Best Practices for Eliminating Fragmentation with Modern Storage Technologies

(SAN/RAID/Virtualization)

Table of Contents

1. Preface 1
2. Overview 1
3. From A to Z; breaking down the I/O path 2
4. Defragmentation and On-Disk Technologies (HDD) 4
5. Defragmentation and Solid State Storage (SSD) 6
6. Defragmenting RAID 7
7. Defragmenting Virtual Hard Disks (VHD, VMDK) 10
8. Defragmentation and Storage Area Networks (SANs) 14
9. Defragmenting Network Attached Storage (NAS) 21

CONCLUSION 22

Appendix A 23

Appendix B 25

I. Preface

The definitions and explanations of storage technologies provided throughout this report focus on their specific use in the paper. There are new advancements and variations of these technologies beyond the scope of this paper's purpose and therefore, are not covered. This paper is also focused on Microsoft Windows and the Windows file systems.

II. Overview

Due to the significant complexity and breadth of the software and hardware used in modern storage environments, from disk-level technologies to massively scalable network storage facilities, there are many myths and misconceptions regarding the continuing need for a solution to fragmentation. Although, it is understandably easy to accept many of them as replacements for eliminating fragmentation as many seek to solve the same issue, the fact remains that the disk is the weak link.

From the time non-volatile storage was introduced decades ago, there have been layers of abstraction between the users/applications and those devices. The fact that modern datacenters go well beyond the single direct attached drive(s), and employ advanced storage infrastructures that add additional layers of abstraction, does not eliminate the need to solve fragmentation.

***Regular jobs to** defragment your disks will have a positive impact on performance. – HP*

In short, no matter how or where you store your data, solving file fragmentation is as vital for peak system performance and reliability as it has ever been.

The purpose of this paper is to briefly define these new storage technologies, how they contribute to I/O throughput, and how the various new solutions can work together with solutions to eliminate fragmentation for optimal disk subsystem performance.

As part of a discussion of the relevance of fragmentation and best practices for removing it, the two principal approaches for eliminating fragmentation must also be covered.

Historically the solution has been to allow the file system to generate the fragmentation and then, sometime thereafter, run a process to consolidate files and/or free space on the file system. A new and modern approach is to actually prevent file fragmentation from occurring in the first place, largely obviating the need for moving files after the fact. Both designs will be discussed, as they relate to the storage technologies covered.

If there is one key piece of data to remember after reading this document, it is that a disk file system is abstracted[1] from the actual underlying hardware and software that make up the storage subsystem. In other words, those same underlying disk systems (software and hardware) have no knowledge of what the file system is doing.

1. In the context used here it means: separated from other real or virtual components.

The first section of this paper will follow I/O from start to finish through some of the various layers of abstraction. The remainder of the paper will break out the various subtopics in detail, and offer recommendations and best practices for each.

III. From A to Z; breaking down the I/O path

Before covering complicated storage infrastructures, it's important to start from the simplest storage environment, the single direct-attached, hard disk drive.

In the beginning:

With any non-cached disk I/O[2] activity there is always a "block" involved. A block (sometimes referred to as a sector) is the smallest unit in which data is transferred to and from a disk device. A block is created by the disk drive manufacturer in a low-level format. It is a physical location, of a set size (typically 512 bytes), which has a unique address from any other block on the disk. No matter what other technologies are layered on top of the disk to access data, the block always exists as the smallest indivisible unit.

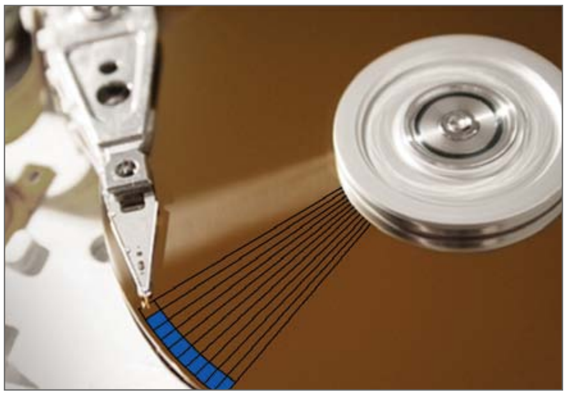


Figure1.0: Sectors on a disk formed into a 2KB cluster

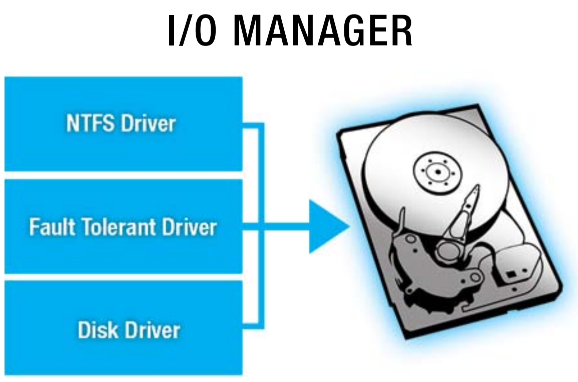


Figure1.1: I/O path from OS to physical disk

For a disk device to connect into the computer's system bus, it must use a host bus adaptor (HBA). That HBA is often built in to the motherboard for SATA and EIDE/PATA based disks. The HBA is hardware that extends (i.e., the adaptor part of the name) the controlling computer's (the host) circuitry (the bus) to other, typically storage, devices. The use of an HBA requires a software driver be loaded into the operating system.

The disk controller describes the firmware that controls a disk drive. It interprets a Logical Block Address (LBA) to locate data on a "block" (sector) level. Disk controllers require that a software device driver be loaded into the operating system to support two way communications in the form of I/O Request Packets[3] (IRP).

The file system, with respect to disks, is a high-level format mapping of logical units known as clusters. The file system uses an index to assign these logical clusters to file objects (e.g.

2. I/O (input/Output): refers to the data transferred from one device to another.
3. Kernel mode communication structure between device drivers and the operating system.

report.doc). In NTFS this data is stored in an index file called the Master File Table. Clusters are mapped to files by recording the logical cluster numbers (LCNs), in which the file is stored, in the index record for a given file[4]. A good analogy would be the index in the back of a book, directing you to the page number where a keyword is used.

A file system, such as NTFS, operates in Kernel mode[5] in the operating system. For each "logical" fragment in the file system, a separate disk I/O must be generated and passed on to the disk subsystem. Disk subsystems, no matter how intelligent the controller, operate at the block level and cannot recognize a file object. Therefore they cannot re-assemble or pool incoming I/O requests related to logical fragments and minimize the amount of physical motion. Here is the I/O path:

I/O REQUEST:

- Application requests to read a file

User mode

Kernel mode

- The request is passed to File System
- The File System maps the file clusters to an LBA and passes it to the driver (HBA)
- HBA driver maps LBA to particular physical disk in array
- Physical disk onboard controller maps the request to a specific block

The request then traverses this path back up, in sequence without skipping a step, to the application.

I/O RETRIEVAL:

- Physical disk acquires specific blocks
- Disk array controller acquires blocks from disk
- Blocks are mapped to LBA's and passed to the file system
- File system maps LBA to file clusters and passes to application

User mode

- Application receives file

Kernel mode

The last two kernel *mode steps for I/O retrieval are generally accessed direct from memory (DMA) into/out of the user buffer. However, the last step does typically occur on Windows, due to the design of Cache Manager.*

"I think defrag is an excellent tool for keeping your performance and the health of your drive up to par. So the larger these drives get the more data people are storing on them, and the more you store on them, you edit and save and pull it back up another day. It sort of gets spread out across the hard drive… so when you defrag you are pulling all of these files closer together. … he doesn't have to search over this 750G drive to find pieces of a file, they're all aligned…."

– Joni Clark, Product Manager, Seagate (as heard on the Computer Outlook Radio Show)

4. For a detailed explanation, see How NTFS Reads a File in the Reference section at the end of this paper.
5. The kernel defines the trusted core system component responsible for managing hardware operation requests (e.g. process time, disk and memory management). To run in kernel "mode" defines the execution of instructions at this level (ring 0).

