vSphere 6.5 Storage

January 08, 2018

Table of Contents

1. Core Storage Whitepaper

This whitepaper describes in detail the various features of the vSphere 6.5 Core Storage.

1.1 Storage Limit Improvements

Storage Limit Improvements

Paths

ESXi hosts running version 6.5 can now support up to 2,000 paths in total. This is an increase from the 1024 paths that were supported in previous versions of vSphere.

Devices

ESXi hosts running version 6.5 can now support up to 512 devices. This is a two-fold increase from previous versions of ESXi where the number of devices supported per host was limited to 256.

Datastore connectivity

This is a future-proofing feature. With improvements to the heartbeat metadata area on VMFS-6, there is now support for up to 1,000 hosts connecting to the same datastore. However, there is still a LUN connectivity limit that needs to be considered, so if host-to-LUN connectivity limits increases in future release of vSphere, VMFS will also be able to support increases host connectivity.

512e Advanced Format Device Support

The storage industry is hitting capacity limits with 512N (native) sector size used currently in rotating storage media. To address this issue, the storage industry has proposed new Advanced Format (AF) drives which use a 4K native sector size. These AF drives allows disk drive vendors to build high capacity drives which also provide better performance, efficient space utilization and improved reliability and error correction capability.

Given that legacy applications and operating systems may not be able to support 4KN drives, the storage industry has proposed an intermediate step to support legacy applications by providing 4K sector size drives in 512 emulation (512e) mode. These drives will have a physical sector size of 4K but the logical sector size of 512 bytes and are called 512e drives. These drives are now supported on vSphere 6.5 for VMFS and RDM (Raw Device Mappings).

1.2 Pluggable Storage Architecture (PSA) Improvements

Descriptor Format Sense

ESXi version 6.5 introduces Descriptor Format Sense support throughout its Pluggable Storage Architecture (PSA). This enables better support for larger disk drives and third party plugins, which must use Descriptor Format Sense to report media errors and other check conditions referring to Logical Block Addresses (LBA).

Support for Descriptor Format Sense was also a necessary step for pass-through Raw Device Mapping (RDM) device support. This enables a more correct pass-through mechanism.

Third-party plugins in the PSA may also wish to utilize this information for other purposes.

1.3 VMFS-6

VMFS-6

VMFS-6 is the new filesystem version that is included with the vSphere 6.5 release. In this section, some of the new features and characteristics of this new file system are explored.

Sector Readiness

As part of future-proofing, all metadata on VMFS-6 is aligned on 4KB blocks. This means that VMFS-6 is ready to fully support the new, larger capacity, 4KN sector disk drives when vSphere supports them.

File System Resource Management

File Block Format

VMFS-6 introduces two new block sizes, referred to as small file block (SFB) and large file block (LFB). While the SFB size can range from 64KB to 1MB for future use-cases, VMFS-6 in vSphere 6.5 is utilizing an SFB size of 1MB only. The LFB size is set to 512MB.

Thin disks created on VMFS-6 are initially backed with SFBs. Thick disks created on VMFS-6 are allocated LFBs as much as possible. For the portion of the thick disk which does not fit into an LFB, SFBs are allocated.

These enhancements should result in much faster file creation times. This is especially true with swap file creation so long as the swap file can be created with all LFBs. Swap files are always thickly provisioned.

System Resource Files

When formatting a VMFS-6 volume, the number of system resources (e.g. pointer blocks, sub-blocks, file descriptors) is set to (RESOURCE_PER_TB_FOR_VMFS5 * VOL_SIZE_IN_TB). If this value turns out to be less than 16384, 16384 resources are automatically created. The reasoning behind this is to initially create enough resources to avoid frequent resource file extensions. If this value turns out to be greater than 16384, we cap the initial number of resources to 16384. The reason is to cap the initial resources is to save on the disk space used by these resources. But of course for larger volumes, we will extend the system resources are needed.

System resource files (.fdc.sf, .pbc.sf, .sbc.sf, .jbc.sf) are extended dynamically for VMFS-6. This means that they may show a much smaller size than observed with previous versions of VMFS, but will grow over time.

If the filesystem exhausts any sub-blocks / pointer blocks / file descriptors, the respective system resource file is extended to create additional resources. That way, VMFS-6 can support millions of files / pointer blocks / sub blocks (as long as volume has free space).

