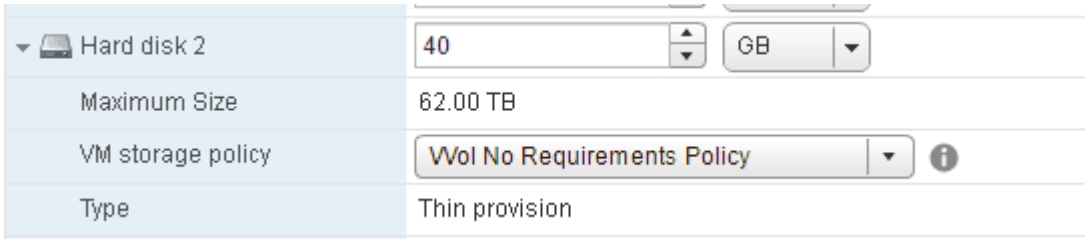
VMware Capacity Reporting Part V: VVols and UNMAP



Storage capacity reporting seems like a pretty straight forward topic. How much storage am I using? But when you introduce the concept of multiple levels of thin provisioning AND data reduction into it, all usage is not equal (does it compress well? does it dedupe well? is it zeroes?). In Part V of our series, VVols and UNMAP, we will discuss this and more.

This multi-part series will break it down in the following sections:

1. VMFS and thin virtual disks
2. VMFS and thick virtual disks
3. Thoughts on VMFS Capacity Reporting
4. VVols and capacity reporting
5. VVols and UNMAP

Let's talk about the ins and outs of these in detail, then of course finish it up with why VVols makes this so much better.

NOTE: Examples in this are given from a FlashArray perspective. So mileage may vary depending on the type of array you have. The VMFS and above layer though are the same for all. This is the benefit of VMFS–it abstracts the physical layer. This is also the downside, as I will describe in these posts.

The last topic in this series I wanted to discuss is Space Reclamation and Virtual Volumes.

With any file system on block storage, when you delete a file, the underlying storage doesn't know that the file has been deleted and that the corresponding physical space can be freed up. This is what UNMAP does. Until UNMAP is issued to the device underlying the file system and the underlying storage platform (commonly an array) frees it up.

If UNMAP is not issued, the capacity is still reported as used on the array and the physical space is still reserved. This is not deallocated until it is overwritten, or the physical device is actually deleted from the array. Or of course UNMAP is eventually issued.

OSes and hypervisors have implemented support (in various ways) of UNMAP to make sure they are "thin provisioning friendly". In other words: if the file system is not using certain blocks any more, the file system tells the array (via UNMAP) that they are now free.

The array deallocates which allows someone else to use those blocks.

Windows 2012 R2 and later has implemented this in two ways:

- Automatically. By default, when a file is deleted (and removed from the recycling bin) on NTFS, Windows issues UNMAP to reclaim the space automatically. This can be disabled by setting DisableDeleteNotify via fsutils to 1 (default is 0, meaning UNMAP is enabled)
- On demand. The Disk Optimizer tool, can be run manually, or on a schedule to issue UNMAP. This can be run via the GUI application, defrag.exe or optimize-volume with PowerShell.

Linux also supports it. Pretty much all of the modern file systems. There are also a few options:

- Automatically. Unlike Windows, this is not a default behavior. This is a mount option. If you mount ext4 or XFS or whatever, with the "discard" option, the file system will issue UNMAP as soon as a a file is deleted
- Manually. You can run fstrim to a file system to issue UNMAP as needed. This will issue it to any current "dead blocks" (places where files have been deleted but UNMAP has not yet been issued since).
- On a schedule. For most distributions, you can use fstrim.timer. Ubuntu though has a default script that does it. Google your way to your specific version/distro for more info specific to you.

A hypervisor like ESXi with VMFS is an interesting case. This is because there are two levels of file systems here. There is the OS in the VM managing a file system (NTFS, XFS, whatever) that sits on a virtual disk. Then there is ESXi managing a file system (VMFS) on a storage device that hosts those virtual disks.

So if a person deletes a file from a file system on a virtual disk, that UNMAP needs to be reflected by VMFS in shrinking the virtual disk then also transmitted to the array.

Support for this was introduced in vSphere 6.0. The process is:

- File is deleted in the guest.
- The guest issues UNMAP.
- The virtual disk shrinks.
- ESXi issues UNMAP at some point to the array.

The first part of this (allowing the guest to issue UNMAP that shrinks the VMDK) had and still has a fair amount of requirements. The virtual disk must be thin. If it is vSphere 6.0, it only works with Windows. Linux requires 6.5. In 6.0 CBT cannot be enabled. Windows doesn't align UNMAPs well, so a lot of them are not translated down to the array. Etc. Etc. Etc.

Then there is the VMFS part of it. If someone moves a VM or a virtual disk, or deletes a VM or a virtual disk, a lot of dead space is now present on the array. These objects can be large.