IV. Defragmentation and on-disk technologies (HDD)

There are a number of technologies applied in modern hard disk drives (HDD), designed to improve data read and write performance. In this section we'll address them one at a time, and go into detail in each subsection.

A. QUEUING AND SEEK OPTIMIZATION:

There are a number of disk level algorithms to minimize the impact of physical limitations such as rotational latency (waiting for the disk to spin back around). They include variants of elevator-seeking and shortest-seek-first. These algorithms leverage the disk buffer, prioritizing retrieval of data physically closest, by measure of the data's cylindrical location and/or how close the requested data is to the current location of the disk head[6].

Seek algorithm optimizations tend to only care about minimizing head movement/rotational latency. As a result, an I/O in another part of the disk may wait for many seconds to be serviced, causing serious delays in getting data back to an application's now superseded request. Add the scattering of data (fragmentation) to that equation and an application may wait for a very long time to get the data it wants, while other applications get their data right away. That can then lead to that I/O starved application eventually getting full service from the disk, to the now detriment of some other more recent and more important application's request.

One of the next considerations that might come to mind is doesn't "disk queuing" eliminate the need to defrag?

Native Command Queuing (NCQ) is a technology that allows a SATA drive to re-prioritize and queue disk requests while completing others. It's kind of like multi-tasking at the disk level. The big benefit is that it excludes the CPU from having to be active in backlogs from what is the slowest component in the modern PC. SCSI disks have led the way, and continue to do so, supporting up to 256 queued commands.

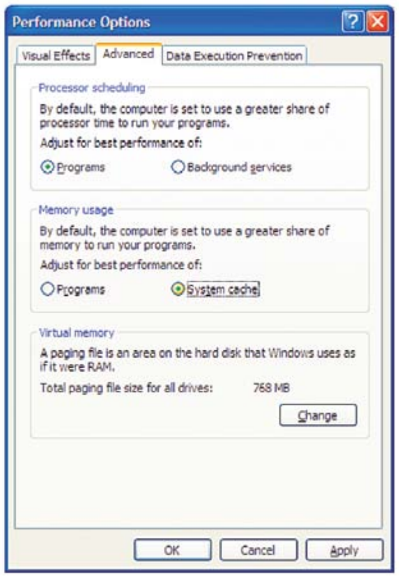
The answer to whether seek optimization and disk queuing eliminate the need to defragment, is simply no. While seek algorithms improve on rote data retrieval methods of the past, they cannot account for fragmentation as they are "block" based. They will organize and prioritize data retrieval based on physical location of data blocks, not per file object. Queuing will improve on prioritization strategies and improve overall seek time for asynchronous I/O, and synchronous I/O from multiple separate and simultaneous threads. However, queuing does not address fragmentation, as it too is block based and does not optimize activity for a particular file.

6. For more information on disk architecture see The Shortcut Guide to Managing Disk Fragmentation in the reference section.

B. DISK CACHE:

Caches are commonly available on the disk/disk array controller. This is a volatile memory, requiring constant power, through which data is stored temporarily (buffered en route to being written to the disk (write-back).

The cache can be of benefit for re-reading data that has been loaded into the cache (either from a recent read or write), as the cache duplicates data found on the disk platter. Reading from cache improves performance as it eliminates the need to retrieve data from the disk platter.



Performance Options

Visual Effects | Advanced | Data Execution Prevention

Processor scheduling

By default, the computer is set to use a greater share of processor time to run your programs.

Adjust for best performance of:

[✓ Programs] Background services

Memory usage

By default, the computer is set to use a greater share of memory to run your programs.

Adjust for best performance of:

Programs [✓ System cache]

Virtual memory

A paging file is an area on the hard disk that Windows uses as if it were RAM.

Total paging file size for all drives: 768 MB

Figure1.2: System Cache on Windows Server

Many controllers also offer read-ahead (pre-fetch) caching for sequential I/O. This attempts to pre-read blocks from the disk and place them into the non-volatile storage, ahead of the actual system request. Data located in physically non-contiguous blocks (i.e. due to file fragmentation) impedes read-ahead technologies, since disk devices do not map blocks to file objects. As a result they do not read in the proper data. For this reason, file defragmentation will aid controller read-ahead efforts.

The primary purpose for this on-drive cache is sequential read improvement; it does not offer significant benefit to random reads.

It should also be noted that the operating system maintains a system cache as well. This cache is housed in the computer's system RAM and allows an operating system to consolidate I/Os in a buffer (write-back cache), making for more efficient I/O. Recently accessed data can be retrieved from this cache, without need to request it from the disk, though of course to be read into the cache those fragments must first be read off the disk.

C. ELECTROMECHANICAL:

Serial ATA (SATA) drives are the new standard for desktop PCs and most laptops, while SCSI has been the mainstay on servers for over a decade, and Fibre Channel (FC) is becoming more common in SAN environments.

Today's fastest disks spin at 15,000 RPM with average seek times on these high-end drives measured in the mid 3 milliseconds[7]. While this is a great improvement over the 5200 RPM disks of the 1990's, it's a far cry from the operational speed of CPU and memory which have improved at a far greater pace and are now measured in nanoseconds.

7. Disk speed only affects rotational latency. 15K drives have a rotational latency of 8msecs; 1000/((15000/60)/2). Seek time is the average time it takes to move the head ½ the radius of the platter. Nowadays, vendors enhance seek times using seek optimization and caching to offset the actual time required by the heads to move ½ the radius of the platter.

High end SATA, serial attached SCSI (SAS) and Fibre Channel attached disks transfer rates move data through at a rate of 300MB/sec – 400MB/sec. Given that CPUs and memory can process 14 GB/sec+ of data (processor-to-system bandwidth), disks simply cannot keep up.

Keep in mind that the reported transfer rates already include all the technologies that optimize data access mentioned above. Some disk models are faster than others, and some interfaces allow greater throughput than others. The performance trade-off is typically price. Faster drives usually means higher costs.

SUMMARY:

Queuing and disk seek algorithms do afford improvements for inherent mechanical performance restrictions, but they simply do not make up for the fact that these devices cannot keep up with electronic speeds. If all data could be permanently maintained in high-speed RAM, fragmentation would not be the performance issue it is, but price-per-GB of RAM isn't affordable or appropriate for long-term or mass storage. It should also be noted that huge caches still do not help with file writes, which suffer in a fragmented free space environment.

When a "bottleneck" in a process (any process) is restricted, the entire process is significantly impeded. The greatest gain to be realized is the resolution of that bottleneck. Arguments that "today's disks are faster" and hence fragmentation is not an issue, hold no validity. Based on transfer rates alone, one can quickly conclude that the disk drive is still the weak link in the chain.

Recommendation: Prevent Fragmentation at the file system level, and use automatic background defragmentation for full performance benefit on SATA/SCSI/EIDE(IDE) hard disks.

V. Defragmentation and solid state storage

HYBRID HARD DRIVES:

Hybrid Hard Drives (HHD) offer a non-volatile RAM (NVRAM), in typically larger capacities than formerly available with on-disk caches. The added benefit is that the added memory capacity does not require a constant power source.

Hybrid drives can improve data access by essentially caching data on the attached NVRAM. However, the NVRAM is limited in size and durability, and data prepended to the NVRAM relies on a predictive algorithm that does well for commonly used data, but cannot account for all activity. Disk platter access is still necessary, even on only moderately active systems. Spinning up a "resting" disk will extend total seek time, more than disks that are not spun-down.

SOLID STATE DRIVES:

When solid state storage implements a hard disk drive interface (e.g. SATA), it is known as a Solid State Drive (SSD). NAND Flash is the predominant technology used in SSD, and also used in pocket storage devices like USB attached jump drives.

Solid state disks do not suffer from electromechanical read latency as do rotating magnetic media. While asynchronous read operations are efficient (hence fragmentation of files tends to be less serious), SSD/Flash is not the best choice for sequential I/O or heavy write activating due to erase-on-write requirements. Fragmented free space will dramatically impact write speed on these devices. This is true for both multi-level cell (MLC) and single-level cell (SLC) NAND Flash drives.

