Windows runs CIFS natively; there are significant performance benefits because CIFS is an integral part of a Windows environment. CIFS is well suited for Windows HPC clusters.

CIFS has the following advantages:

- *File access* – CIFS allows multiple clients to access and update the same file, but also prevents conflicts by using advanced file-sharing and locking mechanisms. These mechanisms also permit aggressive caching, in addition to read-ahead / write-behind methods, without losing cache coherency.
- *Directory paths* – When CIFS is used, there is an efficient cross-platform method of identifying SMB shared resources, both internally on a LAN and on a wide area network (WAN). Windows clients are fine-tuned to yield the best possible performance, throughput, and file transfer capabilities when establishing native CIFS connections between client and server.
- *Network performance* – CIFS increases performance if a Windows client can buffer file data locally. The client does not have to write information into a file server if the client knows that no other processes are accessing the data. The client can buffer read-ahead data from the file if no other processes are writing the data.
- *Natively supported performance advantages* – With CIFS, users can share files without having to install additional software to facilitate data transfers. Internally, CIFS runs over TCP/IP, but it uses the SMB protocol found natively in Windows for both file and print access. Efficiency is maintained; when CIFS is used to make changes in any given file, the changes are saved on both the client and the server.

- *File security* – CIFS servers support both anonymous transfers and secure, authenticated access to named files. File and directory security policies are easy to administer. With CIFS, operating within Active Directory® and with NTFS (Windows NT file system) enables file security based on user credentials and global policies.

University of Toronto Study Results

The University of Toronto study was conducted in July of 2007. The goal of the study was to determine if CIFS could provide an adequate global file system for the university's HPC cluster.

Hardware Used

The Windows-based HPC cluster at the University of Toronto is used primarily for geophysical research on rock dynamics. The work requires a throughput of about 600 MB/sec, or 100 MB/sec for each of six storage servers.

In this study, four of the storage servers and 61 compute nodes were used with an Ethernet network dedicated to I/O traffic. The hardware used is described in Table 1. It is important to note that the four storage servers in the test setup were not identically configured; there were differences in the RAID settings and in the spindle counts.

Table 1 Hardware Used in University of Toronto Study

Storage servers	Four Dell 2950 servers, each with: Dual socket, dual core Six SATA disks in a RAID-5 configuration Two GigE connections to the Nortel switch
Compute nodes	61 Dell 1950 servers, each with: Dual socket, dual core Two GigE connections to the Nortel switch 15 nodes with 8 GB of memory 46 nodes with 4 GB of memory
Head node	One Dell 2950 server with: Dual socket, dual core Two GigE switches, five Nortel 5510 switches running multiple virtual local area networks (VLANs)

Data Transfer Rate Tests and Results

Only four of the six storage servers were available for gathering data during the study. Each of the four storage servers was configured to have four shares. To measure the throughput, an MPI program that handled simple C-based I/O was used. Three basic tests were run: local I/O, remote I/O with four processes per node, and remote I/O with one process per node. For the local I/O test, a serial version of the MPI program was used. For the remote tests, data was striped across all of the storage servers and across the separate shares on each of the servers. The program sent 8 GB of I/O per process to the storage system.

There were 61 compute nodes in the cluster: 15 with 8 GB of memory and 46 with 4 GB of memory.

In order to measure performance, it was necessary to defeat I/O caching. To ensure that there was no client-side caching, three-fourths of the local memory for client program

data storage was used, and all tests were run at 8 GB of I/O per client. The total I/O per remote test suite exceeded 2 terabytes.

Local I/O

This test, which used a serial version of the MPI program, reported the performance of the RAID on each of the four storage servers. Two variables were used: S (size of record) and N (number of reads or writes). The test was run with a record of 1 MB (S=1,048,576 bytes); this ensured that the individual I/Os were large enough to achieve near-peak performance. The test used a number of reads and writes that assured a sufficient volume of data to overwhelm I/O caching.

The test was run twice on each of the four storage servers with N=8,192 of read/write (2x over-subscription of local memory, or 8 GB of I/O written). It was run twice on one server with N=16,384 read/write (4x over-subscription of local memory, or 16 GB of I/O written). The results show that 2x over-subscription of memory was enough to overwhelm the cache effects on the measured performance.

