Best Practices for Defragmenting Thin Provisioned Storage

Why Defragment?

Before we cover considerations and recommended configurations in thin provisioned storage environments, it's important to revisit why defragmentation of Windows operating systems is so important in a virtualized machine and/or virtualized storage environment.

The problem is that fragmented data in a local disk file system, such as NTFS, causes the operating system to generate additional I/O requests. For each "logical" fragment in the file system, a separate I/O request packet (IRP) must be generated and passed on to underlying storage layers. So, for example, a file in 100 fragments would generate 100 separate smaller I/Os, rather than a single larger I/O.

We use it [Diskeeper® performance software] 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

This translates to an operating system processing a great deal more *unnecessary* I/O traffic, thereby increasing CPU and memory demand. In many cases, that excess I/O is passed on to a Storage Area Network (SAN) and/or virtualization platform, causing *additional* unnecessary overhead.

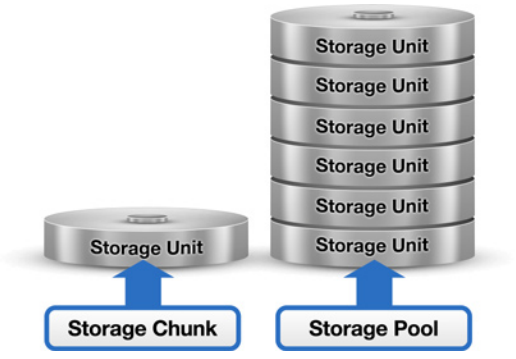
In some cases, data that is in a contiguous sequence of clusters in a local disk file system will be physically contiguous on the actual storage media, i.e., the disk drive/array. This is generally a valuable added benefit, but by no means required for defragmentation to greatly increase performance.

Some file systems (e.g., log-structured file system) used in SANs may intentionally fragment data at the "block" level. They may coalesce random writes from the OS into sequential writes within the storage. While this will minimize I/O activity in the SAN, it actually increases the likelihood that the data in those sequentially written stripes is physically fragmented, because the coalescing process is not based on re-ordering of blocks as they map to a common file – it simply dumps the data to the media. For these environments, you'll need to check with your storage vendor regarding proprietary defragmentation solutions for their SAN.

Regardless of spatial proximity, the benefit of a fragment-free local disk file system (NTFS) is that your OS and virtualization platforms aren't processing extra I/Os generated due to fragmentation, and will therefore be able to host more operating systems and process more data, faster.

Thin Provisioning 101

Thin provisioning allocates resources from an aggregate storage pool, which is essentially divided into assignable units commonly referred to as "chunks." Provisioning storage in "thin" environments is done in chunks that are pulled from that pool of available, and as yet unallocated, storage.



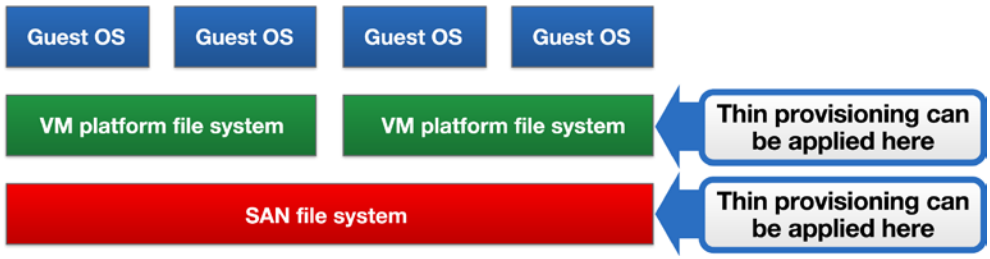
Storage pools are made up of many small storage units (chunks)

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

As data is added to a thin provisioned container, such as a Dynamic/Thin virtual disk or a LUN, that container increments, usually in a just-in-time basis, by a chunk or number of those chunks, depending on how many chunks are needed to house all the incoming writes. A chunk can be anywhere from a few kilobytes to gigabytes in size, and varies from one thin provisioning technology vendor to the next. In some cases it is a fixed size, in other solutions the chunk size is user-selectable.

How and when chunks are allocated also varies from vendor to vendor.

Many thin provisioning technologies provision *for every write*. They monitor blocks, and specifically changes to blocks. As new data is written, space is provisioned for it on a just-in-time basis, and it is stored.



Another method to provision space is 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 cluster/block of data (the highest used Logical Cluster Number, or LCN, on the volume). Everything beyond the high-water mark is assumed to be null.