Recommendation: Defragment Hybrid drives on an occasional basis (e.g. once a day or once a week). Make sure to use an advanced tool that focuses on performance improvement, thereby not causing excess write activity to the NVRAM to achieve a “pretty disk display”. Optimize SSD disks with special programs designed specifically for NAND Flash (e.g. HyperFast®). Such tools will emphasize free space consolidation and defragment files to the degree they improve SSD performance. Tools built for HDD will generate unnecessary activity on SSDs, and are not recommended.

VI. Defragmenting RAID

Redundant Array of Inexpensive Disks (RAID) describes a software technology that pools multiple (two or more) physical storage drives for the purpose of data performance and in most definitions, data redundancy in the event of a physical device failure.

“Just as with a single disk, files stored on RAID arrays can become fragmented, resulting in longer seek times during I/O operations.”
– Microsoft

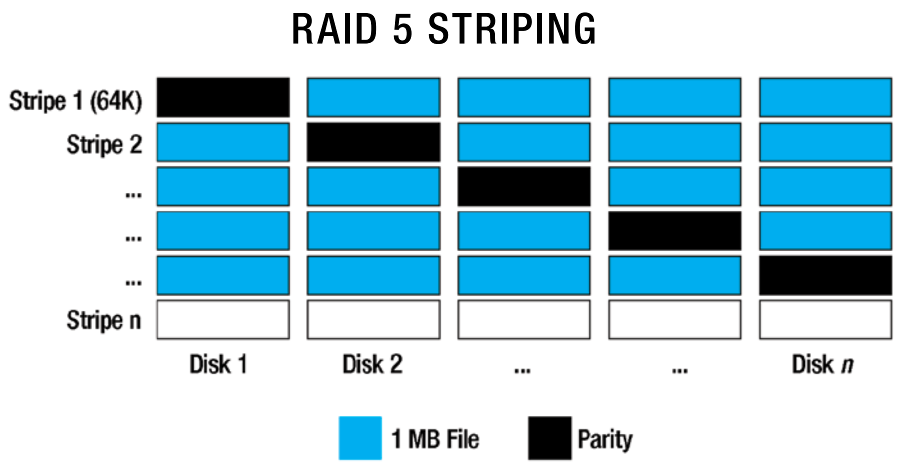


Figure1.3: 1MB file in one logical I/O, evenly striped (w/parity) across 5 disks

Multiple disks pooled together in a RAID array connect in to the host operating system the same way a single disk would, via a host bus adaptor. I/O in disk subsystems, wherein RAID is used, follows the same path as the single disk, but the data and the I/O load are physically distributed over many disks.

While a RAID array may be contained inside a computer system casing, it is more common for server operating systems to have the drives kept externally in disk arrays – especially when more than a half dozen or so drives are pooled together.

A *disk array* is a physical storage container with power supply that contains multiple hard disks, a disk controller card for the array, with a cache, and as a standard, offers disk striping and fault tolerance (i.e. RAID software).

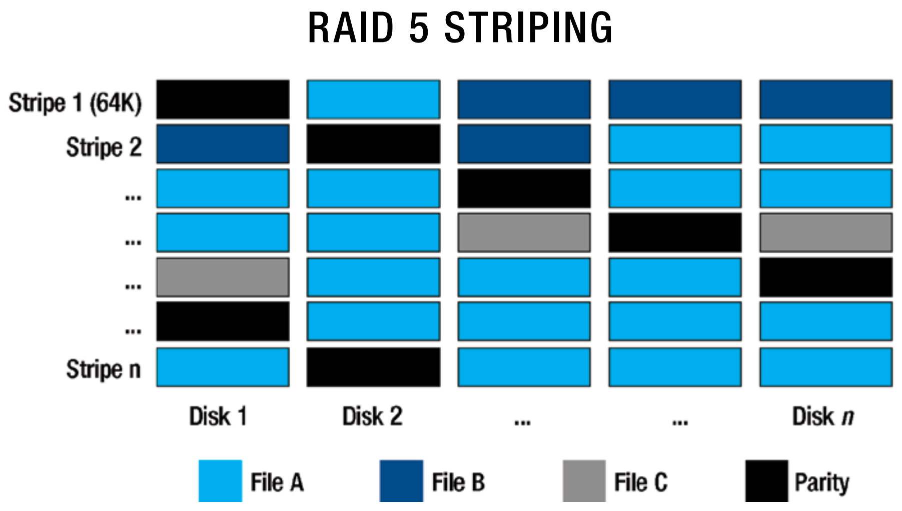


Figure1.4: 1MB "File A" unevenly written as simultaneous I/O writes are coalesced

Fragmentation degrades the performance of a RAID environment due to the unnecessary generation of I/Os issued by the file system. All these numerous and separate I/Os are then passed to the RAID controller to process (read or write). The RAID controller, unaware that the multiple I/Os all map to the same file, treats each I/O as a separate entity. A file object split into multiple I/Os is more likely to be interspersed with other disk I/O in a RAID stripe, than if the file I/O delivered to the controller was single.

Defragmenting files at the file system level and consolidating data into a single I/O can, depending on the RAID controller, better fill the entire (e.g. 64K) chunk (RAID stripe) size with that I/O; now taking full advantage of the RAID.

If a logically fragmented file, does get interspersed with other I/O (due to the fact that multiple I/Os have to be generated to write a file), it is theoretically possible that the data for that file is not evenly spread across the disks. Following that theory, the possibility of uneven file writes is increased on busier disks using smaller stripe/chunk sizes.

Figures 1.3 and 1.4 depict what can occur. This is for illustrative purposes only, as each stripe can, of course, contain data from multiple files all combined into a single stripe.

Rewriting of parity data is possible with disk defragmentation. For that reason, it is important to have I/O-sensitive defragmentation technology. This will prevent I/O overhead at the controller cache.

The type of distribution methodology also makes an impact in the layout of data. Proprietary metadata in the form of a tree structure may be used to map out data within a storage subsystem. In other cases simple correlations are created. Where vendors claim that there is no "fragmentation" this refers to the metadata (or other methodology) used for file layout within the storage system. This does not refer to fragmentation at the disk file system level (NTFS).

***…due to the sequential** nature of backup data, prefetching is often beneficial. Restore operations (reads) will profit from the use of prefetching, provided that the file system is not too badly fragmented…*

– EMC

ADVANCED/INTELLIGENT CONTROLLERS AND CACHING:

In a RAID device, read-ahead and write-back caching is also done at a block (not file) level. Read-ahead is valuable for sequential reads, and advanced technologies apply read-ahead in an intelligent manner. As was noted earlier, the issue is that if the file object requested by the operating system is not block-contiguous, read-ahead caching will not operate effectively.

Write-coalescing describes technology used by the RAID software's controller cache to buffer large sequential incoming data, (e.g. in a FIFO method), until an entire stripe's worth of information is gathered. The buffer also supports the generation of parity data prior to writing, in order to avoid a mechanical penalty for writing parity on-the-fly. The buffer is; of course, block based, waiting for enough I/O write data before it stripes the data (and parity) across the disks. It does not natively coalesce data from a single given file object. Defragmented files and free space can improve the performance and viability of write coalescing, by increasing the likelihood of sequential I/O writes.

***Physical members [disks]** in the RAID environment are not read or written to directly by an application. Even the Windows file system sees it as one single "logical" drive. This logical drive has (LCN) logical cluster numbering just like any other volume supported under Windows. … fragmentation on this logical drive will have a substantial negative performance effect*

– Diskeeper Corp.

ADJUSTING QUEUE DEPTH:

While disk level queue-depth mitigates taxation of the CPU to some degree, it still bottlenecks the disk itself, so one must be careful when modifying this function. Done correctly it can increase performance, but done improperly it can be damaging and impact available resources for other devices on that same HBA, decreasing the throughput for those devices. As an example, it is highly recommended to monitor and adjust SCSI queue depth in virtual machine environments to accommodate the increased disk activity that multiple VMs generate. Setting the value too high, however, can expose data in queue buffers to corruption, SCSI time-outs, and cache flushes.