Journaling

Previous version of VMFS used journal resource blocks allocated as regular file blocks (1MB in size). In VMFS-6, journal blocks are tracked in a separate system resource file called *.jbc.sf.* Each time a VMFS-6 volume is opened, the relevant host allocates a journal block on that volume for itself. This journal block comes from *.jbc.sf* system resource file. The journal block is released when the host closes the volume.

This new mechanism was introduced to address journal issues on previous versions of VMFS, due to the use of regular files blocks as journal blocks and vice-versa. Tracking journal blocks separately in a new resource file reduces the risk of issues arising due to journal blocks being interpreted as regular file blocks. Note that the journal resource file can also be dynamically extended. Initially, 128 journal

blocks (256 MB in size in total) are allocated, and the journal resource file is extended when number of free journal resource blocks drops below a threshold of 64.

VM-based Block Allocation Affinity

Host contention issues were identified in the past when a VM/VMDK was created on one host, and then vMotion was used to migrate it to another host. If additional blocks were allocated to the VM/VMDK by the new host, the different hosts could contend for resource locks on the same resource pool of allocated blocks for the same VM. This change introduces block allocation affinity for VMs, which will decrease resource lock contention. With this VM-based block allocation affinity, sharing the resource clusters used by a VM will be reduced as much as possible.

ATS Miscompare Handling

This is quite a well-known issue. The heartbeat region of VMFS is used for on-disk locking, and every host that uses the VMFS volume has its own heartbeat region. This region is updated by the host on every heartbeat. The region that is updated is the time stamp, which tells others that this host is alive. When the host is down, this region is used to communicate lock state to other hosts.

In vSphere 5.5 U2, we started using ATS for maintaining the heartbeat. ATS is the Atomic Test and Set primitive which is one of the VAAI primitives. Prior to this release, we only used ATS when the heartbeat state changed. For example, we would use ATS in the following cases:

- Acquire a heartbeat
- Clear a heartbeat
- Replay a heartbeat
- Reclaim a heartbeat

We did not use ATS for maintaining the 'liveness' of a heartbeat. This is the change that was introduced in vSphere 5.5 U2 and which appears to have led to issues for certain storage arrays.

When vSphere receives an ATS Miscompare, it aborts all the outstanding IOs. This led to additional stress and load being placed on the storage arrays, and in some cases, led to the controllers crashing on the array. In vSphere 6.5, there are new heuristics added so that when we get a miscompare event, we retry the read and verify that there is a miscompare. Note that if, after re-read, we determine the miscompare is real, then we go into reclaim as we have been doing. However, if we determine that the on-disk HB data has *not* changed, then we determine this is a false miscompare. In the event of a false miscompare:

- VMFS will not immediately abort IOs.
- VMFS will re-attempt ATS HB after a short interval (usually less than 100ms).

A miscompare could also occur under the following circumstances: HB ATS fails with timeout or abort (reclaim). This meant that it could not be determined whether or not the IO made it to disk. The previous way of handling this was to retry the HB ATS again. But the same "test" buffer as before was used. If the IO had made it to disk on the earlier ATS, a miscompare occurred. This also used to cause reclaim (and aborting IOs). This case is now handled in vSphere 6.5 - during the re-read of the HB, an earlier HB IO is checked to see if it has made it to disk. If so, the test buffer is now updated and retry the HB IO instead of going into reclaim. Note that this enhancement works with both VMFS-5 and VMFS-6.

Parallelism/Concurrency Improvements

This next feature introduces lock contention improvements and improved resignaturing and scanning. Some of the lock mechanisms on VMFS were largely responsible for some of the biggest delays in parallel device scanning and filesystem probing on ESXi. Since Sphere 6.5 has higher limits on number of devices and paths, a big factor in enabling this support was to redesign device discovery and filesystem probing to be highly parallel.

These improvements are significant for Site Recover Manager, especially with a fail-over event, as the changes here lead to improved resignature and rescan/device discovery.

There are also benefits to Thin provisioning operations. Previous versions of VMFS only allowed one transaction at a time per host on a given filesystem. VMFS-6 supports multiple concurrent transactions at a time per host on a given filesystem. This results in improved IOPS for multi-threaded workloads on thin files.

