
Thin Provisioning and NTFS Volumes

Niriva LLC – All rights reserved

There are no warranties express or implied that the information in this document is correct – consume the information or misinformation in this document at your own risk!

Abstract

Storage virtualization and iSCSI SANs are gaining momentum and/or recognition. All of this has lead to a number of vendors offering thin provisioning block storage wherein storage is overcommitted to applications. Applications such as NTFS file systems labor under the illusion that they have full access to a volume, whereas in reality, the block storage needed by the volume is allocated as needed.

The trade press and vendor literature expound the virtues of thin provisioning, but none of them dive into the specifics of thin provisioning in the Windows world. This document examines the implications of using thin provisioned NTFS volumes.

Contents

Introduction	4
NTFS Operational characteristics	5
NTFS and volume sector allocation	5
Defragmentation	6
Resizing an NTFS volume.....	8
Conclusion	9
References	Error! Bookmark not defined.

Introduction

Traditional storage provisioning takes a designated amount of block storage and allocates it for the sole use of a particular computer system volume. This results in at least temporarily unused storage. For example, an NTFS volume is formatted and only a small fraction of the volume is initially occupied. Until the contents of the volume grow to occupy a significant portion, a measurable amount of block storage is provisioned, allocated, but not really being used.

Thin provisioning is based upon the premise that a lot of systems and applications request more storage than they can use, or at least, than they can use right away. With thin provisioning, when an amount of block storage is requested, a smaller amount of block storage is provisioned from a pool and the requesting application is provided an illusion that the larger amount of storage it requested has been allocated. This can lead to a number of advantages:

- Storage is used more efficiently and only allocated as needed, on the fly
- Storage administration costs are reduced

Thin provisioning does have some drawbacks as well. Thin provisioning works well as long as there is adequate free block storage that can be allocated as needed.

- Somebody needs to periodically procure additional block storage as needed and add it to the block storage pool
- Inadequate free block storage can lead to data loss when data is being written to a file
- Worse yet, a volume may become corrupted if critical file system metadata cannot be written because a thin provisioned volume cannot be provisioned with additional block storage.

This document also defines and uses a term “high water mark provisioning”. Consider a volume that occupies say 1000 clusters of block storage. One scheme could be that the underlying block storage solution keeps tracks of the highest cluster unit the file system is writing to and provisions all clusters from zero to that highest cluster unit. This is deemed “high water mark provisioning.”. For example, if NTFS writes to cluster number 400, all clusters from zero to 400 are provisioned. The block storage solution could also implement a solution where only the cluster units that are actually written to are provisioned. The tradeoff between the two approaches is in terms of cost, speed, and savings of block storage.

NTFS Operational characteristics

This document refers to only the relevant characteristics of NTFS as they apply to thin provisioning. A complete description of NTFS is outside the scope of this document.

NTFS On disk layout

NTFS keeps a Master File Table (MFT) towards the beginning of a volume. The MFT occupies approximately 15% of the total volume space and the size of the MFT can be changed by changing a registry key and rebooting. NTFS also keeps an MFT mirror in the middle of the volume, though this mirror contains entries for only a few tens of files and is thus, not a complete copy of the MFT. NTFS also has lays out some so called System files on a volume and insists upon those files being immovable. These files are typically in the middle of the volume.

NTFS and volume cluster allocation

NTFS manages block storage in units often referred to as clusters. A cluster is a unit multiple of the underlying block storage sector size and is decided when an NTFS volume is first created. NTFS keeps track of what clusters within a volume are in use and what clusters within a volume are free.

When a new file is being created and written to, or an existing file is being extended, NTFS needs to allocate and use free clusters. The NTFS algorithm to decide what free cluster to use is fairly complex. For the purposes of this discussion, it is sufficient to understand that

- The algorithm does NOT necessarily pick the volume cluster that is closest to the beginning of the volume
- The algorithm favors, at least to some extent, skipping clusters that belong to files in the "Recycle Bin". These are files that have been deleted, but *may* be restored.

This leads to some interesting conclusions:

- Emptying the Recycle Bin will increase the chances that an already provisioned cluster will be reused, rather than extending the thin provisioned volume. Defragmenting the volume also increases the chances that an unused, but already provisioned volume cluster will be used.
- No matter what you do, over time, NTFS will slowly evolve the volume in a manner where it has unused clusters in the middle and has some clusters in use at the very end of the volume. This implies that the volume which started out as a thin provisioned volume has now become fully provisioned, assuming the thin provisioning uses high watermark provisioning.