Table 2 shows the maximum and minimum server results. Complete results are shown in Appendix 1. Storage server 2 was slowest for both reads and writes. This is most likely due to the differences in configuration between the four storage servers. (Storage server 2 had one less spindle in its RAID than the other servers.) Correcting this would likely have a noticeable impact on the peak aggregate bandwidth results.

Table 2 Local I/O Minimum and Maximum Results

Minimum I/O	
Storage server 2	125.9 MB/sec write
Storage server 2	91.5 MB/sec read
Maximum I/O	
Storage server 3	149.7 MB/sec write
Storage server 4	94.7 MB/sec read

Remote I/O

This test reported results for the compute cluster nodes running an MPI program to create multiple I/O streams to the CIFS servers. The same I/O was used as was used in the local I/O test. An additional parameter, *m*, was added to set the amount of memory that the application on the cluster used for record storage. It was very important to make sure that the data did not get cached on the local compute nodes; therefore, the variable *m* was larger for runs with fewer processes on a compute node.

For the remote I/O tests, the data was striped across the four servers and across the shares on each storage server. (For more information about the data striping, see Appendix 4.)

Four processes per compute node

In the first set of remote I/O tests, there were four MPI processes per compute node. The 8-GB compute nodes were used first (with m=1,536 MB). In this test, a single compute node initiated four streams of I/O to the four storage servers. The number of compute nodes was then increased (2 nodes / 8 processes, 4 nodes / 16 processes, 8 nodes / 32 processes, 15 nodes / 60 processes).

Individual and aggregate I/O rates for the multiple streams of data from the four storage servers into 1, 2, 4, 8, and 15 compute nodes were measured. (Note: 16 compute nodes were not run because there were not enough compute nodes to run 64 nodes with one process per compute node.)

The test that used four processes per node was repeated on the 4-GB compute nodes (with m=768 MB) to make sure that the memory size of the compute nodes did not affect the performance. Tests were run for 1 node / 4 processes, 2 nodes / 8 processes, 4 nodes / 16 processes, 8 nodes / 32 processes, and 16 nodes / 64 processes.

Because the results showed that there were no appreciable differences between the 4-GB compute nodes and the 8-GB compute nodes, they were mixed in the next section, which measured bandwidth with one process per compute node.

Note that each storage server had two GigE connections, while each compute node had only one. All processes were active, and they shared a single GigE connection. The Ethernet switch was fully non-blocking.

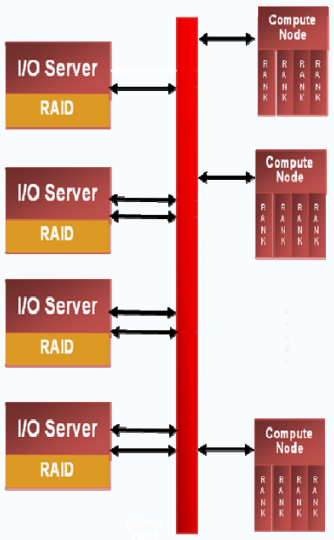


Figure 1 Remote I/O Testing, Four Processes per Compute Node

Figure 1, above, shows the remote I/O test setup for four processes per node. Table 3 shows the maximum aggregate and the per-server bandwidth achieved in the remote I/O tests with four processes per node. Additional test results are shown in Appendix 2.

Table 3 Remote I/O Results, Four Processes per Compute Node

Maximum Peak I/O 4 Processes per Compute Node	
32 processes, 8 4-GB nodes	139.2 MB/sec per server write (556.9 MB/sec aggregate)
64 processes, 16 4-GB nodes	109.2 MB/sec per server read (436.9 MB/sec aggregate)

One process per compute node

In the next set of tests, only one process was run on each compute node. The number of compute nodes, and therefore the number of streams to the four storage servers, was increased each time. The 4-GB compute nodes and the 8-GB compute nodes were mixed in this test.

The first set of data was gathered for four single-streams for each of the four servers. This is essentially four single-stream results gathered at the same time. The second set of data was gathered for two streams of I/O for each of the four storage servers; the third data set, for four streams for each of the four storage servers; the fourth data set, for eight streams for each of the four storage servers; and the last, for 15 streams per server for each of the four storage servers. (The University of Toronto cluster had a total of 61 compute nodes, so almost all were used in this final test.)

