

Diskeeper 2011: Improving the Performance of SAN Storage

The Challenge of Managing Storage Performance

The task of maintaining peak performance is critical for the modern storage administrator. More and more mission-critical applications rely on the shared storage infrastructure for real-time activity, and enterprise-level businesses are graduating live workstations and production servers to SAN systems. Meanwhile, the potential for resource bottlenecks becomes more complicated due to the necessary interoperability of numerous software and hardware elements.

Admins have a variety of weapons in their arsenal to combat performance issues once capacity has been achieved; commonly, logs from servers and storage arrays can direct attention to timeouts, I/O retries, or failing elements in the array. I/O metrics provide a real view of performance, and investigation of what is happening to I/O requests down the stack at a granular level is essential.

The reasons behind performance loss can be numerous. Lack of multipathing, insufficient cache, inefficient virtual load balancing, all of these can be contributing factors (and certainly not mutually exclusive). However, there is one universal but often overlooked cause of performance loss and even the physical defects which may be cropping up themselves: file system-level fragmentation.

Fragmentation and SANs

The problem of file system fragmentation has been with us for decades. Direct-attached storage systems were plagued by it first, and then administrators of NAS solutions had to contend with it. Now SAN infrastructures are suffering its effects, inherited from the overlying file systems.

Understanding the Problem

"The reality is that although SAN vendors do an excellent job of optimizing the performance and reliability of the networking component that they control (the storage), the SAN, by the nature of its design, is server operating system agnostic. This means that the SAN has no control over how the operating system treats its storage. The SAN's role is to provide the storage; the operating system's job is to deliver that data in a manner that works best."

– Windows IT Pro, Maximize the Performance of your Windows SAN Infrastructure

The nature of a decoupled storage model precludes the operating system from being aware of the type of storage it's using. This greatly limits the operating system's ability to optimize for a particular storage model. One of the most significant potential issues, and probably the most unrecognized, is fragmentation in a storage system. With the implementation of a SAN in their storage environments, many Windows Server® administrators believe that file system fragmentation, which they accepted and dealt with when using DASD (Direct Attached Storage Device) storage, has gone away.

To better visualize the problem, begin with the closed system of the Windows® operating system. Within a workstation or server, any I/O request has a minimum number of I/Os required based on file size. One could consider this the “Shortest I/O Path,” with the greatest efficiency of data throughput.

File fragmentation generates excess I/O along the I/O path by increasing the number of searches required to complete an I/O operation. This directly results in slower data access as well as slower writes.

With the rise of abstracted storage layers, the problem of fragmentation is not miraculously eliminated – it is worsened. File system fragmentation also increases the amount of data accesses required to read/write at the block level due to how file pointing works between the OS and SAN. No matter how abstract you make the storage model, pieces of a file exist somewhere physically and each of those pieces is still associated, footstep by footstep, with the file representation within the operating system. I/O that occurs at the operating system level also occurs down at the SAN layer; excess fragmentation leads to more I/Os above as well as creating more I/Os below.

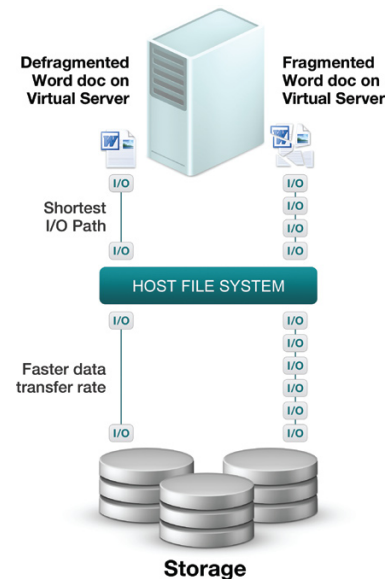


Figure 1 Fragmentation over SAN storage

Solving the Problem

While virtually every SAN implementation is pre-packaged with proprietary tools to address storage-level issues, none of these are equipped or able to handle file system fragmentation. Diskeeper® storage performance software is the first true solution for file fragmentation that really increases data performance over SAN infrastructures.

“We use it [Diskeeper] on our big SQL box (8-way processor, hundreds of gigs of space on a SAN, 16 gigs of RAM), and it has increased our disk performance by a factor of about 8 or 9. We were looking at adding more spindles to our SAN to help with some disk I/O issues we had, but this wonderful software did it for us.”