SUMMARY:

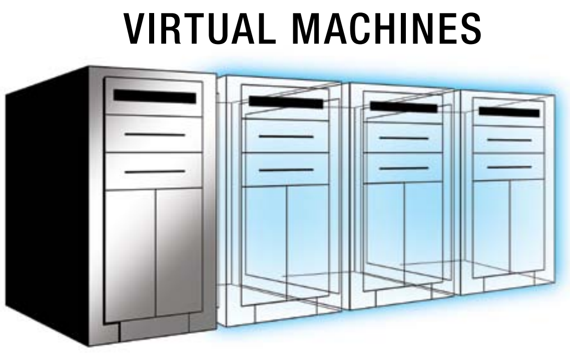
For defragmentation to be of benefit, it is a mistake to think that the end result must provide a one-to-one logical-to-physical correlation. In the case of fault tolerant or I/O distribution efforts of RAID or virtualization, it is standard for the data to be physically split across multiple disks. Defragmentation never forces a one-to-one mapping, nor does it need to in order to provide benefit. However, a single I/O delivered to the RAID controller, for a given file, is more likely to be optimally striped across the RAID disks than multiple I/Os to a file interspersed with other file I/O.

***If an application has to** issue multiple "unnecessary" I/O requests, as in the case of fragmentation, not only is the processor kept busier than needed, but once the I/O request has been issued, the RAID hardware/software must process it and determine which physical member to direct the I/O request.*
– Diskeeper Corp.

Recommendation: Prevent Fragmentation at the file system level, and use automatic background defragmentation for full performance benefit on RAID arrays.

VII. Defragmenting virtual machines/systems (virtual hard disks)

Machine virtualization describes the creation of one or more isolated virtual instances of a "guest" operating system either on top of a "host" operating system (Hosted Architecture) or directly on top of a specialized thin software layer called a hypervisor (Hypervisor/Bare-Metal Architecture).



In either architecture, the host system's virtualization of other operating systems is accomplished by software, proprietary to the vendor (e.g. Hyper-V™, ESX™), which resides between the physical hardware (CPU, memory, etc.) and the "guest" operating systems. Each guest or host operating system runs its own applications independently, as if it were the only system operating on the hardware.

Popular virtualization platforms such as those from VMware, Citrix, Microsoft, and others offer a variety of proprietary "virtual disks" that can be used with their platforms. The choice of virtual disks is a decision made on business needs, but does not affect the performance loss associated with fragmentation. However, the choice of virtual disk does affect how you need to solve fragmentation.

USING RESOURCE SENSITIVE TECHNOLOGIES TO THROTTLE ACTIVITY:

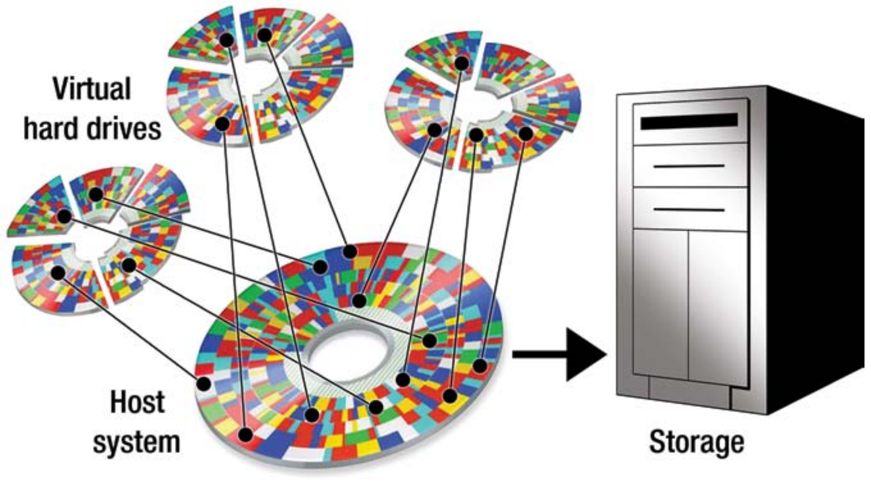
One of the technologies offered by defragmenters is resource (I/O, CPU) throttling. The concept here is to monitor, for example, disk I/O traffic, and throttle application related disk activity (i.e. defrag) until the I/O pipeline is free. Collection of resource data, such as I/O queues, is dependent on counters returned from the operating system on which the application is installed. In cases where multiple virtualized operating systems share common hardware (disks), as in a VMware's ESX/vSphere or Microsoft's Hyper-V, I/O throttling techniques will not function appropriately. The case being that an application using I/O throttling in one virtual guest operating system may detect that a hardware array is not busy, but a second guest operating system on the same hardware platform may be processing a very disk intensive operation. In that event disk contention would occur, bottlenecking the disk intensive process originating from the second guest VM.

Specialized programs, such as V-locity™, are designed to coordinate defragmentation across all VMs on a given hardware platform, eliminating potential resource contention issues.

VIRTUAL DISKS/PARTITIONS:
Hypervisors implement parent/root and child partitions. The parent partitions host the hypervisor itself, and the child partitions hosts the guest operating systems, using some form of virtual disk.

Most hypervisor virtualization platforms also have a disk type that allows the guest OS to bypass the host file system e.g. NTFS or VMFS. This pass-through/raw disk essentially allows the guest OS direct access to the storage system e.g. a physical disk/disk array.

Commonly used, given the generally broader array of features and functionality available, are virtual hard disks container files. Two popular data file formats are VMDK (virtual machine disk), used by VMware solutions, and VHD (virtual hard disk) used by Microsoft, and licensed by Citrix. In addition to the data container file, a virtual hard disk has small accompanying descriptor/configuration files.



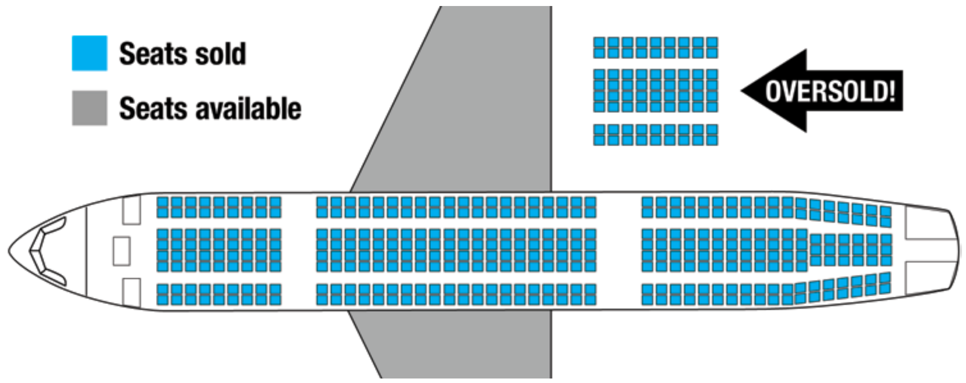
Virtual hard disks map to the virtualization platform's root file system and are stored there as files, for example VirtualFileandPrintSrv.vhd. That root file system then maps to the storage system.

These virtual disk formats support what is called fixed/thick virtual disks. In this case the guest OS provisions the entire logical volume(s) all up front. For example, a Windows Server Guest OS is given a 10GB allocation. This translates to a roughly 10GB virtual hard disk file that resides on the host file system. There are benefits to this format, but it can also lead to over-provisioning, a situation where more space is reserved for a system than it may ever use. In other words over-provisioning can lead to waste.

While the logical volumes of a fixed/thick disk are fully allocated, it does not mean that the physical storage under the host file system is fully allocated at the time of creation (with the exception of eager zeroed thick disk in VMware). That means that any type of file movement on most fixed/thick disks can translate to new physical allocations on the disk/disk array. This is not an issue in and of itself as a fixed/thick disk has by definition, the physical storage space available, but it is something to be aware in certain environments.

Another type of virtual disk is called the dynamic or thin virtual hard disk. The difference between a fixed/thick virtual disk and a dynamic/thin disk is that these types of disks can grow, as needed, when needed. This just in time growth occurs with the data file (VMDK, VHD) at the root file system, and simultaneously in the physical storage.

Over-subscribing storage means that dynamic/thin disks, as a collective whole, may potentially grow larger than the host file system can support. A good analogy is an airline that sells more tickets for a given flight than there are seats on the plane (oversell). The assumption is that there will be some no shows or cancelations. It is risky, but increases the likelihood of filling all the seats with customers.



