

Transferring Large Files in Real-Time

Jeffrey Menoher, James Hope, Andrew Holmes, Robert Cooper, Ronald Mraz
Owl Computing Technologies, 38A Grove Street, Suite 101, Ridgefield, CT, 06877
{jmenoher, jhope, aholmes, rcooper, rmraz}@owlcti.com

Abstract

This paper presents empirical real-time throughput test data for writing large files (100 Gigabytes in size to a Terabyte) to disk arrays using a Windows 2003 server operating system installed on an Intel-based hardware platform. It is shown that standard Windows 2003 I/O function calls for writing to disk are effective for small and moderate sized files, but deteriorate in performance as file sizes grow beyond 100 Gigabytes in size. Disk writes of large files writing directly to disk arrays using the Win32 API offers performance improvement of 10X over standard Windows function calls for real-time applications that require consistent performance over the entire file. Disk write performance data is also presented for the Sun Solaris operating system for comparison.

1 Overview

The disk write process often proves to be the throughput limiting step in the overall process of transferring information from one network location to another.

As an example, if two networks are connected with a communication channel with a throughput capacity of 17 Megabytes/sec (OC-3), the disk write process must match or exceed this throughput rate in order to fully utilize the channel. During tests of an Owl Computing Technologies OC-3 data transfer system [1][2] for a customer who routinely transferred large files with sizes in excess of 500 Gigabytes, uncharacteristically low throughput performance was observed on a Windows 2003 server platform. On closer examination, it was observed that throughput performance was changing during the file transfer process; starting at expected performance levels and deteriorating over time.

The phenomenon of throughput slowdown was traced to memory management of disk writing functions in the Windows operating system, which were not optimized for a sustained use without allowance for periodic flushing of cached data. Problems with disk write caching for large files may be solved by accessing the Win32 API and writing directly to disk hardware in modulo 512 byte packet sizes [4]. While other researchers report throughput performance gains while writing to disk arrays using

the Win32 API [7][8], few researchers have examined the specific case of real-time disk write throughput performance for very large files. Microsoft reports NTFS file read/write speeds of 1.7 Gigabytes/sec using striped arrays of 32 disks [9]. Our tests are limited to striping across 4 disks in a RAID-0 array, available as a commodity server from Dell.

This paper presents empirical throughput test data for writing large files, in excess of 1000 Gigabytes in size, to disk using a Windows 2003 server operating system installed on a Dell 2850 Intel-based hardware platform. Disk writes using standard Windows File I/O function calls, and using the Win32 API non-buffered write are compared. It is shown that disk writes of large files using the Win32 API offers significant performance improvements over standard Windows File I/O function calls [4].

Disk write throughput data is also presented for a Solaris operating system [10] running on a Sparc platform for comparison. The Solaris example is throttled to a disk write throughput rate that roughly matches the Windows tests. No slowdown in write throughput was observed for the Solaris system.

2 Experimental Results for Windows File I/O

In this paper, the timing characteristics of disk writes are measured during the creation of a large files, in excess of 1000 Gigabytes. Test results for peak disk write times and disk write throughput using standard Windows File I/O function calls while writing a 1000 Gigabyte file are shown below in Figures 1 and 2. Initial throughput was set for OC-3 (155 Mbit/sec). The test time duration was 25.37 hrs. Note that peak disk write times increase with time, and throughput declines correspondingly. Tests were repeated at three thread Priority settings: Normal, High, and Real Time. Test results were found to be largely independent of Priority settings, with higher Priority settings resulting in slightly lower performance.

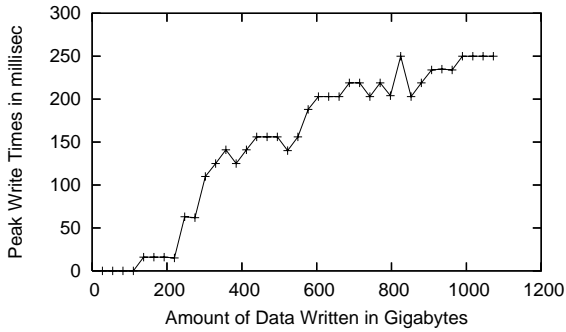


Figure 1: Peak disk write times for 800 Kilobyte data packet, Windows File I/O function calls