It is important to note that the remote tests were dominated by the slowest server because of the way in which the test was constructed with MPI. (In this study, the slowest storage server was server 2.) Note also that each I/O server had two GigE connections, while each compute node had only one. Only one process was active, and it had exclusive access to a single GigE connection. The Ethernet switch was fully non-blocking. Figure 2, below, shows the parallel testing setup for measuring throughput by compute node.

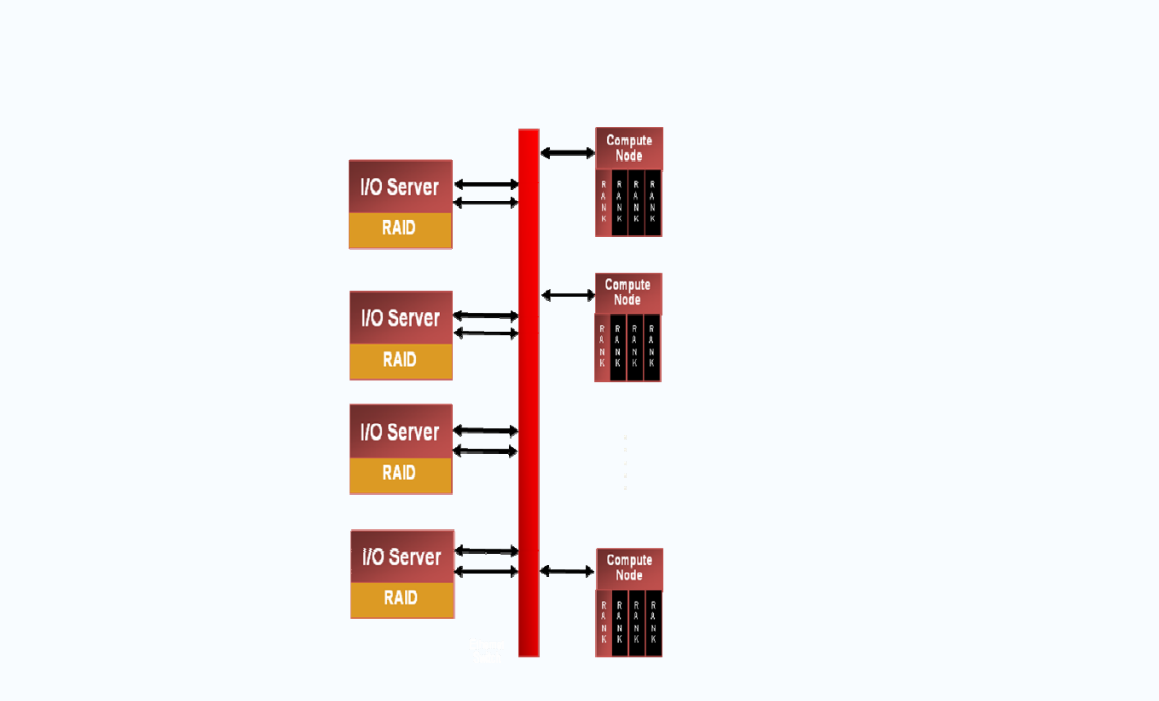


Figure 2 Remote I/O Testing, One Process per Compute Node

Table 4 shows the peak results of the tests for a single stream.

Table 4 Remote I/O Results, Single Stream Peaks

Maximum Peak Single Stream	
	85.9 MB/sec write
	48.9 MB/sec read

Table 5 shows the peak aggregate and the peak per-server results of the remote I/O tests with one process per compute node. Additional test results are shown in Appendix 3.

Table 5 Remote I/O Results, One Process per Compute Node

Maximum Peak I/O **1 Process per Compute Node**	
32 processes	147.2 MB/sec per server write (588.9 MB/sec aggregate)
64 processes	119.4 MB/sec per server read (477.7 MB/sec aggregate)

Discussion

In this study, remote I/O aggregate disk bandwidths in excess of 100 MB/sec per storage server were measured for each of the four storage servers used to gather data. The aggregate performance appears to scale linearly; the design goal of an aggregate data transfer rate of 600 MB/sec is therefore achievable with six servers.

The aggregate disk bandwidth appeared to be limited by the performance of the RAID on the storage servers. This could be verified by adding another string of RAID on the servers and rerunning the tests. It is also possible that the limiting factor was the network, the server operating system, or the CPU performance.