The ability to oversubscribe combined with the just-in-time provisioning afforded with dynamic/thin virtual disks eliminates guesswork in pre-determining how large a guest virtual disk will grow, and the potential waste of space on a host file system. Needless to say using a dynamic/thin type of disk requires very close administrative monitoring to ensure that resources (space on that host file system) are always available for the guest systems. Unpredicted or unnecessary growth in such a virtual disk can be very problematic, so it is important to understand how the applications/usage that may cause growth in these guest systems behaves.

SNAPSHOT, LINKED CLONES AND REDO/DIFFERENCING DISKS:

In a storage context, a "snapshot" is a stately point in time capture of data. Storing a snapshot and making it available allows for recovery to a prior state, which effectively affords the ability to undo all changes after the snapshot was taken (i.e. rollback). On a virtualization platform, when virtual disk snapshot solutions are implemented the original disk becomes a read only disk, called a parent. A linked clone/differencing disk is essentially an enhanced version of the snapshot disk, allowing multiple VMs to share a common parent disk, where separate virtual disks (e.g. a linked clone) contain the unique change-data for each separate guest VM. To that effect linked clones/ differencing disks, become a space saver, as a 20GB OS image can be shared rather than redundantly duplicated in the host's file system; quite handy for desktop virtualization.

As noted, with these unique virtual disk types, all write activity to the read-only "parent" disk, is redirected to differencing disk(s), which are stored as separate files from the parent disk. These types of virtual disks cannot recognize defrag related write activity at the guest OS, to them it appears as new data writes that need to be captured. Therefore defragmentation creates unnecessary growth of the differencing disk, and can potentially delete that differencing data if thresholds are exceeded.

Defragmentation of a virtual disk with differencing data should only be performed prior to creating the differencing disk or following a procedure to merge the differencing/snapshot data back into the parent disk.

Also keep in mind that these virtual disk solutions do block level differencing, and will create significant fragmentation of files (as the source file is a read only copy in a parent) and changes to that file may be spread, block by block, across numerous virtual disk deltas. Performance can degrade over time with these disk types.

From an optimization standpoint, there are certain precautions that must be taken with differencing disks. Specialized programs, such as V-locity, that are designed to optimize virtualization platforms automatically recognize differencing disks and apply the recommended approach.

***Most virtual disk formats** do not provide a means to overlay actual virtual disk storage against clusters on the underlying host disk/file system. In other words, a request from the guest OS for cluster 1 on the virtual disk might translate to sectors 7-8 of cluster 80 in the virtual disk container file and sectors 1-6 of cluster 200 of the virtual disk container file.*

SUMMARY:

Machine Virtualization offers new challenges to traditional defragmentation routines, but it does not absolve the need to eliminate fragmentation. If anything, given the shared storage infrastructure, fragmentation (which reduces disk I/O efficiency) becomes more of an issue. This technology adds additional layers of abstraction adding to the "allocation tables that map to allocation tables that map to allocation tables…" translation between each storage layer. And, just as covered previously, none of these layers has the ability to map clusters back to individual files in the above source file system(s).

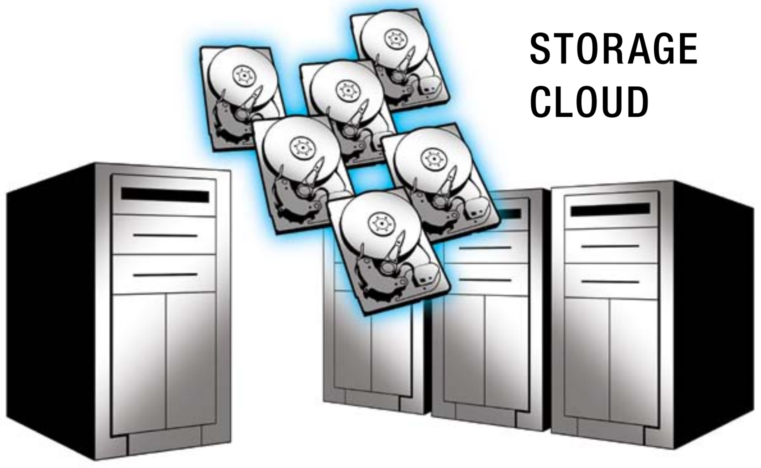
File system drivers that prevent fragmentation at a source (the file system) with file writes offer an excellent solution in a virtualized environments as they will not incur any added overhead from after-the-fact removal of fragmentation. They allow for more efficient pass through of I/Os to the underlying abstraction layers and ultimately the physical storage itself.

Recommendation: Prevent Fragmentation at the file system level in all instances, except as noted above. It is also recommended to undertake a proper evaluation of the environment to determine if defragmentation time frames are better suited to off-peak production hours for traditional defragmenters (so they do not interfere with business needs), or use a specially designed virtual platform disk optimization solution to automate and coordinate background performance maintenance. If you use dynamic/thin virtual disks, it is recommended to check with both your defragmentation software vendor, for proper configuration (e.g. disable file placement features).

VIII. Defragmentation and Storage Area Networks (SANs)

OVERVIEW:

A SAN affords the administrator the ability to make remote disks appear to be local. It does not matter what protocol they use to connect; iSCSI, Fibre Channel, etc.



SAN storage units are referred to as LUNs. Originally the term LUN (Logical Unit Number) only meant the SCSI disk address on an array for a particular disk, but it is now commonly used to represent the physical disk array when it is implemented in a SAN as a logical volume(s).

There are a great many tangential implementations in SAN technology so this section will focus on standard applications or usage. This does not, by any means indicate that a new, or proprietary technology eliminates the fact that fragmentation impacts performance, they simply aren't relevant.

The statement that a SAN volume appears local to the Windows operating system is a vital concept.

SAN STORAGE VIRTUALIZATION:

Storage virtualization involves the creation of a usually very large, logical-pool of data. Via software, that pool appears to be physically located all on one server. In actuality, that data may be located across hundreds of physical disks spread across dozens of servers. This is the concept implemented by Storage Area Networks (SAN).

This technology essentially abstracts “logical storage” (what the operating system sees and uses – i.e., the file system) from physical storage (the striped RAID sets). The key differentiator in virtual storage is that the multiple physical storage devices (e.g. a RAID array) are combined into one large grouping, on top of which a virtual storage container is created.

SAN file systems (a.k.a. cluster file systems) such as VMFS from VMware or EMC’s Celerra, are a third and different category of file system known as shared-disk file systems and are the backbone of storage virtualization (different from previously defined Local or Remote file systems). An operating system defragmenter, only recognizes the “local” disk file systems that it natively supports. Vendors of proprietary files systems typically include specialized technologies to optimize performance. These file systems are the foundation for storage virtualization.

***The SAS (Serial Attached** SCSI) software does not know what the total size of a file will be when it is created, therefore, it cannot be contiguously allocated. File fragmentation is a problem because additional file system activity must take place to access a file that is stored in multiple, noncontiguous locations on a volume. When defragmenting a volume, all the files on the volume are rearranged so that each file is in one contiguous extent.*

– HP

I/O MAPPING AND REDIRECTION:

Storage virtualization uses metadata to properly channel I/O. Software on a storage virtualization device (such as a SAN Switch) will translate logical disk locations to physical disk ones.

***The Cluster Shared** Volume (CSV) feature for Windows Server 2008R2 Hyper-V effectively allows NTFS to function as a cluster file system. Defragmentation is fully supported on CSVs.*

Here is an example:

1. A storage virtualization device gets a request for a logical location of LUN#1, LBA 32
2. It then performs a metadata lookup for that address and finds it actually maps to LUN#4, LBA16[8].
3. The device then redirects the request to the actual physical location of the data
4. Once it retrieves the data, it passes it back to the originator without the originating requestor ever knowing that the request was completed from a different location than what it knew.

The fact that there is not a one-to-one mapping of file system clusters to LBAs (due to LUN virtualization) is not an issue. Logical, file system level fragmentation causes the operating system to generate additional I/O requests to the virtualization software. Using metadata, the software then redirects I/O from the logical disk to its physical location.