– Dave Underwood, Senior Engineer, CustomScoop

How Diskeeper Performance Technology Works

- **IntelliWrite®** fragmentation prevention technology eliminates up to 95% of fragmentation before it ever occurs. A revolutionary file system driver leverages and improves on the “Best Fit” file write logic of Windows, ensuring that files are sequenced properly at initial write. This incurs no additional I/O and prompts no additional writes in association with SAN-specific technologies such as Snapshots, CDP (Continuous Data Protection), or Deduplication.
- **InvisiTasking®** technology is specifically engineered to allow “background” applications to operate with zero impact/overhead on a system. It invisibly monitors CPU, memory and the significant hardware bottlenecks of the disk drive and network. It takes a proactive approach to instantly detect resource usage; taps into *unused* system resources (e.g., CPU and I/O that are almost never fully utilized); maintains complete granular control over its own activity, and *never* pre-empts users or services.
- **Instant Defrag™** technology dramatically minimizes I/O activity, and exponentially speeds up defragmentation. IntelliWrite delivers real-time information about fragments which were not prevented to the Instant Defrag engine. Instant Defrag immediately handles this residual fragmentation. Without the traditional need to run a time- and resource-intensive whole-volume fragmentation analysis, Instant Defrag can address the recently fragmented files as they occur. This dynamic approach prevents a buildup of fragmentation, which could incur additional I/O overhead to solve at a later date/time.
- **Efficient Mode** maximizes performance while minimizing disk I/O activity. By focusing on efficiency and performance as opposed to presenting a “pretty disk” visual display, Diskeeper 2011 minimizes negative side effects (e.g., reduced snapshot storage requirements or thin LUN growth, etc.) while maximizing performance benefits.

“Implementing Diskeeper on a SAN is simple and straightforward. Diskeeper Server configured for SANs arrives pre-packaged to deliver maximum data efficiency over SAN. Installation is straightforward, and can be easily done with Diskeeper Administrator. Just set up a defrag schedule that conforms to your production schedule and install.”

– Best Practices for using Diskeeper data performance software on Storage Area Networks (SANs)

Performance Testing

In order to evaluate the impact of Diskeeper 2011 over SAN storage, a battery of tests was run. The results showed that consolidation of logical files within the local disk file system (NTFS) means less total disk I/O generated and passed to the SAN. This increases efficiency of the SAN-attached OS and reduces I/O overhead.

Testing focused on what effect the Diskeeper storage performance optimization technologies had on two server models: file server with a standard production load, and SQL Server.[®] Iometer, an industry standard I/O measurement and characterization tool, was used to measure system performance while the SQLIO utility was used to generate Reads and Writes. SQLIO also provided IOPS, throughput and latency information. The test environment consisted of a SAN with two Windows Server 2008 R2 hosts connected to a disk array over a Fiber Channel interface.

Full methodology and test environment details can be found in the Appendix at the end of this document.

Test 1: Fragmentation impact on SAN disk performance using Iometer

This test involved running Iometer with a variety of access specifications that together modeled a mixed-use file server load. It involved random and sequential reads and writes with sizes ranging from 4KB to 32KB. The test host was a Windows Server 2008 R2 system with the test volume located on the SAN and formatted with 4KB cluster size. The test file was heavily fragmented and then performance was measured before and after automatic defragmentation by Diskeeper 2011. The result: performance of the defragmented file was over three times higher. Fragmentation levels were purposefully amplified in order to display the depth of performance loss that can be attributed to file fragmentation over SAN storage.

Details on fragmentation levels can be found in the Appendix at the end of this document.

Total I/Os per second			Average Response Time (milliseconds)		
Before Defrag	After Defrag	% Improvement	Before Defrag	After Defrag	% Improvement
31.26	100.35	221.02%	63.96	19.93	220.92%