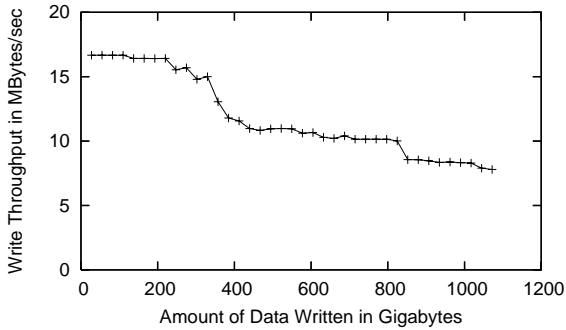


Figure 2: Disk write throughput as function of data written, Windows File I/O function calls

3 Test Configuration

Tests were performed using an Owl software application called createfiles, which was originally used to create arbitrarily large files for use in testing Owls DualDiodeTM one-way data transfer system. Application createfiles defines a software buffer of fixed size (in RAM), fills it with data, then writes it to a file (on hard disk) in append mode. A typical buffer size is 800 Kilobyte, which is an empirically determined optimum for performance.

Every time the buffer contents (in RAM) are written to the file (on disk), the amount of time taken to perform the write operation is logged. The amount of time is determined by logging timestamps before and after the write operation. The timestamps used have one millisecond resolution. A time delay may be inserted between each write of the buffer, using a variable software parameter InterPacketDelay. InterPacketDelay is one of two parameters used to adjust the throughput of the overall disk write process. For the test results shown in Figures 1 and 2, the InterPacketDelay parameter was set to 40 msec.

After a predetermined number of writes occurs, defined by the samplingInterval software parameter, the average and peak write times are calculated, as well as the overall data throughput during the sampling interval.

A typical sampling interval is 100 Megabytes; over one hundred cycles of writing the fixed-size buffer to the file.

A pseudocode rendering of the test algorithm is as follows:

1. load bufferContents (a single packet) into RAM
2. set IntraPacketInterval
3. set InterPacketDelay
4. initialize InterPacketCounter and other variables
5. record timestampInitial and write it to logfile
6. for (j=0; j<M; j++) /* loop over SampleIntervals */
7. record timestampThisInterval
8. initialize inner loop variables
9. for (i=0; i<=N; i++) /* loop within SampleInterval */
10. record timestampBeforeWrite
11. write bufferContents to disk
12. record timestampAfterWrite
13. calculate diskWriteTime
14. if (diskWriteTimeMax<diskWriteTime)
15. diskWriteTimeMax=diskWriteTime
16. end if
17. increment InterPacketCounter
18. if(InterPacketCounter=IntraPacketInterval)
19. sleep for time duration InterPacketDelay
20. reset InterPacketCounter
21. end if
22. end inner (i) loop
23. calculate diskWriteTimeAvg
24. calculate throughputThisInterval
25. write test results to logfile
26. end outer (j) loop
27. record timestampFinish
28. calculate throughputOverall
29. write test results to logfile

The amount of time taken to write the buffer contents to the hard disk varies significantly. Most of the time, the disk write process takes a small amount of time; often less than a millisecond. However, the operating system is occasionally burdened with overhead administration duties and cannot immediately service the write request. When this happens, individual write times may increase by several orders of magnitude.

Both peak disk write times and overall throughput are significant indicators of system performance. Peak disk write time determines the amount of cache buffer memory required to temporarily store data while the operating system is occupied with disk writing and other administrative tasks.

Experimental data is rendered as a series of plots of write throughput as a function of the cumulative amount of data written. Primary tests of disk write throughput were performed using a Windows 2003 Server operating system on a Dell 2850 (Intel) hardware platform equipped with internal hard disks. The hardware platform

contains a single 3.0GHz Intel Xeon processor, 800MHz front side bus, 2MB L2 cache, and 4GB RAM (DDR2 400MHz (4X1GB), Single Ranked DIMMs) [6]. The data directory comprises five large hard disks operating together as a RAID-0 (striped, non-redundant) entity. Each data disk is a Seagate model ST3300007LC with a 300 Gigabyte capacity operating at 10,000 rpm [5].

These tests were run to compare performance differences between file writing function calls as implemented using standard Windows File I/O, and using Win32 API disk writes in non-buffered mode. Secondary tests were also performed using a Solaris operating system for general comparison.