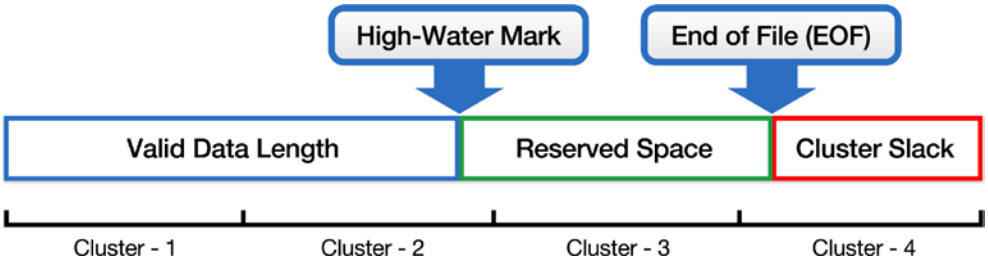
NTFS Write and Delete Design

Since Windows XP, all parts of a file stream, up to and including the allocation size (the file tail between valid data length and end of file) can be defragmented.

While not exactly "thin friendly," NTFS is undeserved of the reputation of being a problem for thin provisioned disks/LUNs. It has been mistakenly stated that NTFS carelessly writes to continuingly new and higher LCNs, until it has written to every cluster on the volume, before circling back around to clusters since freed up from file deletes. This is not correct.

When describing NTFS design as it relates to storage provisioning, we should first describe the various file sizes. There are three sizes for files in NTFS, and they use high-water marks, too.

The Valid Data Length (VDL) is the distance into the file that data has actually been written, as it resides in the cache. It is depicted as the blue bar in the diagram (next page). A VDL can include sparse runs interspersed between data. The highest written LCN that constitutes the VDL is the high-water mark for that file. There is no data, at least related to this file, that resides past the high-water mark. Without having to actually write zeroes, and just as with high-water mark storage volumes, reads attempted past the high-water mark return zeroes.



The next step up is the File Size. It is the VDL plus some extra pre-reserved space that has yet to be written to (uninitialized); also called the file tail. This is the full logical size of the file, shown as the combination of blue and green in the diagram, and is terminated by EndOfFile (EOF) flag.

Lastly, there is the Allocation Size, which indicates the full physical size of the file, and is comprised of the VDL and its following reserved space, up to the last cluster the file occupies any part of (may be some cluster slack). It is shown as the combination of blue, green, and red in the diagram.

To aid in writing new data, the NTFS file system driver maintains a list of the largest free spaces on the volume (i.e., the starting LCN and run length). When a file gets created, it gets created in the free space that most closely matches the size of data available to write; in other words, a "best fit." Additionally, a presumption is made that a newly created file will end up larger than the size that is currently available for the operating system to write, and extra free space, an "over-allocation," is reserved for the file so as to minimize fragmentation (see Microsoft Knowledge Base article ID 228198). The presumption is that the file will be 2, 4, 8 or 16 times larger than the currently known data size, depending on how much data is currently available for writing to the file in the operating system's file cache.

In addition to "best fit" attempts, Windows will also attempt to write new extents of a file in close cluster proximity to existing extents.

The file data is written to the volume, and the file is closed. Any over-allocation is then released, returning to the free space pool and to the NTFS file system driver, if it qualifies as one of the largest free spaces on the volume. For this part, and this is a critical point, NTFS is very thin-friendly, as when it reserves that over-allocation, it can do so without writing to the volume (i.e., writing out zeroes).

All said, this process does not eliminate fragmentation by any stretch, and hence the continuing necessity to defragment the file system.

One issue that *does* exist with NTFS, that presents universal challenges for thin provisioned storage, is the ability to recover space previously occupied by deleted files.

This is an issue because when files are deleted in NTFS, the file system simply updates its metadata to indicate that the space occupied can be re-used for new file writes. A deleted file is not actually removed/wiped from the volume. Therefore, abstracted storage layers residing underneath NTFS may not be informed about this newly available free space.

This creates a problem for thin provisioned storage which, if presented with limitations on re-use of space, could eventually exhaust all storage in the available pool.

A solution for this challenge, commonly known as Thin Reclamation, encompasses the awareness of space formerly occupied by deleted data and actions then undertaken to recover and re-provision that space. There are a variety of solutions available to aid with thin reclamation, such as zeroing deleted clusters to the SCSI UNMAP / SCSI WRITE_SAME commands, and will vary from vendor to vendor.

Defragmentation and Thin Provisioning

As covered earlier, defragmentation is vital to achieve and maintain peak performance. When Thin Provisioning is implemented on a shared virtualization host file system, it creates a high degree of probability of thin/dynamic virtual disk files themselves becoming fragmented, adding additional I/O overhead. In those storage systems, solving fragmentation becomes even more important.