The local disk file system (e.g. NTFS) does not know of, nor control the physical distribution or location in a virtualized storage environment, and as a result of fragmentation, NTFS has to make multiple requests regardless of the physical or virtualized storage environment.

8. With disk arrays often abstracting LUNs out of large RAID sets or RAID Volumes, multiple LUNs can be presented from a single stripe set and presented to different hosts.

In SAN file systems, block size (the smallest addressable virtual unit) is a configurable metric and varies based on the software used. Vmware's VMFS, for example supports 1MB to 8MB blocks.

Logical Cluster Numbers (LCNs) are a file system construct used to map a file on a volume to LBAs. Disk Controllers take those logical blocks and make the appropriate translation to a physical location. Disk controllers do not, no matter how "smart" they are, independently map fragmented file I/O into consecutive or linear block requests. They cannot "pool" incoming block-based data back into a file.

This means that regardless of the fact that the file system does not map directly to a physical location, file system fragmentation will create the exact same kind of phenomenon on RAID as it does on virtualized storage (multiple RAID arrays group together).

SANs can offer extremely efficient and high performing data storage, but it is not the job, nor within the scope of ability for a SAN system (hardware or software) to address file system level fragmentation. Proprietary technologies employed by one vendor can be more efficient at retrieving data blocks than another. Architectures can vary as well. No matter how efficient data retrieval can be, and how much physical disk limitations can be mitigated, the overhead on the operating system that is retrieving the file is beyond the scope of SAN technology and is impacted by file fragmentation.

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

USING RESOURCE SENSITIVE TECHNOLOGIES TO THROTTLE ACTIVITY:

As covered in the section on virtualization, defragmenters may offer technologies to monitor I/O traffic, and throttle application related disk activity (i.e. defrag) until the I/O pipeline was free. Again, decisions to throttle activity are dependent on counters returned from the operating system on which the application is installed. In cases where multiple operating systems (multi-headed) connect to common shared disks, as in a SAN, I/O throttling techniques will not function appropriately. The case being that an application using I/O throttling may detect that a shared disk array is not busy, but a secondary server also using that same array may be processing a very disk intensive operation. In that event disk contention may occur.

Recommendation: File system drivers that prevent fragmentation at the source (the file system) with a file write offer an excellent solution in a SAN environment as they will not incur any added overhead in after the fact removal of fragmentation.

Proprietary technologies such as InvisiTasking® technology, which eliminates overhead on direct attached storage, will provide more effective resource-sensitivity for storage networks through granularity of its actions. However, with the possibility of overhead conflict in more I/O demanding SANs, it is recommended to undertake a proper evaluation of the environment to determine if defragmentation time frames are better suited to off-peak production hours.

CACHING:

More cache is always better. However, cache is far more expensive, and volatile, than magnetic media. In some business cases a 99%+ cache hit ratio may be the mandate – at all costs. That is addressed by spending and integrating a lot of cache through various tiers in the storage infrastructure. However, most storage infrastructures can, or must, do with a lower rate of cache hits. For data requests that miss cache and must be fulfilled from the disk, an optimized disk subsystem is important. Even for environments that dictate very high cache hit rates, the occasional miss is best mitigated by an optimized disk subsystem. And of course, before data is placed in the cache it must first be read from the disk; a process slowed by fragmentation.

THIN PROVISIONING:

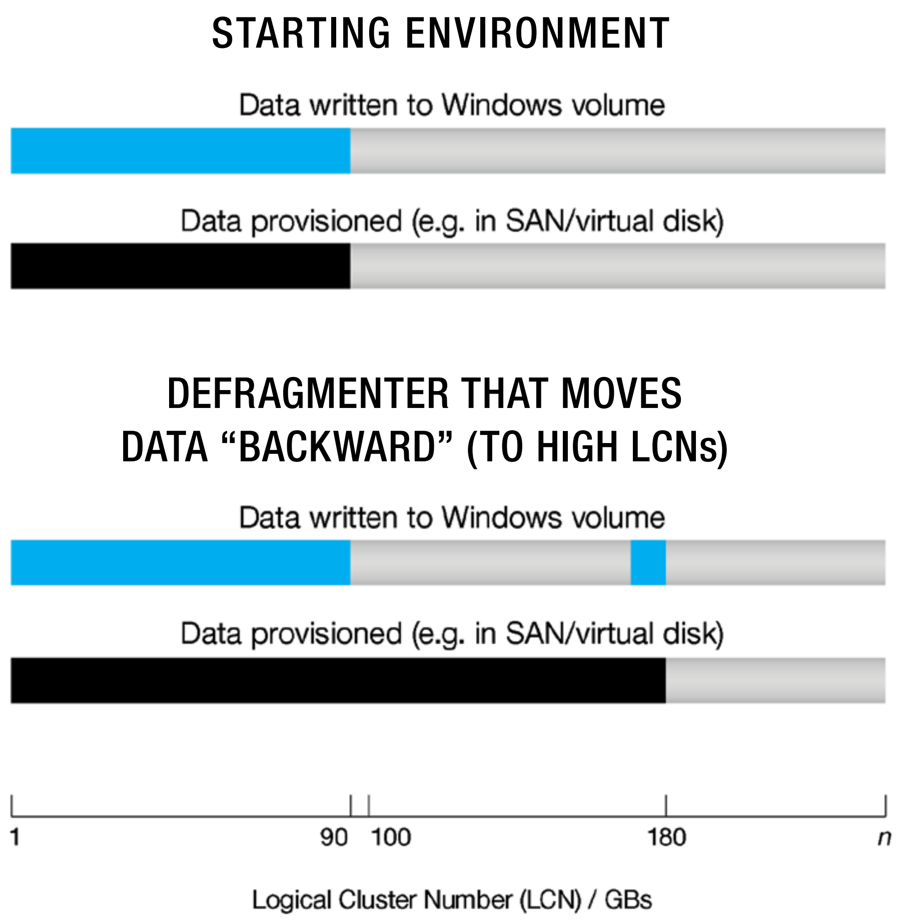
Earlier we described thin provisioning as it applied to a virtualization vendor's host file system, here we cover that technology's use on physical storage as it applies to a SAN.

Given that allocated disk space often goes unused, oversubscription combined with Thin Provisioning were technologies developed to make it appear more disk space existed virtually so SAN storage can allocate physical space dynamically to the volumes that need it. Space in the SAN file system is allocated on an as-needed and amount-needed basis in a storage pool shared by multiple servers. Provisioning accommodates the unpredictability and allocation of future storage growth needs and eliminates the need to assign storage to one volume/ computer when the system is built.

The term "Thin on Thin" *refers to the use of thin provisioning technology at both the virtual platform and physical storage level.*

Many SAN solution providers provision space based on the Windows volume high water mark. A high water mark, with respect to a volume in this definition, is the term that describes the last written block of data (the highest used LCN on the volume). That high water mark always increases and never decreases (on Windows), indicating less available space to the SAN. This creates a problem in properly provisioning space.

If a file is written (or moved via defragmentation) to a higher LCN, the SAN will need to provision space in the LUN to accommodate. That is true even if the file is only moved to a high LCN temporarily.


STARTING ENVIRONMENT
Data written to Windows volume
Data provisioned (e.g. in SAN/virtual disk)
DEFRAGMENTER THAT MOVES DATA "BACKWARD" (TO HIGH LCNs)
Data written to Windows volume
Data provisioned (e.g. in SAN/virtual disk)
1
90
100
180
n
Logical Cluster Number (LCN) / GBs

If you implement thin provisioning, it is recommended to check with both your SAN technologist and defragmentation software vendor, for proper configuration (e.g. disable file placement features).

DEFRAGMENTING CONTINUOUS DATA PROTECTION (CDP)/SNAPSHOT VOLUMES:
Data protection solutions seek to offer a copy of data either synchronously on every write using a copy-on-write (COW) design, or asynchronously at some point in time. Similar to file-based backup software, CDP offers full and differential approaches to data protection.

A copy-on-write design provides synchronous real time protection and recovery to any committed state.