Because the single-server bandwidth often exceeded the 107 MB/sec speed of a single GigE NIC (as measured on the University of Toronto hardware using node-to-node MPI ping-pong test), it is likely that there was a benefit in using dual NICs. This could be verified in future testing by shutting off one of the NICs on each server and rerunning the tests.

It is important to note that the data transfer rates were achieved by ensuring that all I/O was spread across multiple shares on all of the servers. This layout does place the burden for achieving high performance on the users and administrators, who must ensure that the I/O is properly spread across the servers and their shares.

Conclusion

The strong single-stream I/O performance and the high sustained-aggregate I/O performance obtained in the University of Toronto study confirm that CIFS can provide a low-cost, high-performance global file system for small and mid-sized Windows-based HPC clusters. With aggregate performance peaking at over 100 MB/sec per server, the addition of two more servers should enable bandwidth to exceed 650 MB/sec for writes and 550 MB/sec for reads. This exceeds the design goal of 600 MB/sec by 50 MB/sec for writes, and is very close to the design goal for reads.

Performance of this magnitude using CIFS opens the door to high-performance remote I/O for low-cost clusters. CIFS is the default remote I/O file system for Windows; when combined with an easy-to-use HPC solution such as Windows Compute Cluster Server 2003, CIFS puts high-performance computing within reach of a wide range of researchers and commercial users.

Related Links

For more information about CIFS, see:
http://msdn2.microsoft.com/en-us/library/aa302188.aspx

For the Windows Compute Cluster Server Web site, see:
www.microsoft.com/hpc

For more information about Windows Compute Cluster Server technology, see:
www.microsoft.com/windowsserver2003/ccs/technology.aspx

For Windows Compute Cluster Server partner information, see:
www.microsoft.com/windowsserver2003/ccs/partners.aspx

For Windows Compute Cluster Server and HPC events, see:
www.microsoft.com/windowsserver2003/ccs/events.aspx

For the HPC Community Web site, see:
http://windowshpc.net

Appendix 1 – Summary of Results for Local I/O

Table 6 gives the results for local I/O for 2X memory over-subscription. This test used 8 GB (N=8,192) and a record size of 1 MB (S=1,048,576 bytes). This size was selected so that the results showed the rate of transfer of data to the disk, not just to the local memory.

Table 6 2X Memory Over-Subscription

	Sample 1		Sample 2	
	Write (MB/sec)	Read (MB/sec)	Write (MB/sec)	Read (MB/sec)
Storage server 1	136.5	103.4	133.3	102.9
Storage server 2	125.9	91.5	126.4	91.6
Storage server 3	149.7	94.2	142.6	94.0
Storage server 4	149.3	94.4	141.1	94.7

Table 7 gives the results for local I/O for 4X memory over-subscription. This test used 16 GB (N=16,384) and a record size of 1 MB (S=1,048,576 bytes).

Table 7 4X Memory Over-Subscription

	Sample 1		Sample 2	
	Write (MB/sec)	Read (MB/sec)	Write (MB/sec)	Read (MB/sec)
Storage server 1	134.1	102.5	133.8	77.3

Appendix 2 – Summary of Results for Remote I/O, Four Processes per Compute Node

In the remote I/O tests, the four storage servers were used, and there were four processes per compute node. Table 8 gives a summary of the results of the remote I/O for the 8-GB compute nodes, with 1,536 bytes of memory used by the application to store records.

Table 8 Results for 8-GB Compute Nodes, Four Processes per Compute Node

	Aggregate Results	
	Write (MB/sec)	Read (MB/sec)
1 node, 4 processes	118.9	67.8
2 node, 8 processes	238.6	123.2
4 node, 16 processes	465.0	206.7
8 node, 32 processes	573.1	314.7
15 node, 60 processes	505.9	431.8

Table 9 gives a summary of the results of the remote I/O for the 4-GB compute nodes, with 768 bytes of memory used by the application to store records.