Table 1: Iometer Test Results

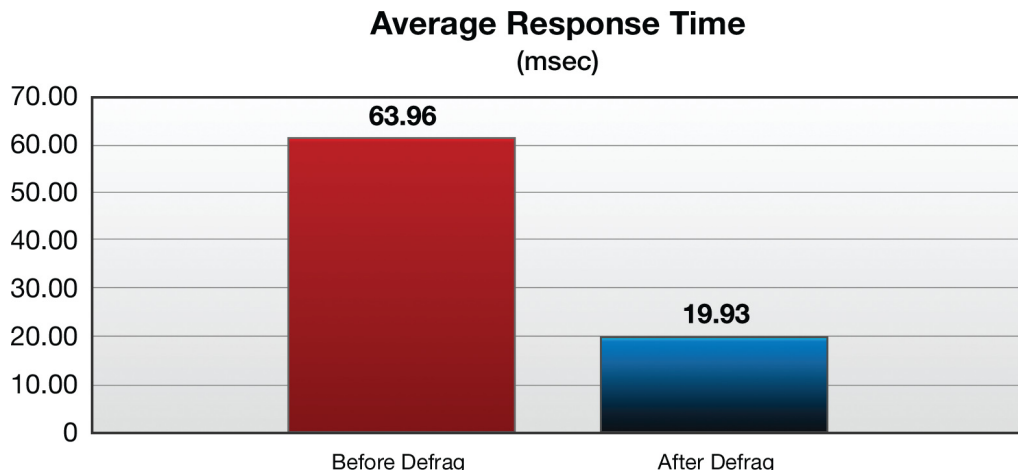


Figure 2: 220% Improvement in Average Response Time

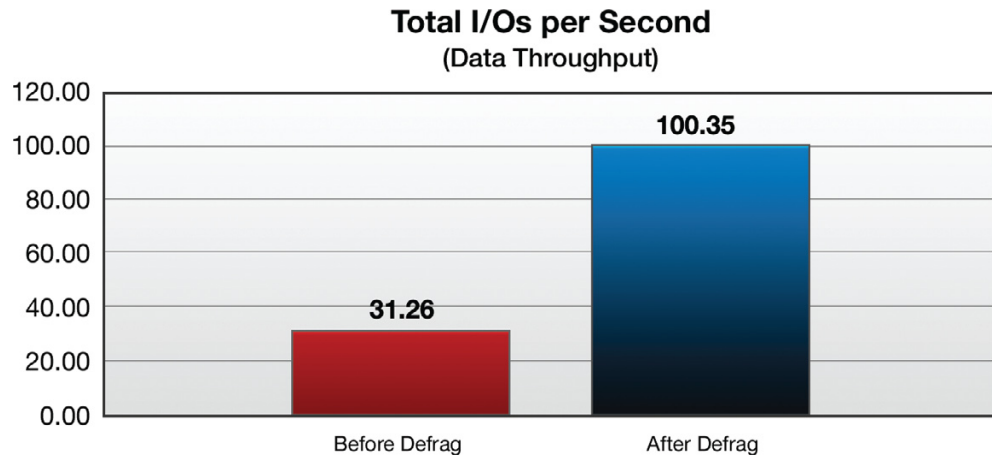


Figure 3: 221% Improvement in I/Os per second

Test 2: Fragmentation impact on SAN disk performance using SQLIO

This test involved running SQLIO (a Microsoft tool for measuring database performance) with a variety of access specifications. It involved random and sequential reads/writes of various sizes. The test host was a Windows Server 2008 R2 system with the test volume located on the SAN and formatted with 4KB cluster size. A test file was severely fragmented and then performance measured before and after automatic defragmentation. Performance improvement attained using Diskeeper 2011 ranged from 578% to 5,402%. Fragmentation levels were purposefully amplified in order to display the depth of performance loss that can be attributed to file fragmentation over SAN storage.

Details on fragmentation levels can be found in the Appendix at the end of this document.

Access Type	Data Set Size (bytes)	Before Defrag (I/Os per sec)	After Defrag (I/Os per sec)	% Improvement
Random write	64	7.34	49.73	577.52%
	128	3.69	43.36	1,075.07%
	256	1.84	34.56	1,778.26%
	Average	4.29	42.55	891.84%
Sequential write	64	7.58	51.79	583.25%
	128	3.8	45.59	1,099.74%
	256	1.9	36.35	1,813.16%
	Average	4.43	44.58	907.00%
Random read	64	21.06	122.48	481.58%
	128	10.97	107.18	877.03%
	256	6.36	89.87	1,313.05%
	Average	12.80	106.51	732.33%
Sequential read	64	31.67	1636.37	5,066.94%
	128	15.88	871.52	5,388.16%
	256	7.92	435.78	5,402.27%
	Average	18.49	981.22	5,206.78%