4 Throughput Throttling

The software application createfiles repeatedly writes the contents of a buffer to disk. Each instance of the buffer contents written to the disk may be considered an information packet. Createfiles may be configured to send a given number of packets in a burst, followed by an adjustable time delay. The number of packets in a burst is controlled by software parameter IntraPacketInterval. The time delay between bursts is controlled by software parameter InterPacketDelay. Together, IntraPacketInterval and InterPacketDelay are used to adjust disk write throughput.

Throughput depends on a number of factors, including operating system type, processor speed, disk controllers, presence or absence of RAID control, number of concurrent software processes, and others. In this paper, disk write throughput tests were performed using parameter settings compatible with throughput performance of Owl Computing Technologies DualDiode™ one-way data transfer systems, which routinely achieve transfer rates of 15 Megabyte/sec with an optimized RAID controller and striping.

5 Solaris Experimental Results

Tests were also run on a Solaris 10 Sparc V210 platform writing to a remote network storage device (NAS) capable of accommodating large files. Disk writes were throttled to approximate the same throughput as for Windows tests. The network was active during tests. The test time duration was 14.75 hrs. InterPacketDelay was set to 26 msec. No decline in throughput performance was observed over time on the Solaris platform.

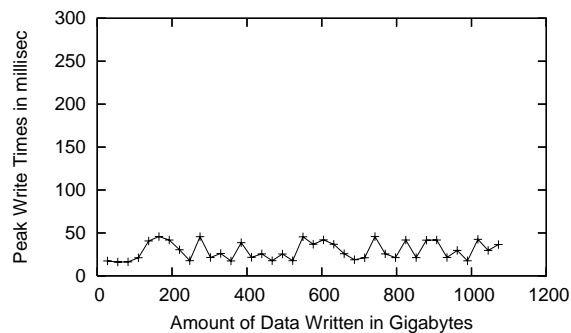


Figure 3: Peak disk write times for 800 Kilobyte data packet, Solaris/NAS

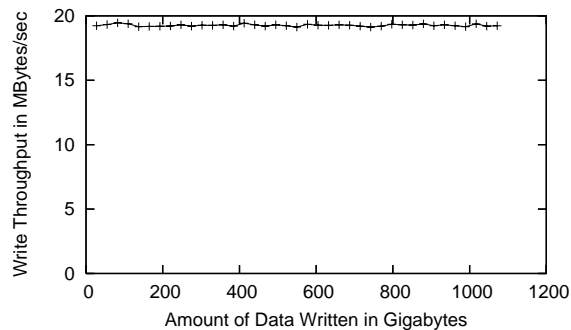


Figure 4: Disk write throughput as function of data written, Solaris/NAS

6 Direct Writes using Win32 API in non-buffered mode

The phenomenon of throughput slowdown was traced to memory management of disk writing functions in the Windows operating system, which were not optimized for a sustained use during Terabyte file writes. Problems with disk write caching for large files may be solved by accessing the Win32 API and writing directly to disk hardware in modulo 512 byte packet sizes in non-buffered mode. In the test results shown below in Figures 5 and 6, it is shown that disk writes of large files using the Win32 API offers significant throughput performance improvements over standard Windows File I/O function calls. The test time duration was 16.23 hrs. InterPacketDelay was set to 40 msec, with Priority set High.

7 Performance of Win32 API at higher throughput rates

Tests were repeated for higher throughput levels corresponding with OC-12 (622 megabyte/sec) and 1 gigabyte/sec Ethernet. At both throughput levels, disk write

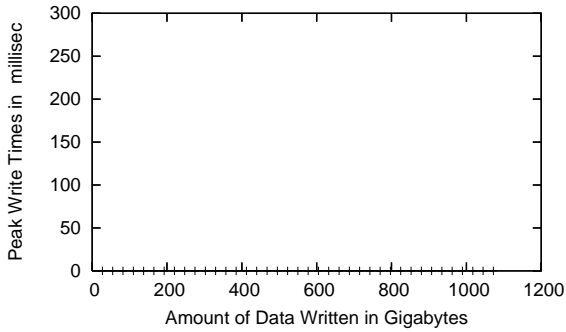


Figure 5: Peak disk write times for 800 Kilobyte data packet, Win32 API non-buffered Write

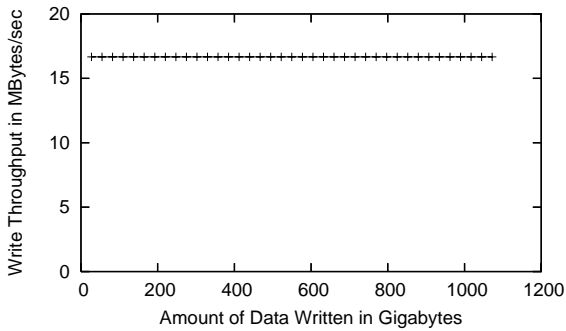


Figure 6: Disk write throughput as function of data written, Win32 API non-buffered Write

performance was greatly improved using the Win32 API non-buffered Write.