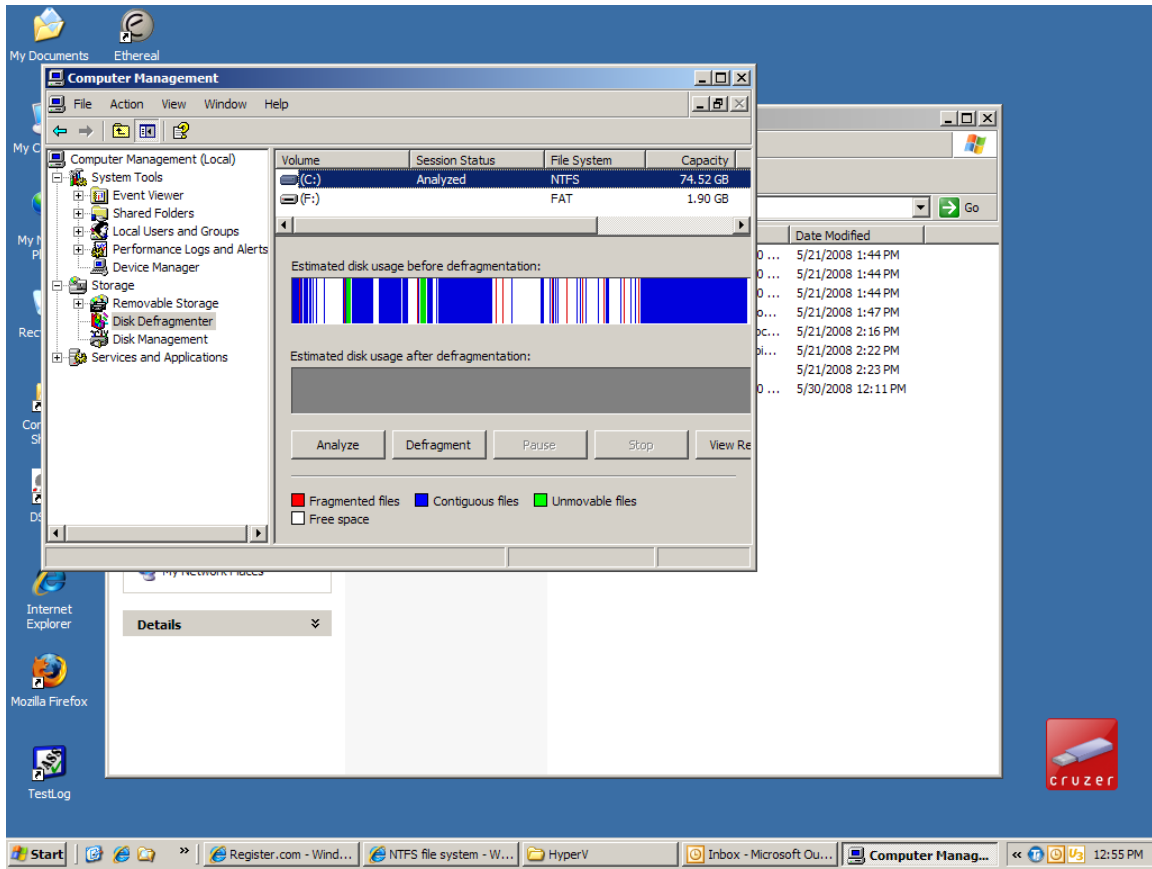


Figure 1: An NTFS desktop volume after approximately 12 months of operation

Figure 1 depicts an NTFS desktop volume after approximately a years worth of use. Granted thin provisioning is applied only to server volumes and not desktop volumes, but the key assumption here is that NTFS uses the same cluster allocation algorithm on desktops and servers. The desktop volume example is used here as a matter of convenience. Figure 1 clearly shows some free space in the middle of the volume and a substantial amount of files being stored at the end of the volume.

Defragmentation

When a file system volume is created, all of the free space is contiguous. As files are allocated, edited, copied and deleted, free space tends to get scattered in between existing files. Also, in the beginning, files can be stored in contiguous sectors. Later on, a single file may tend to be stored in non contiguous sectors.

Defragmentation is the act of re-arranging the allocated sectors so that

- Most if not all files are allocated in contiguous sectors
- The free sectors are all in a single or a few contiguous runs

Defragmenting an NTFS volume with the proper tools and procedures helps ensure that the free space within a volume is mostly contiguous. Interestingly, where the free space will be

accumulated depends upon a number of factors including the nature of the volume usage, the frequency with which defragmentation is done, and the nature of the defragmentation product used.