Table 2: SQLIO Test Results

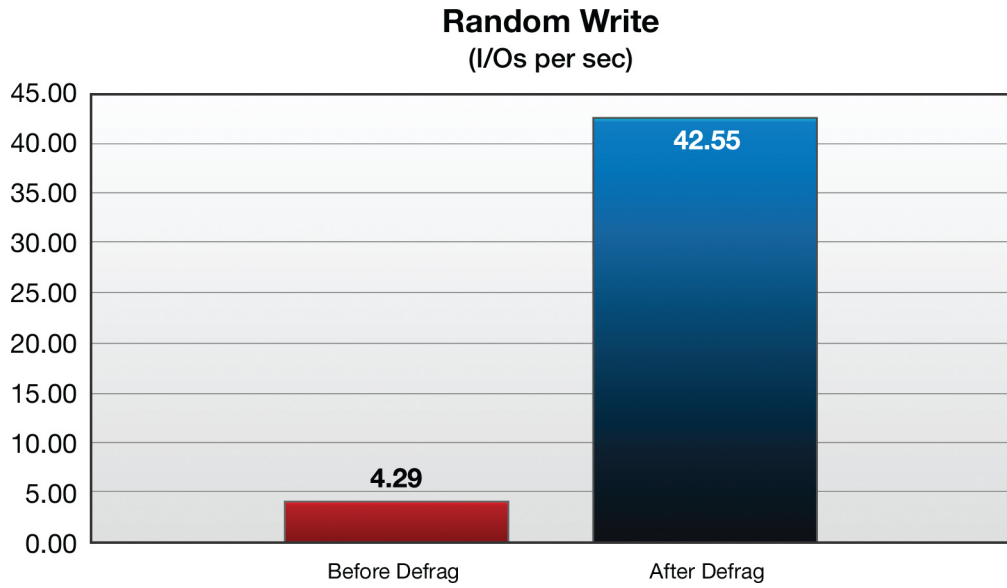


Figure 4: 891% Improvement in Random Write speed

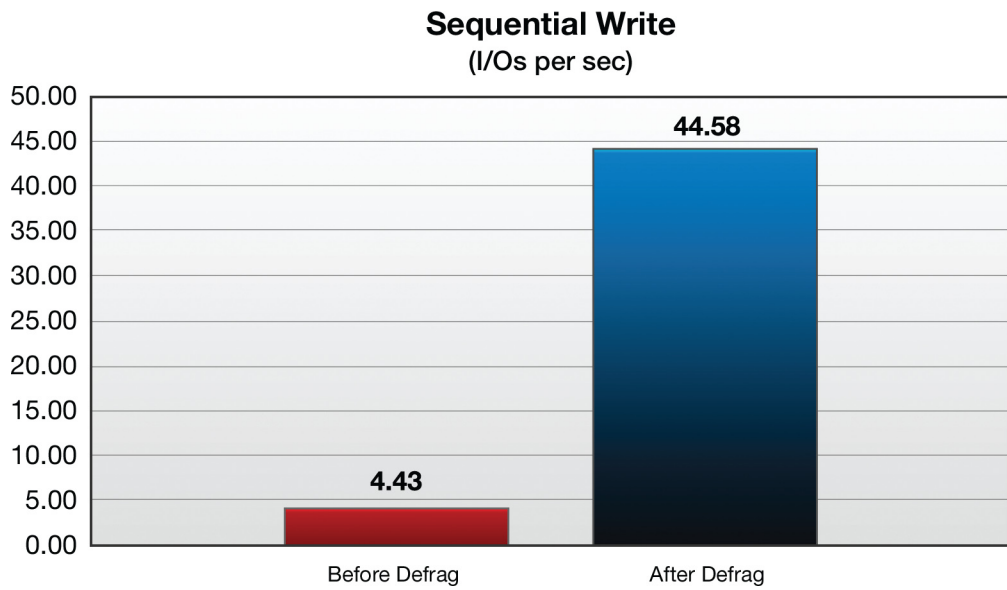


Figure 5: 907% Improvement in Sequential Write speed

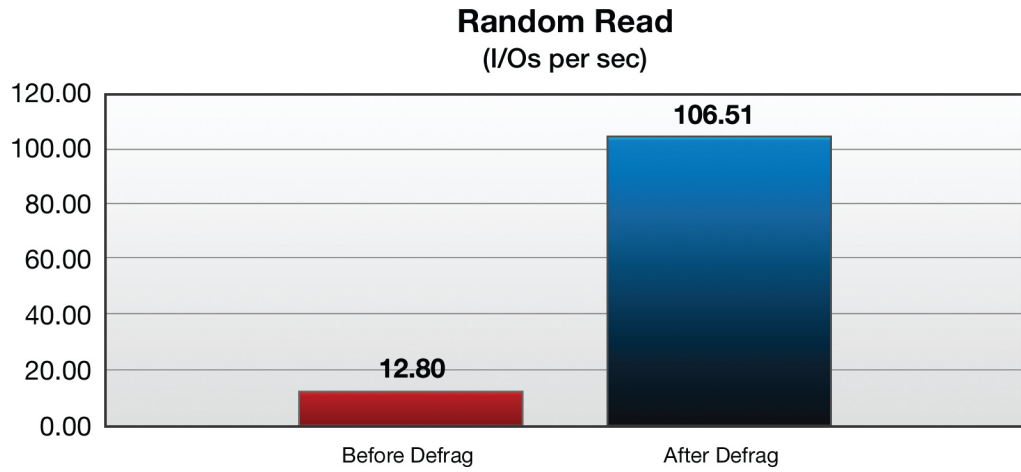


Figure 6: 732% Improvement in Random Read speed

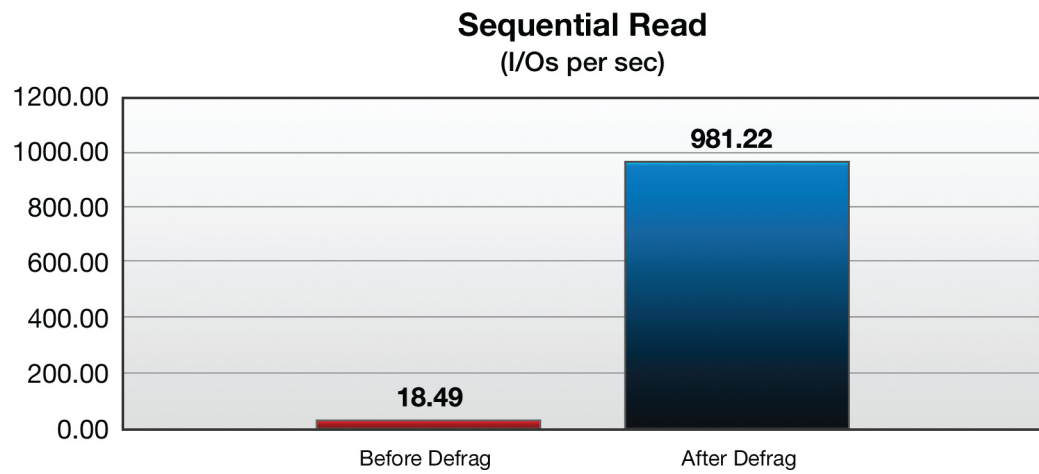


Figure 7: 5,206% Improvement in Sequential Read speed

Summary

Aggregate performance benefits exceeding 5,000% speak clearly of a need to handle fragmentation-related performance issues over SAN storage. While shared storage arrays have evolved how the enterprise manages resources from the ground up, defragmenting at the NTFS layer is still necessary to take advantage of the increased efficiency. Maximizing I/O efficiency increases storage hardware life and dramatically reduces energy costs because the infrastructure is working smarter.

Diskeeper 2011 optimizes I/O while catering to SAN-specific data redundancy measures. Increased data throughput shouldn't come at a hidden cost associated with defrag that doesn't throttle based on resource monitoring and prompts uncontrolled snapshot growth. Diskeeper Server configured for SANs ensures complete optimization without resource overhead or storage bloat.

Additional Reading

Inside IntelliWrite Technology:

http://downloads.diskeeper.com/pdf/IntelliWrite_Technology_brief.pdf

Maximize the Performance of Your Windows SAN Infrastructure (Windows IT Pro paper):

<http://downloads.diskeeper.com/pdf/Performance-Windows-SAN.pdf>

*Best Practices for using Diskeeper storage performance software
on Storage Area Networks (SANs):*

http://downloads.diskeeper.com/pdf/Best_Practices_for_using_Diskeeper_on_SANs.pdf

Appendix

This Appendix describes test procedures for evaluating Diskeeper 2011 performance improvements in a SAN environment. OS defragmentation offers two principal benefits for SAN attached systems:

1. By consolidating the logical file in the local disk file system (NTFS), less total disk I/O is generated and passed into the SAN. This increases efficiency of the SAN attached operating system(s) and reduces their I/O overhead.
2. In cases where a volume mapped to a LUN maintains a degree of association of logical orientation with physical placement, this can result in physically aligned related data as is a common benefit in DAS environments.

The tests demonstrate performance improvements from handling fragmentation in a SAN environment. Specifically, the tests show that a system affected by fragmentation performs less efficiently than a system on which fragmentation has been handled by Diskeeper 2011.

The tests described in this document rely on using several industry-standard benchmark tools.

Benchmark Applications Overview

Diskeeper storage performance software:

Diskeeper 2011 is primarily used to collect fragmentation reports for all test cases. IntelliWrite and automatic defragmentation should be DISABLED for all tests, except when Diskeeper 2011 is used to defragment the test volume. After defragmentation is completed, automatic defragmentation should be disabled again.

lometer:

lometer is the industry-standard I/O measurement and characterization tool to measure system performance. It creates a single test file: iobw.tst. The size of this file can be configured. While 20GB files should be sufficient, it is important to experiment.

It offers a wide variety of options to mimic reads and writes. It is best suited for replicating file server activity, but particular patterns can be used to represent access to a database. When using lometer, it is important to set up numerous “worker threads” to simulate a production-level use case with multiple concurrent reads and writes.