VMFS then has its own UNMAP. This is for the following situations:

- A deleted or moved file
    - VMDK
    - snapshot
    - swap
    - other VM files
    - ISO, etc.
- Space from a file that has shrunk. Like a thin virtual disk shrunk by in-guest UNMAP.

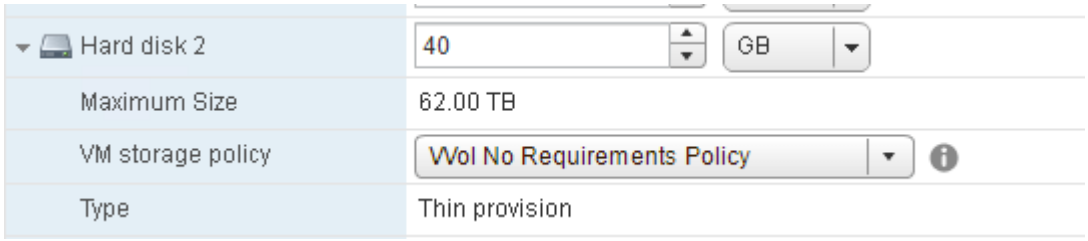
This VMFS UNMAP process issues unmap to where that file used to be, or where that part of that file used to be. This process has been through some iterations.

1. In ESXi 5.0, it was automatic and synchronous. You deleted a VM or a virtual disk, UNMAP would be issued to the array. This caused problems.
2. In 5.0 U1, it was made into a CLI command in vmkfstools.
3. There were issues with that, so manual VMFS UNMAP was improved and also moved into esxcli in 5.5 and 6.0.
4. Also in 6.0, EnableBlockDelete was offered to allow for VMFS to issue UNMAPs automatically in response to a VMDK being shrunk from in-guest UNMAPs.
5. In 6.5, VMFS UNMAP was made automatic again with VMFS-6, but asynchronous in fashion, so reclamation didn't happen immediately. But at least when you delete a VM or virtual disk, that space will eventually be reclaimed without the need for user intervention.

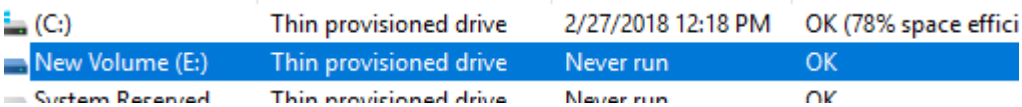
All of this complexity and translation goes away with VVols

VVols and UNMAP

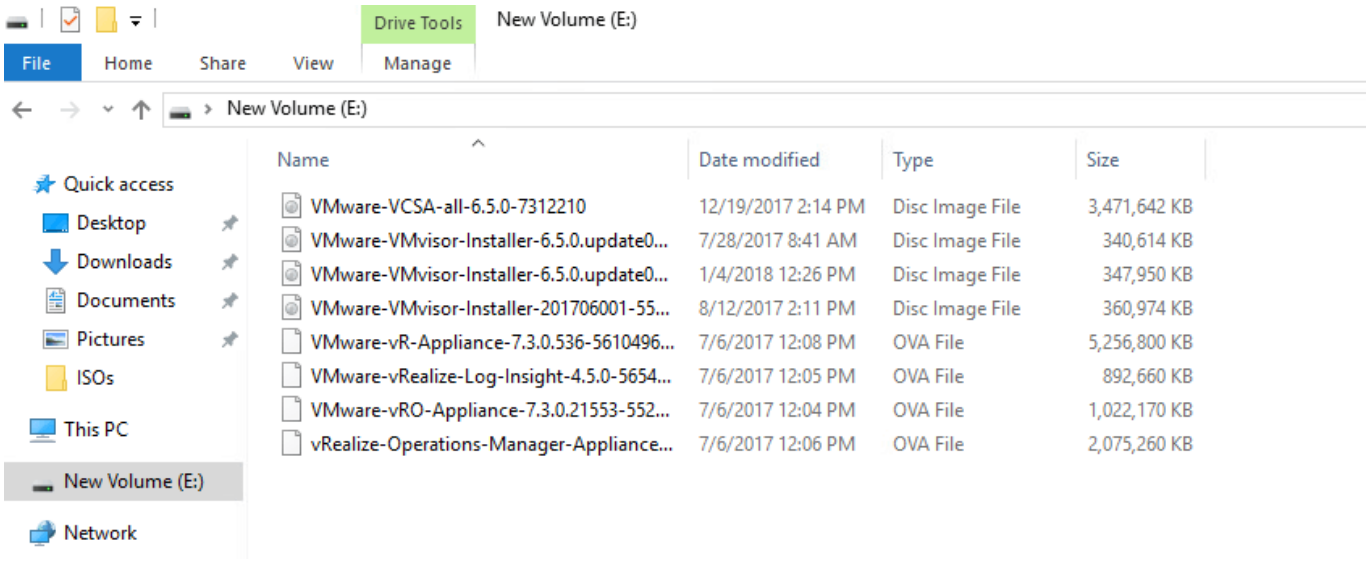
VVols are direct volumes. A virtual disk in a VVol world is a volume on the array. So when someone deletes a file in a VM, the guest file system is actually issuing UNMAP directly to the array volume. So no VMDK shrinking etc is needed.