Table 9 Results for 4-GB Compute Nodes, Four Processes per Compute Node

	Aggregate Results	
	Write (MB/sec)	Read (MB/sec)
1 node, 4 processes	119.3	68.1
2 node, 8 processes	233.8	117.8
4 node, 16 processes	461.5	207.0
8 node, 32 processes	556.6	310.1
16 node, 64 processes	504.5	436.9

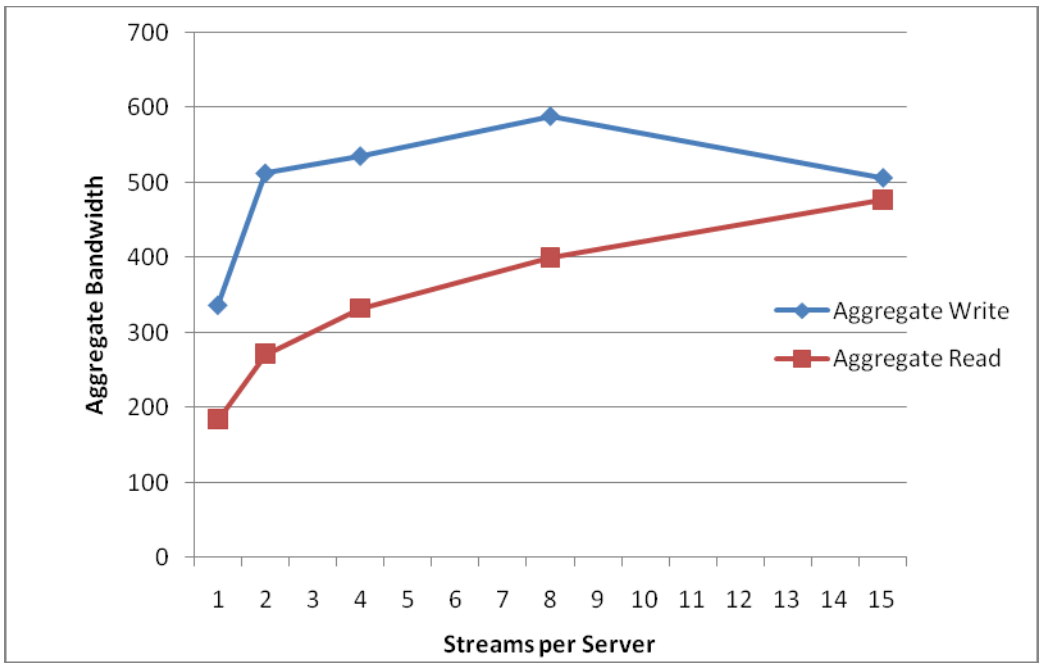
Appendix 3 – Summary of Results for Remote I/O, One Process per Compute Node

Table 11 gives a summary of the results of the remote I/O test with one stream at a time for mixed 4-GB and 8-GB compute nodes. The application used 6,144 bytes of memory to store records. Figure 3 shows a graph of the results.

Table 10 Results for One Process per Compute Node

	Aggregate Results	
	Write (MB/sec)	Read (MB/sec)
1 I/O stream/server	336.8	183.5
2 I/O stream/server	512.7	271.0
4 I/O stream/server	535.7	332.0
8 I/O stream/server	588.9	400.1
16 I/O stream/server	506.7	477.7

Figure 3 Results for Remote I/O by Streams per Server



Appendix 4 – Striping for Remote I/O Testing

Striping, the splitting of data across more than one server, was performed first by storage server, then by share on each server. There were four storage servers used, each with four shares. In the first test, each of the four processes went to share 1 on each of the four storage servers. In the next test, the eight processes went to shares 1 and 2 on the four storage servers, and so on through 15 compute nodes and 60 processes.

Table 11 Striping for Remote I/O Testing

Process	Storage Server	Share
0	1	1
1	2	1
2	3	1
3	4	1
4	1	2
5	2	2
6	3	2
7	4	2
8	1	3
9	2	3
10	3	3
11	4	3
12	1	4
13	2	4
14	3	4
15	4	4
16	1	1
17	2	1
18	3	1
19	4	1
20	1	2
21	2	2
22	3	2
23	4	2
24	1	3
25	2	3
26	3	3
27	4	3
28	1	4
29	2	4
30	3	4
31	4	4
32	1	1
33	2	1
34	3	1
35	4	1
36	1	2
37	2	2
38	3	2
39	4	2
40	1	3
41	2	3
42	3	3
43	4	3
44	1	4
45	2	4
46	3	4
47	4	4
48	1	1
49	2	1
50	3	1
51	4	1
52	1	2
53	2	2
54	3	2
55	4	2
56	1	3
57	2	3
58	3	3
59	4	3
60	1	4
61	2	4
62	3	4
63	4	4