Depending on the implementation, software or hardware, these copies may be written into a circular log/journal file, differencing area, or onto a separate volume/disk (e.g. split mirror/snapshot mirror/clone).

Generally a Full Copy/Split Mirror/Clone CDP implementation provides a one-time read-only data source that is moved off to a secondary volume. Only the original source volume then keeps all new incremental changes.

It is also possible that the original data volume inherits read-only status and snapshots, again using copy-on-write, duplicate the "changes" off the primary data source into a special change file or differencing area.

You may recognize the terminology used here from the earlier section on virtual disk types (i.e. Differencing disks/linked clones). That is because the concept is essentially the same.

A common issue with most COW solutions is that they are unable to distinguish between changes to data generated by applications, versus changes made to an application's location, as is done by a file system defragmenter. COW solutions that integrate with the Windows file system, for example a provider that integrates with Microsoft's Volume Shadow Copy Service, are more likely to understand fragmentation. A proprietary solution is unlikely to be able to recognize file movement from defragmentation.

Recommendation: Prevent Fragmentation at the file system level on snapshot/block-based CDP volumes. Background defragmentation should be evaluated for occasional use during a maintenance window or off peak time frame, though note it may increase a differencing area/file or traffic between a source and clone disk (significant usually only for remote clones).

For file system based CDP solutions (such as Microsoft's VSS and third party solutions that integrate with its framework) prevent fragmentation at the file system level, and use an advanced automatic background defragmenter that offers a specialized 'snapshot compatibility' mode.

***We've seen a huge disk** performance gain…using Diskeeper to defragment all databases/images/documents stored by our clients, which include over 1.5 TB of SQL data and file storage data. Data is stored on an IBM FastT500 SAN array on three 10-drive nodes (600GB, 600GB, 1.2TB respectively).*

*– Doug Robb,
VirMedice. LLC*

DEVICE SPECIFIC MODULES (DSMs):

SAN vendors provide a DSM for fault tolerance and bandwidth optimization, but they do not solve the issue of local disk fragmentation.

Typically an I/O request travels one path (as described earlier in this paper) from application to physical storage location. DSM allows a vendor to design alternative "paths" to the physical data storage in the event a component along the path breaks (e.g. a bad cable) or bottlenecks under heavy load. In either of these events the DSM can re-direct the I/O down another pathway. This is called a multipath I/O driver (MPIO). It is great for optimizing the performance of block level requests generated by the file system, but cannot minimize the overhead that file system occurs in generating multiple I/Os for one file object. It must accept the multiple unnecessary I/Os and optimize the retrieval of those multiple requests as best as possible.

MPIO (disk class drivers) reside below NTFS.sys in the I/O stack, and are reliant on IRPs initially generated from the file system and passed down the stack. "Fixing" local disk fragmentation is not the job of the SAN vendor, nor even their responsibility, as the issue occurs are a higher level (closer to the requesting application) than a SAN is, or should be, integrated with system I/O.

A QUICK OVERVIEW OF FILE REQUESTS THROUGH THE WINDOWS STORAGE STACK[9]:

- Application (e.g. SQL)
- I/O Manager
- NTFS.sys
- Volsnap.sys
- Disk.sys (e.g. SAN replacement of this driver such as MPIO)
- Hardware

While SANs implementing DSM can distribute I/Os more efficiently than Direct Attached Storage (DAS), fragmentation will still create application slows, as the operating system where the requesting application resides, still has to generate more I/Os than it should, and that data is still likely to be dispersed in a less than optimal set of locations on the storage array.

DATA OPTIMIZATION ACROSS SAN STORAGE TIERS/SECTORS OF A RAID ARRAY:

It is important to remember that defragmenters never move data to specific physical sectors on a storage device. Defragmentation only moves files from file system clusters to new file system clusters. How file system clusters map to physical sectors is addressed by non-Windows components, and is beyond the scope of a defragmenter.

It is possible that defragmenters with poorly architected algorithms may move the same allocated clusters (associated with files) over and over again. This can potentially trigger block oriented storage systems into believing the blocks associated with those files contain important data that should reside on faster storage tiers. Advanced defragmenters employ efficient algorithms that dramatically minimize future defrag effort and activity.

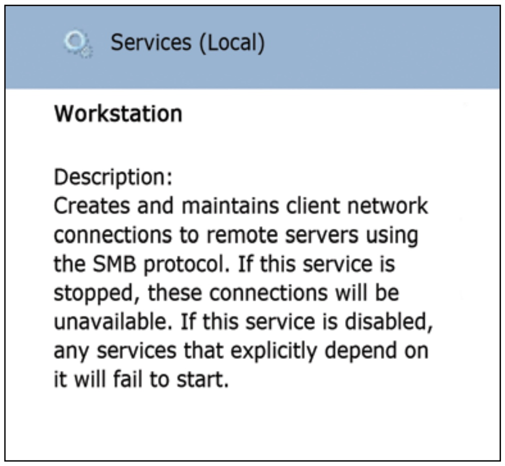
Also note that advanced technologies, such as I-FAAST™ (Intelligent File Access Acceleration Sequencing Technology) measure data transfer performance at the Windows volume level. I-FAAST will then move frequently used data into regions on the volume determined to have higher performance characteristic. Subsequently it moves stale data to slower access areas of a volume. Technologies such as I-FAAST will provide enhanced value for static storage/storage tiers where accurate measurements can be achieved at the volume level.

Recommendation: Prevent fragmentation at the file system level on SANs. Use only efficient defragmenters that minimize file movement, and apply intelligent file system level optimization routines.

9. FAST I/O, cached-data and other intermediate drivers are not included, but are irrelevant for this discussion.

IX. Defragmenting network attached storage (NAS)

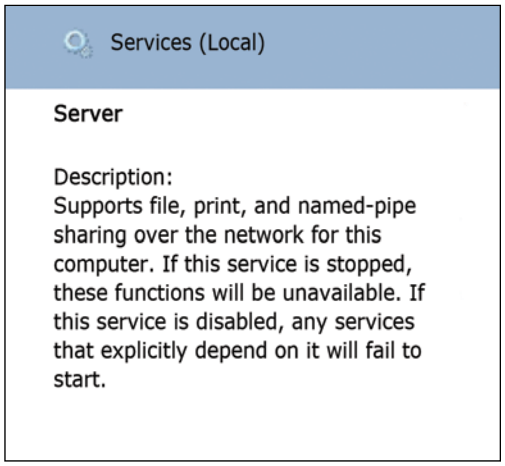
Windows supports “remote file systems” just as it supports “local file systems” (e.g. NTFS. sys). In Windows, the remote file system includes a client (requestor) and a server (provider) component. The Windows “Workstation” and “Server” services make this remote access function.



Services (Local)

Workstation

Description:
Creates and maintains client network connections to remote servers using the SMB protocol. If this service is stopped, these connections will be unavailable. If this service is disabled, any services that explicitly depend on it will fail to start.



Services (Local)

Server

Description:
Supports file, print, and named-pipe sharing over the network for this computer. If this service is stopped, these functions will be unavailable. If this service is disabled, any services that explicitly depend on it will fail to start.

This remote file system (also called distributed file system) is the mechanism used when, as an example, connecting over Ethernet to mapped shares on a file server. The protocol used to communicate between the requestor and the provider in a Windows network is known as Common Internet File System (CIFS), which is a Microsoft variant of IBM’s Server Message Block. CIFS has many other Windows network uses, and while improving, some of its limitations restrict usage in environments where remote data performance is vital. Hence, it is one of the driving forces behind the creation of other technologies such as SAN and Network Attached Storage (NAS).

A NAS box is essentially a disk array, with an operating system (Microsoft offers the Windows Server Appliance Kit and Windows Storage Server/Windows Unified Data Storage Server) that can be plugged into the network (/and SAN). It is a plug-and-play file server.

Fragmentation impedes NAS performance the same as it would a typical file server.

Using a network file system protocol, a Windows client (e.g. Windows 7) may map a drive to a NAS device and store date there; data that will become fragmented. Unlike DAS/ SAN attached storage, NAS is considered a remote drive, and is therefore not available for defragmentation by tools installed on/run from that remote Windows client.