The VVol reports as thin (meaning UNMAP is supported)

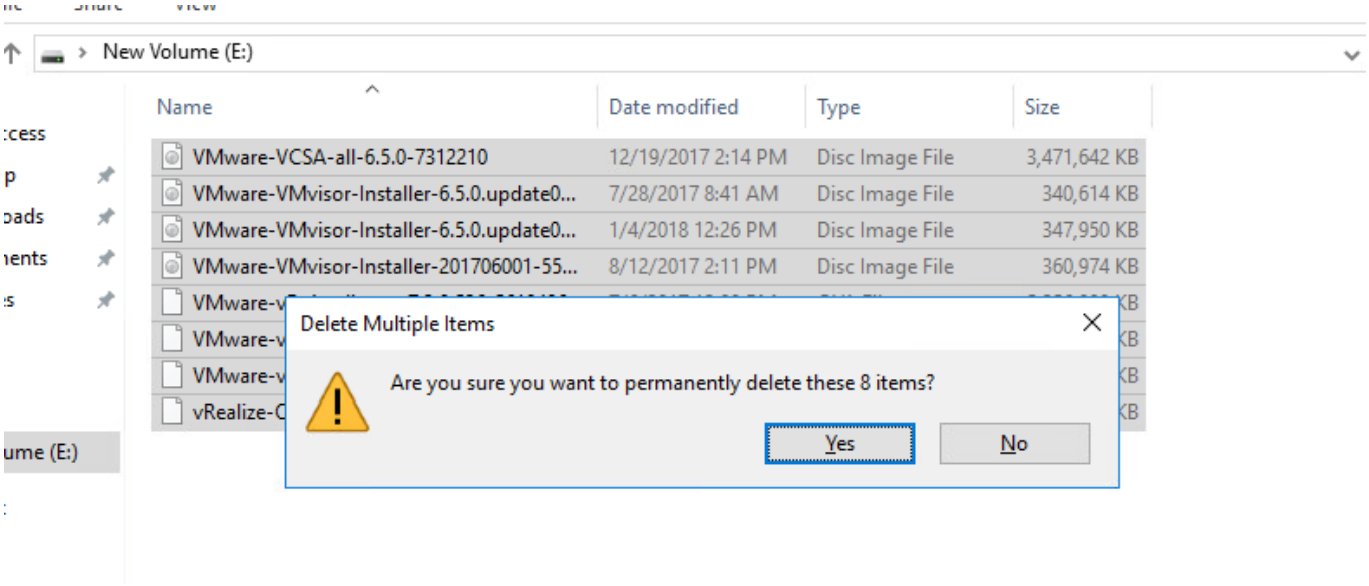


If I then put a bunch of files on it:



The space is consumed on the volume for that virtual disk:

Then I delete the files:

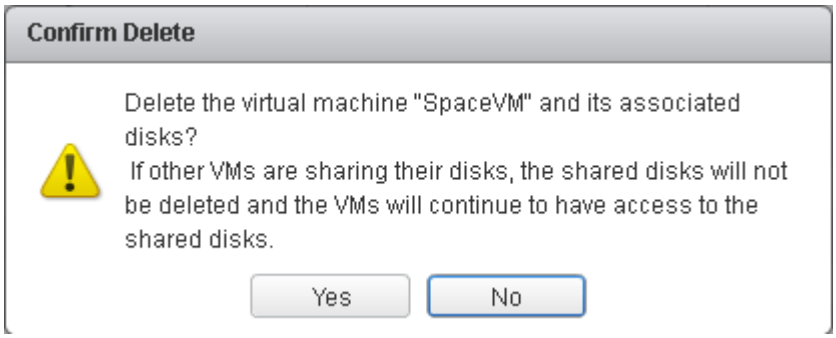


NTFS (in this case will automatically issue UNMAP. Directly to the array. So there is no concern about NTFS allocation unit size, or what version of VMFS is it on. All of the caveats that exist with the VMDK and VMFS UNMAP integration are gone. It has the same requirements of a physical server: does your array volume support UNMAP? Then it works.

The other part of this is of course deleting VMs, virtual disks and snapshots. When these objects are deleted or moved, they are not being deleted from a file system, because a VVol datastore is not a file system (as I went over here).

When you delete a snapshot, the snapshot on the array is deleted, so the space is reclaimed. When you delete a virtual disk, the volume is deleted on the array. When you delete a VM, all of the volumes on the array for that VM are deleted.

So from a VMware perspective UNMAP is a non-issue. There is nothing needed in the ESXi layer to really enable end to end space efficiency. Just enforce practices in the guest VMs, just like with physical servers.



On the FlashArray, those volumes (and the volume group) will go into our destroyed items folder for 24 hours and then be permanently deleted, therefore reclaiming the space.