Disk write throughputs higher than OC-3 were achieved by sending multiple packets in sequence before introducing a time delay of reasonable size; several times that of the operating system clock interval. On the Dell 2850 test platform, the clock interval is 15 millisecond. The number of packets sent before introducing a time delay is controlled by software parameter IntraPacketInterval.

Test results are shown below for throughput rates approximating OC-12 and 1 gigabit/sec Ethernet.

7.1 Tests at OC-12 (622 Mbit/sec) Throughput

For OC-12 tests, 6 packets (each 800 Kilobyte in size) were written to disk before a 50 millisecond time delay was introduced. IntraPacketInterval was set to 6, InterPacketDelay was set to 50 msec, and Priority set High.

Results are shown in Figures 7 and 8. Note that the throughput performance difference between the two disk write methods is roughly a factor of 8 towards the end of the file write process.

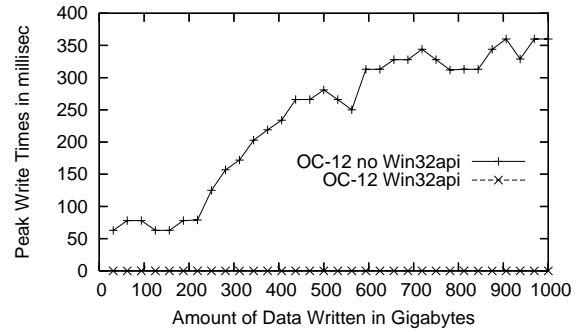


Figure 7: Peak disk write times at OC-12 for Win32API and non-Win32API

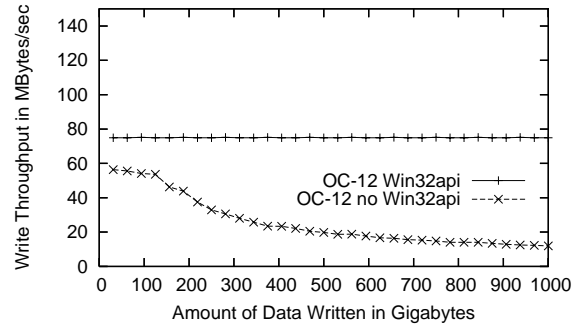


Figure 8: Disk write throughput at OC-12 for Win32API and non-Win32API

7.2 Tests at 1 Gigabit/sec Throughput

For 1 gigabit/sec throughput, 12 packets (each 800 KB in size) were written before a time delay of 50 milliseconds was introduced. IntraPacketInterval was set to 12, InterPacketDelay was set to 50 msec, and Priority set High.

Results are shown below in Figures 9 and 10. Note that the throughput performance difference between the two disk write methods is roughly a factor of 10 towards the end of the file write process.