SQLIO:

SQLIO is CLI (command-line interface) only. It runs definable Read/Write patterns to a definable sized file. It provides IOPS, throughput and latency information. Full use information is included in Using SQLIO.rtf file located on the program root folder (part of the installation files).

The param.txt file dictates the volume the test file will be created on and the size of the file.

Example string inside param.txt: “d:\testfile.dat 4 0x0 100”

Below are some options. The “LS” parameter (Latency > System) is vital. Also important is to use a larger I/O request. Experimentation of various patterns will be required.

Option	Description
-o	Specify the number of outstanding I/O requests. Increasing the queue depth may result in a higher total throughput. However, increasing this number too high may result in problems (described in more detail below). Common values for this are 8, 32, and 64.
-LS	Capture disk latency information. Capturing latency data is recommended when testing a system.
-k	Specify either R or W (read or write).
-s	Duration of test (seconds). For initial tests, running for 5-10 minutes per I/O size is recommended to get a good idea of I/O performance.
-b	Size of the I/O request in bytes.
-f	Type of I/O to issue. Either “random” or “sequential.”
-F	Name of the file which will contain the test files to run SQLIO against.

Test Environment Setup

Testing is done on a SAN with two Windows Server 2008 R2 hosts connected to a disk array over a Fiber Channel interface (FCI). Each of the test hosts should have a 100GB test volume in addition to the system volume.

The following tools to create the environment are used in this test:

1. FragmentFile.exe

The following applications are used to benchmark performance in this test:

1. Iometer (mimics file server and, with various patterns, a Database)
2. SQLIO (SQL Server)

Environment Setup (needs to be performed on both hosts):

1. Install all tools and applications to the system volume of each test host
2. When installing Diskeeper 2011, make sure to **disable** IntelliWrite and automatic defragmentation (they are both enabled by default). Disabling these automatic real-time operating features of Diskeeper 2011 is done so that a steady state of the disk can be analyzed, as a baseline, before running Diskeeper 2011.
3. Configure Iometer for the test using an access specification with I/Os of different sizes and types. Specify a test file size of at least 20 GB. Use at least two workers. Save the test parameters in the Iometer configuration file – Iometer.icf.
4. Perform a trial test pass to create the test file. During the test run make sure the system is I/O bound. If necessary, adjust test parameters.

5. Create a batch file with a set of SQLIO commands with different access types to run a set of performance tests (i.e., SQLIOTEST.BAT). Edit the PARAM.TXT file to specify the name, location and size of the test file. Also ensure that at least two threads are used during the test. The test file should be located in the root of the test volume and its size should be at least 20 GB.
6. Perform a trial test pass to create the test file. During the test run make sure the system is I/O bound (i.e., using Performance Monitor). If necessary, edit test specification in the batch file.
7. After completing all installation steps, make sure to save the image of the test volume for repeated tests (using an imaging product like Norton Ghost). Full-volume block-based imaging software must be used so true comparisons can be made.

Test #1 – Fragmentation impact on SAN disk performance using Iometer

This test should be run from host 1. The first few steps involve measuring file performance on a fragmented file. The procedure is then repeated on the same file after defragmentation. At the end of the test, these results will be compared.

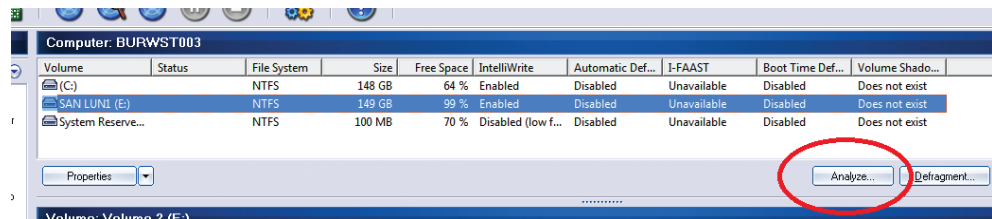
The test procedure is as follows:

1. Use FragmentFile.exe to create heavy fragmentation of the test file. From the command prompt, run the following command:

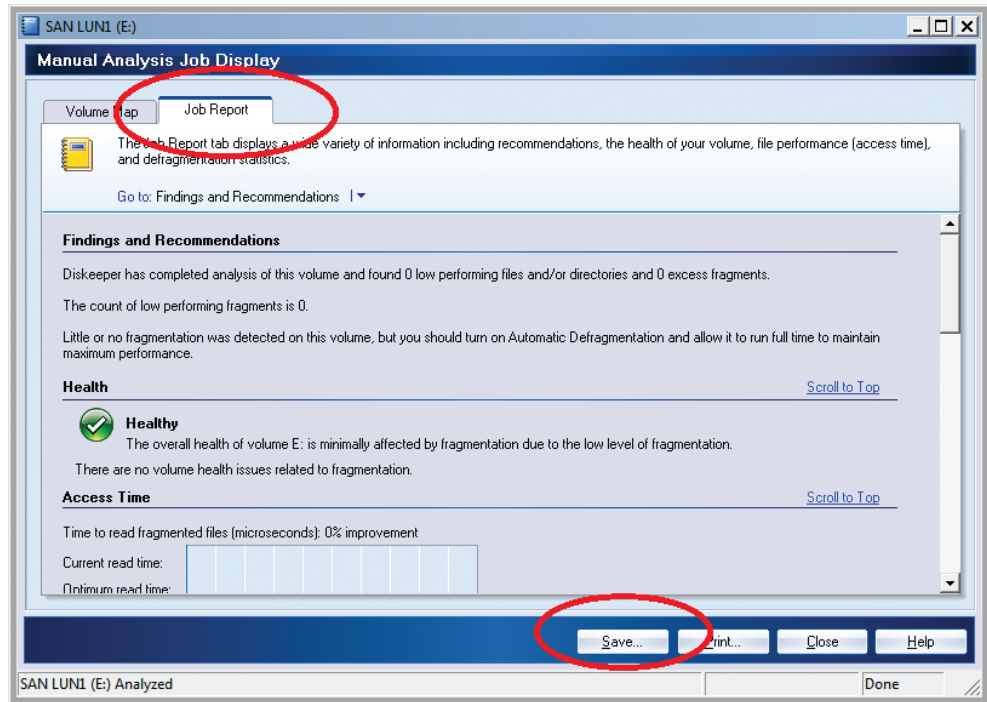
```
FRAGMENTFILE.EXE D:\IOBW.TST 1
```

D:\ needs to be replaced with the letter denoting the test volume. IOBW.TST is the test file used by Iometer and it is created in the root of the test volume.

2. From the user interface of Diskeeper 2011, perform a manual analysis of the test volume. Save the Job Report which shows the fragmentation state of the volume and the degradation it is causing on the read access times on the volume.



Highlight the given test volume(s) in the Volume Pane, and select "Analyze."



Note: there are two tabs available in the upper left-hand corner with data after the analysis completes. Save these reports by selecting the option in the lower right corner.

3. Reboot the test system to make sure the Windows cache is cleared.
4. Reboot the SAN host to eliminate any caching impact on the SAN.
5. Run the benchmark test using lometer, and record the results. From the lometer user interface open the test configuration file: lometer.icf. Specify the name of the results file, i.e., "Results_Fragmented.CSV." Then start the test from the lometer interface.
6. After the test is completed, results will be automatically saved in the results file. Also, save the screen shot with the final results from the lometer interface.
7. Use Diskeeper 2011 to defragment the test disk, including the test file.
8. Use Diskeeper 2011 to record the amount of fragmentation (save reports).
9. Reboot the test system to make sure the Windows cache is cleared.
10. Reboot the SAN host to eliminate any caching impact on the SAN.
11. Run the benchmark test using lometer, and record the results. From the lometer interface open the test configuration file: lometer.icf. Specify the name of the results file, i.e., "Results_Defrag.CSV." Then start the test from the lometer interface.
12. After the test is completed, results will be automatically saved in the results file. Also, save the screen shot with the final results from the lometer interface.
13. Now, compare the before and after data from lometer and Diskeeper 2011 analyses.

Test #2 – Fragmentation impact on SAN disk performance using SQLIO

This test should be run from host 2. The first few steps involve measuring file performance on a fragmented file. The procedure is then repeated on the same file after defragmentation. At the end of the test, these results will be compared.

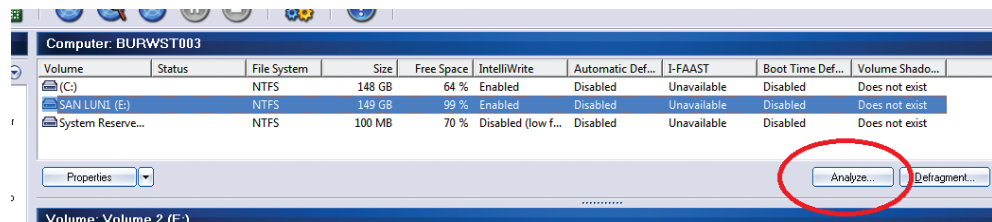
The test procedure is as follows:

1. Use FragmentFile.exe to create heavy fragmentation of the test file. From the command prompt, run the following command:

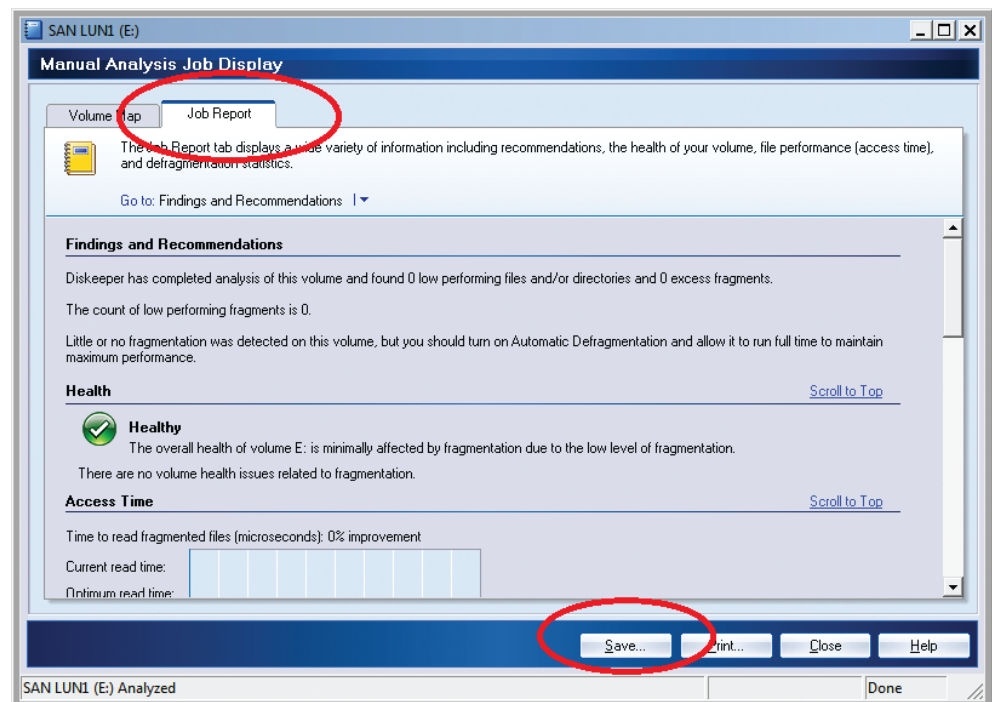
```
FRAGMENTFILE.EXE D:\TESTFILE.DAT 1
```

D:\ needs to be replaced with the letter denoting the test volume. D:\TESTFILE.DAT specifies that previously created by SQLIO test file in the root of the test volume.

2. From the Diskeeper 2011 interface, perform a manual analysis of the test volume. Save the Job Report which shows the fragmentation state of the volume and the degradation it is causing on the read access times on the volume.



Highlight the given test volume(s) in the Volume Pane, and select "Analyze."



Note: there are two tabs available in the upper left hand corner with data after the analysis completes. Save these reports, by selecting the option in the lower right corner.

3. Reboot the test system to make sure the Windows cache is cleared.
4. Reboot the SAN host to eliminate any caching impact on the SAN.
5. Run the benchmark test using SQLIO, and record the results. Redirect SQLIO output to a text file where the results will be saved. For example, run the following command from the command prompt:

```
SQLIOTEST > RESULTS_FRAG.TXT
```

6. Use Diskeeper 2011 to defragment the test disk, including the test file.
7. Use Diskeeper 2011 to record the amount of fragmentation (save reports).
8. Reboot the test system to make sure the Windows cache is cleared.
9. Reboot the SAN host to eliminate any caching impact on the SAN.
10. Run the benchmark test using SQLIO and record the results. Redirect SQLIO output to a text file where the results will be saved. For example, run the following command from the command prompt:

```
SQLIOTEST > RESULTS_DEFRG.TXT
```

11. Now compare the before and after data from SQLIO and Diskeeper 2011 analysis.

Test Fragmentation

Test #1: (IOMeter) Test Fragmentation

Volume size = 149 GB
Cluster size = 4 KB
Used space = 19,665 MB
Free space = 130 GB
Largest free space extent: = 121 GB
Total files = 18
Average file size = 1,086 MB
Most fragmented file = iobw.tst
File Size = 19GB
Total excess Fragments = 1,451,566

Test #2: (SQLIO) Test Fragmentation

Volume size = 150 GB
Cluster size = 4 KB
Used space = 10,342 MB
Free space = 139 GB
Largest free space extent = 130 GB
Total files = 12
Average file size = 856 MB
Most fragmented file = testfile.dat
File Size = 10GB
Total excess Fragments = 1,310,719