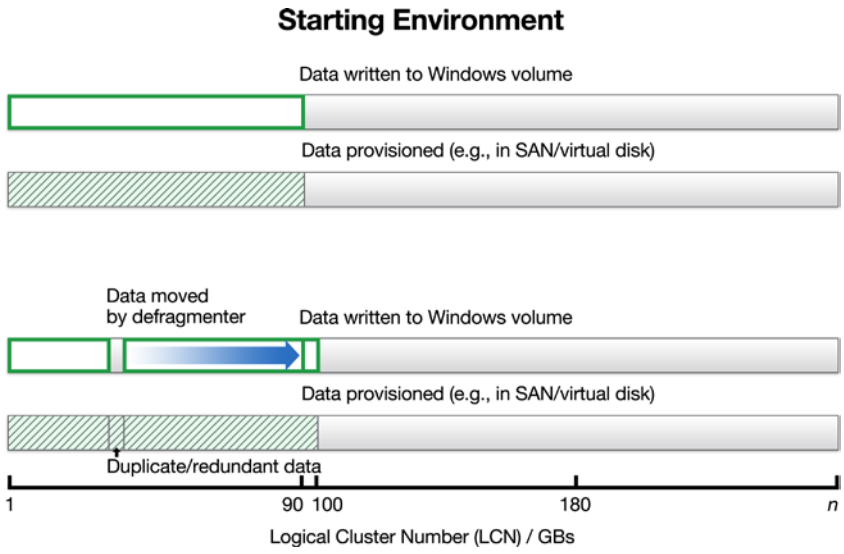
Many storage solutions track block-level changes (e.g., VMFS Change Block Tracking – CBT) to support advanced features like snapshots and live migration of data. So, if the file system blocks used for defragmentation are minimized and the same subset of blocks are re-used through the defrag process, the actual storage growth will likely be minimal.

However, for all the benefits of defragmentation, it is important to be aware of potential side effects. The side effects from defragmentation can vary from one thin technology implementation to the next, so it is important to know how the two technologies interact.

Using special IOCTLs (I/O controls) in Windows, defragmentation is essentially moving data to consolidate file fragments and to pool free space into large contiguous extents.

Where the provisioning technology allocates space on new writes, a defragmentation process (which is actually only moving data) will appear as new writes. Additionally, the former locations of moved data will not necessarily be known to be re-usable. Defrag will therefore generate additional storage capacity requirements for every piece of data moved.

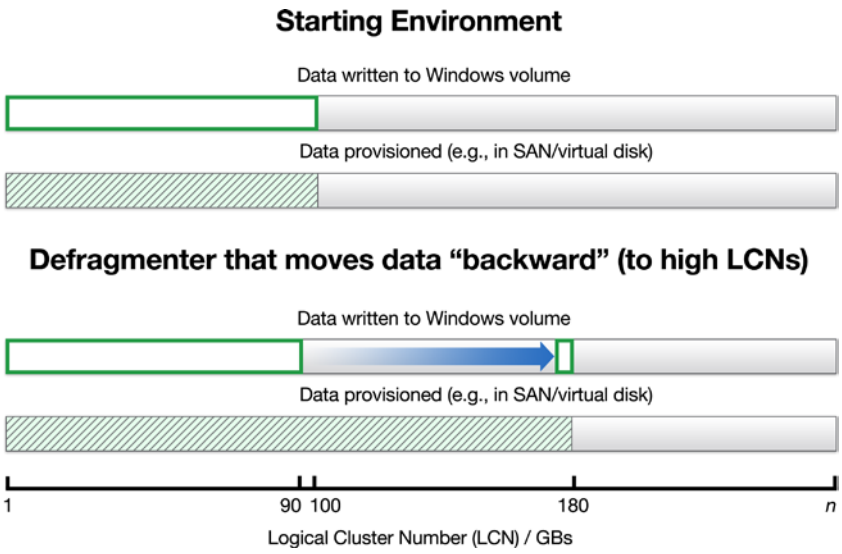
What can occur is that the new writes are redundantly provisioned, which results in unnecessarily consumed space.



Thin reclamation can effectively recover the wasted space, as could executing a data deduplication process (which would recognize and remove redundant data).