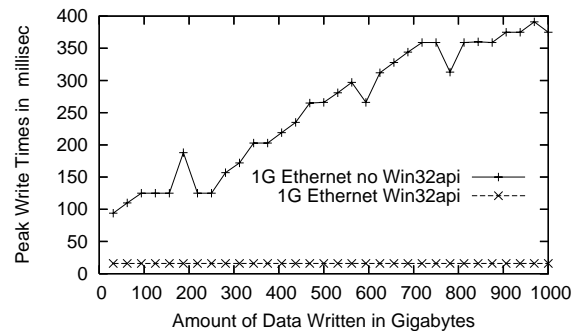


Figure 9: Peak disk write times at 1 gigabit/sec for Win32API and non-Win32API

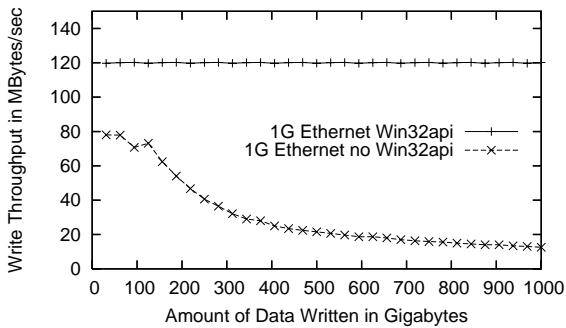


Figure 10: Disk write throughput at 1 gigabit/sec for Win32API and non-Win32API

7.3 Upper Throughput Limits

At throughput levels beyond 1 gigabit ethernet, peak disk write times become sensitive to small changes in the timing parameters for packet writing. In order to test the upper limits of throughput performance, throughput rate and peak disk write times were recorded for a variety of IntraPacketIntervals at a fixed InterPacketDelay of 50 millisecond. Consistent disk write throughput rates of 125 Megabyte/sec were achieved with peak disk write times of 16 millisecond, but beyond this rate, peak disk write times suddenly climb to several hundred milliseconds even while using the Win32 API in non-buffered mode, as shown below in Figure 11.

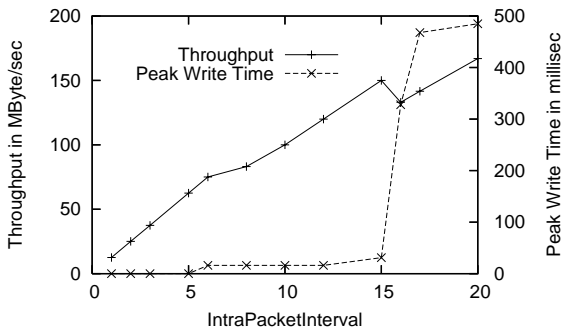


Figure 11: Throughput and peak disk write times as function of IntraPacketInterval Win32API

These test results indicate that a performance 'sweet spot' may be found, where throughput rate is high and peak disk write times are low. Similar tests using Win32 API in non-buffered mode at other values for IntraPacketInterval and InterPacketDelay parameters indicate that higher performance may be achieved. There is also evidence that additional performance gains may be achieved by adjusting the multi-thread time slicing characteristics of the Windows operating system [3].

8 Conclusion

During product testing on a Windows OS platform, we have discovered that the disk write process slows down over time when very large files are written; an instance of operating system interference. Performance problems may be solved by using a lower level API provided by the Windows operating system: the Win32 API. For disk writes of large files, in excess of 100 Gigabytes in size, writing directly to disk arrays using the Win32 API and writing directly to disk hardware in modulo 512 byte packet sizes in non-buffered mode offers significant performance improvements over standard Windows function calls for real-time applications. Throughput performance can be increased from 20 MB/sec to 120 MB/sec for 1000 Gigabyte files.

Furthermore, it appears that the overall file write process may be optimized by carefully tuning a variety of software parameters for maximum throughput. Tunable parameters include IntraPacketInterval, InterPacketDelay, and the Windows OS time quantum [3].

9 References

1. 'Secure Network Packet Transfer System (SNTS) OEM Installation Manual, and User Manual for TCP Configuration', (c)2006, Owl Computing Technologies, Inc.
2. 'Secure Directory File Transfer System, Cross Platform Interface (CPI), OEM Installation Manual and User Guide', Owl Computing Technologies, Inc.
3. Mark E. Russinovich, David A. Solomon, 'Microsoft Windows Internals', (c)2005, Microsoft Press
4. <http://msdn.microsoft.com/library>, Microsoft Developer Network website describing the Win32 API
5. <http://www.seagate.com/cda/products/discsales/marketing/detail/0,1081,641,00.html>, Seagate website describing technical specs for data disk drives
6. <http://www.dell.com>, Dell product information website
7. Erik Riedel, Catherine van Ingen, Jim Gray, 'A performance Study of Sequential I/O on Windows NT4', 2nd USENIX Windows NT Symposium, August 3-4 1998, Seattle, Washington
8. Leonard Chung, Jim Gray, Bruce Worthington, Robert Horst 'Windows 2000 Disk IO Performance', Technical Report MS-TR-2000-55, June 2000, Microsoft Research, Advanced Technology Division, Redmond WA
9. Peter Kukol, Jim Gray, 'Performance Considerations Gigabyte per Second Transcontinental Disk-to-Disk File Transfers', Technical Report MS-TR-2004-62,

July 2004, Microsoft Research, Advanced Technology
Division, Redmond WA

10. <http://www.sun.com/software/solaris/>, Sun Solaris
product information website