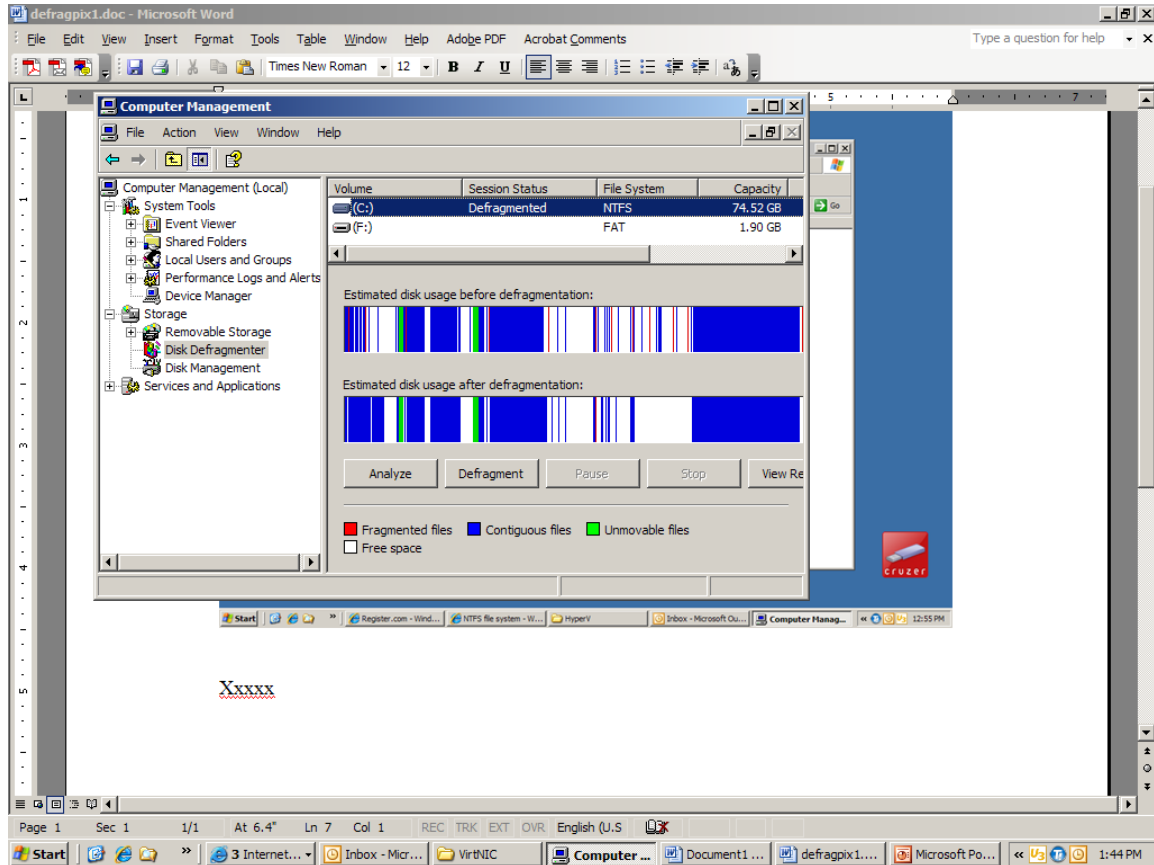


Figure 2 – NTFS volume before and after defragmentation

Figure 2 shows that while this particular NTFS volume has indeed been defragmented, it continues to occupy space at the end of the volume.

Some defragmentation products allow the administrator to specify a file type and all files of that type are placed at the end of the volume. An administrator could use a multi step process to

- First delete all unused files – this would leave the volume fragmented
- Create an unused file type or a series of such files that occupy some space. For example, one could create a bunch of .pst files on a server volume if one is sure there are no such existing files. Or image files with a .png extension to provide another example.
- Defragment and ensure that the newly created file type is specified to be stored last in the volume
- Delete the unused file type

To summarize, defragmenting an NTFS volume will provision some new clusters and from an NTFS point of view, will free up some currently provisioned clusters. The question is whether any

of the currently available thin provisioned systems recognize when NTFS has freed up a cluster and ever bother to de-provision it.

The important consideration is that if high water mark thin provisioning is being used, the volume is no longer thinly provisioned, as data blocks are allocated at the end of the volume during the defragmentation process and also during the regular file cluster allocation process by NTFS

Resizing an NTFS volume

An NTFS volume that has files occupying the last cluster in the volume is no longer a thinly provisioned volume.

After defragmentation in an appropriate manner and/or using the appropriate defragmentation tool, an NTFS volume may have unused, but provisioned block storage at the end of the volume.

Windows Vista and Windows Server 2008 have built in tools to shrink and grow a volume. So once a volume has been defragmented and has free space at the end, the volume may be shrunk. Some vendors offer similar tools for earlier versions of Windows as well.

However, whether the thin provisioning solution recognizes the shrunk volume and de-provisions the block storage remains a characteristic of the thin provisioning solution and not Windows itself.

Conversely, when a volume is grown, the thin provisioning solution should not provision additional data blocks. Most, if not all, the thin provisioning solutions do so.

NTFS quotas and thin provisioning

One problem with thin provisioning is that data may be lost if block storage provisioning fails while data is being written. Worse yet, the consequences can be more severe if the failure to allocate a block storage cluster occurs while NTFS is attempting to write some metadata.

One non trivial solution an administrator may explore is to consider tuning the NTFS quotas in such a way that the NTFS quota limits are encountered before the block storage clusters are exhausted. This is non trivial because it needs continuous refinement e.g. every time a user account is added or deleted, the administrator needs to take some action.

Conclusion

Thin provisioning has its advantages and disadvantages. Using thin provisioning on NTFS volumes requires a disciplined approach that involves deploying proper tools and techniques. There is an opportunity for vendors to provide a better integration between thin provisioning and NTFS volumes.