Recommendation: Using a program installed on the Windows-based NAS device, prevent fragmentation at the file system level, and use automatic background defragmentation for full performance benefit. For NAS devices running non-Windows operating systems, check with the manufacturer for any disk optimization recommendations.

X. Conclusion

New storage technologies have absolutely improved storage performance, flexibility and manageability, but they do not solve issues generated at the file system level, such as file fragmentation.

As storage solutions and infrastructures have evolved, so have solutions to address fragmentation. “Prevention is better than the cure” is a popular and very IT applicable idiom. That philosophy, applied to preventing fragmentation with sophisticated new technology (i.e., IntelliWrite™ fragmentation prevention technology), is the ultimate solution to maximizing performance of modern storage solutions.

Physical storage devices and controllers will optimize the location of blocks across the underlying physical spindles according to their proprietary methods, but none are involved with how the file system requests I/Os. The need to optimize virtual disks, SAN, RAID, SATA, SCSI, NAS, HHD, SSD devices continues today, just as it has in the past.

***We use Diskeeper** EnterpriseServer on our main file server. This particular system has about 4-5 TB assigned to it from our SAN and needs constant defragmenting due to heavy use.*

– Martin Gourdeau, Network Admin, Electronic Arts

When bottlenecks occur in the disk subsystem, file fragmentation is a factor that should always be investigated as a contributing factor. To gauge the impact of fragmentation, use performance monitoring tools such as PerfMon, Iometer, or hIOmon. The appendix at the end of this paper provides examples and links to these tools.

For all the technical data provided on why fragmentation is relevant more than ever with new storage technologies, other real world factors make evaluating fragmentation a worthwhile cause.

Primarily, investigating fragmentation is inexpensive; requiring only some investigatory time. The evaluation software can be obtained for free, and the time required to test is far less than that involved in evaluating hardware solutions. Licensing and managing defragmentation software will be far less expense than hardware solutions and will likely provide a significant and immediate return on investment.

Appendix A

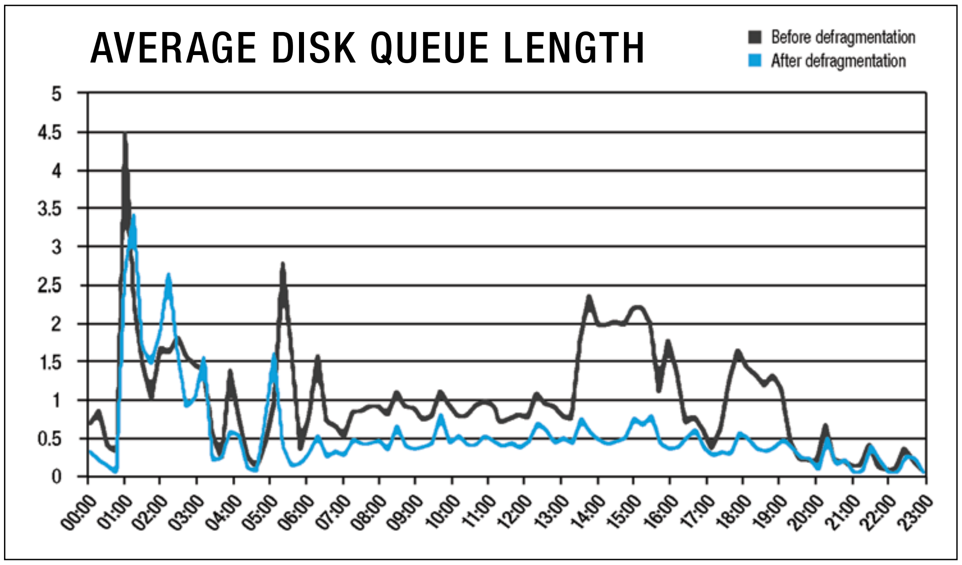
GAUGING THE IMPACT OF FRAGMENTATION:

PerfMon:
To determine fragmentation's impact on a disk subsystem (single disk or RAID), you can employ performance monitoring technologies. Window's includes a built-in tool called PerfMon that can collect and graph this data. Specifically you will want to direct it to the PhysicalDisk object. Performance monitoring for purposes of determining event-based changes (such as defragmentation) requires proper before (baseline) and after comparisons. This means that a similar extended period (e.g. one week) must be compared to determine improvement. No other changes, such as adding new hardware, can be introduced during the test periods. The periods measured must cover, to the degree possible, the same work load.

Here is a sample scenario:

1. On a Friday afternoon, install, but do not activate, an enterprise-class disk defragmenter, and run the tool's native analysis functions.
2. Save the defragmenter's analysis reports.
3. Start the PerfMon baseline on a Monday and let it run without any other hardware/system settings changes for one full week.
    - Avg. Disk Queue Length (should have no more than 2 per spindle)
    - Avg. Disk Read Queue Length (used to further define disk queues)
    - Avg. Disk Write Queue Length (used to further define disk queues)
    - Avg. Disk Transfer/sec (should be less than 50-55 per spindle)
    - Avg. Disk Read/sec (used to further define transfer rate)
    - Avg. Disk Write/sec (used to further define transfer rate)
    - Split IO/sec (should be less than 10% of Disk transfers/sec value)
    - % Disk Time (should ideally be less than 55%, over 70% is typically an issue)
    - % Idle Time (to check legitimacy of % Disk Time)
4. Using the disk defragmentation software, run another analysis and save the results.
5. Activate the defragmentation tool the following Monday morning and let it run for two weeks.
6. Using the disk defragmentation software, run the final "after" analysis and save the results.

7. Compare (see figure below) the first and last week periods and note changes (improvements) in the measured counters from week one (no defrag), to week three (defrag complete and still active). The disk defragmenter's reports will provide you data on the changes to file fragmentation as part of this before-and-after comparison.



8. If desired, stop defrag operations for the fourth week, and continue to monitor disk performance through week 5, to note reversal of achieved performance gains. Accompany this with another disk defragmentation analysis and compare the results of that analysis to data collected from week 3.

The handy Performance Monitor Wizard, available at Microsoft's website can ease the learning curve in setting up and using PerfMon.

No counter will independently determine the impact of fragmentation. If the disk is fragmented, many of these counters will show metrics higher than acceptable levels.

hIOmon™ by HyperI/OSM

HyperI/O, has developed a full "file I/O performance" evaluation kit, targeted specifically at determining the impact of fragmentation on production systems. Due to its robust feature set, this is a recommended product/method for experienced Server Administrators familiar with benchmarking and performance evaluations.

Iometer

An additional benchmarking tool is Iometer/Dynamo (distributed as binaries). It is an open source I/O subsystem measurement and characterization tool. Iometer/Dynamo can be used to benchmark test environments. The key to benchmarking fragmentation with this toolset is ensuring the test file is created in a fragmented state. This can be accomplished by fragmenting the free space on a test volume prior to use of this tool.

Appendix B

REFERENCES:

VMware (KB-1014) - Windows Virtual Machine Blue Screens When I Use SAN LUNs

Microsoft (Disk Fragmentation impacts RAID) - Windows Server™ 2003: Improving Manageability and Performance in Hardware RAID and Storage Area Networks

HP - Configuring Windows Server 2003 with HP Integrity Servers with SAS

Diskeeper Corporation - How NTFS reads a file

Diskeeper Corporation – Virtualization and Disk Performance

EMC – Backup Storage Solutions for CLARiiON and Symmetrix

SQL Knowledge - How to Monitor I/O Performance

Diskeeper Corporation – File Fragmentation: SAN/NAS/RAID

Iometer.org – Downloads and documentation

Hyper I/O - Fragmented File I/O Metrics

VMware – Layers of Virtual Storage (abstraction)

BIBLIOGRAPHY:

RealTime Publishers – The Shortcut Guide to Managing Disk Fragmentation (Mike Danseglio)

Diskeeper Corporation – FRAGMENTATION: the Condition, the Cause, the CURE (Craig Jensen)

Microsoft Press – Microsoft Windows Internals (Mark Russinovich, David Solomon, Alex Ionescu)

O'Reilly – Using SANs and NAS (W. Curtis Preston)