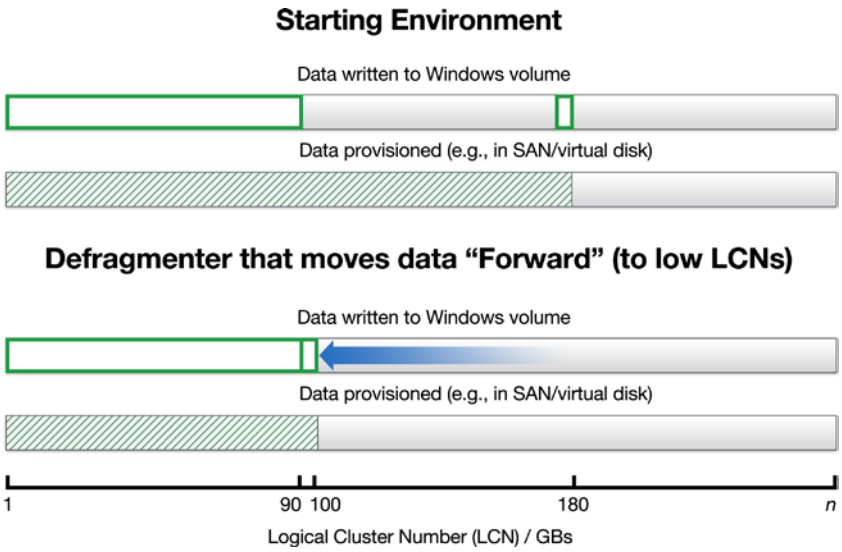
Where high-water mark provisioning is used, the water mark always increases and never decreases (on Windows), indicating less available space, creating a potential problem. If a file is written (or moved via defragmentation) to a higher cluster, the thin provisioning technology will need to provision space to accommodate. That is true even if the file is only moved to a high cluster temporarily.

V-locity® virtual platform disk optimizer, Diskeeper Pro Premier, Diskeeper Server and Diskeeper EnterpriseServer editions use specialized engines (Terabyte Volume Engine™ and Titan Defrag Engine™ technologies) designed to generally defragment/ move files to lower LCNs on Windows volumes.



On the opposite end of the spectrum, moving files "forward" can allow for space reclamation processes to *better* recover over-provisioned space (depicted below).

The process of compacting files to the front of a volume is something defragmenters can assist with.



Proactive Fragmentation Prevention

It is important to evaluate marketing claims from defragmentation vendors about "eliminating/preventing most fragmentation before it happens," as the technology behind the marketing claim can have differing consequences for thin provisioned storage.

Reactive solutions that rely on aggressive "free space consolidation" (packing files together) in order to rely on NTFS's native "best fit" attempts will cause thin provisioned growth.

Proactive technologies that do not require additional movement of any data in order to accomplish their objective do not cause increases in thin provisioned storage. They provide the benefit of a largely fragment-free OS file system without any negative consequences for thin provisioned storage.

Patent pending IntelliWrite™ technology, from Diskeeper Corporation, is such a proactive solution.

IntelliWrite is a superior design (to NTFS native over-allocations) for reserving space at the tail of a file's valid data. IntelliWrite is smarter in that it looks at the source of file writes/modifications and learns their behaviors over time. This heuristic process means that IntelliWrite knows better how much reservation space an open file needs to prevent fragmentation. It may be the file needs more than NTFS would natively offer, or it may pad less. The result of IntelliWrite's intelligent over-allocations is an unmatched degree of successful fragmentation prevention (up to 85% success rate and more).

Best Practices

- Use proactive fragmentation prevention technology, such as IntelliWrite from Diskeeper Corporation.
- Know, from your vendor of choice, how they thin provision and what solutions they have for space (thin) reclamation.
- In Thin-on-Thin provisioned environments, space reclamation at one layer (e.g., thin virtual disk) does not necessarily address other provisioned storage on subsequent layers (e.g., LUN).
- Defragment thin provisioned volumes when the corresponding storage growth can be addressed (e.g., a de-duplication/thin reclamation process).
- For high-water mark provisioning, use a defragmenter that moves files to lower LCNs (i.e., the "front").
- Use an OS/GOS defragmenter, or a defragmenter-mode that focuses on performance and not a "pretty" display.

- Apply SAN/VM vendor tools to eliminate fragmentation per their recommended practices for their proprietary clustered file systems.
- File sequencing/ordering technologies found in enterprise OS defragmenters can be quite valuable in many environments, especially performance-focused solutions on Direct Attached Storage. However, they can cause thin provisioned storage technologies to grow excessively due to their extra movement of data, so the general recommendation is to disable them or run them only when the effects (i.e., storage growth) can be addressed.

Diskeeper Corporation 7590 North Glenoaks Boulevard Burbank California 91504-1052 USA
Toll-Free 800-829-6468 Phone 818-771-1600 Fax 818-252-5514 www.diskeeper.com

Counting over 80% of the U.S. *Fortune* 1000 as volume license customers, and with over two decades of innovation in system performance and reliability (focused on storage performance), Diskeeper Corporation is a recognized expert in the storage performance industry.