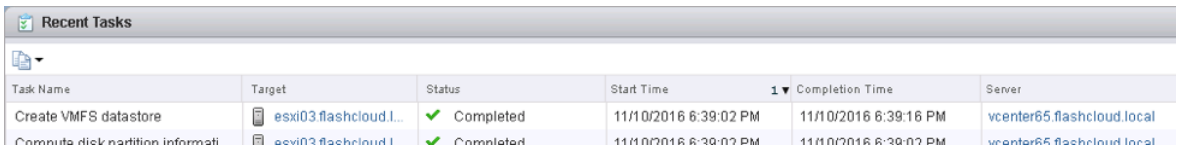
New VMFS-6 Features in action

In this section of the paper, we will attempt to demonstrate some of the benefits of the new features previously discussed.

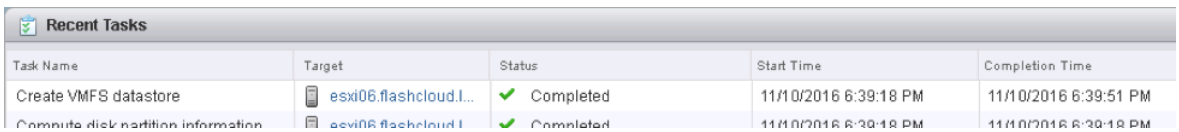
VMFS Creation

Using these new enhancements, the initialization and creation of new VMFS datastore has been significantly improved in ESXi 6.5. For a 32 TB volume, VMFS creation time was halved. In the example shown below, the creations of a 32TB VMFS-6 volume on ESXi 6.5 only takes half the time of creating a 32TB VMFS-5 volume on ESXi 6.0U2.

ESXi 6.5 and VMFS-6: Total time 14 seconds



ESXi 6.0U2 and VMFS-5: Total time of 33 seconds



VMFS Resignaturing improvements

In this section, enhancements and improvements to VMFS resignaturing are reviewed. Using vRealize Orchestrator, eight VMFS-5 datastores are created and presented to an ESXi 6.0 U2 host. All eight datastores are then copied using array-based snapshot on a Pure Storage array, and those volumes are then presented back to the same ESXi host. A simultaneous resignature operation was then run on all eight volumes from the same host. The exact same operation was then repeated using VMFS-6 on ESXi 6.5 a host. Improvement times that were observed were quite dramatic—19 minutes 43 seconds

on VMFS-5 compared to just 15 seconds on VMFS-6. The following are some screenshots of the tasks view in vSphere, showing the time to do this operation:

ESXi 6.5 and VMFS 6: Total time of 15 seconds

Recent Tasks

Task Name	Target	Status	Start Time	Completion Time	Server
Resignature unresolved VMFS volume	esxi04.flashcloud.l...	Completed	11/6/2016 2:49:57 PM	11/6/2016 2:50:11 PM	vcenter65.flashcloud.local
Resignature unresolved VMFS volume	esxi04.flashcloud.l...	Completed	11/6/2016 2:49:56 PM	11/6/2016 2:50:11 PM	vcenter65.flashcloud.local
Resignature unresolved VMFS volume	esxi04.flashcloud.l...	Completed	11/6/2016 2:49:56 PM	11/6/2016 2:50:11 PM	vcenter65.flashcloud.local
Resignature unresolved VMFS volume	esxi04.flashcloud.l...	Completed	11/6/2016 2:49:56 PM	11/6/2016 2:50:11 PM	vcenter65.flashcloud.local
Resignature unresolved VMFS volume	esxi04.flashcloud.l...	Completed	11/6/2016 2:49:56 PM	11/6/2016 2:50:11 PM	vcenter65.flashcloud.local
Resignature unresolved VMFS volume	esxi04.flashcloud.l...	Completed	11/6/2016 2:49:56 PM	11/6/2016 2:50:11 PM	vcenter65.flashcloud.local
Resignature unresolved VMFS volume	esxi04.flashcloud.l...	Completed	11/6/2016 2:49:56 PM	11/6/2016 2:50:11 PM	vcenter65.flashcloud.local
Resignature unresolved VMFS volume	esxi04.flashcloud.l...	Completed	11/6/2016 2:49:56 PM	11/6/2016 2:50:11 PM	vcenter65.flashcloud.local

ESXi 6.0 U2 and VMFS 5: Total time of 19 minutes, 43 seconds

Task Console

Task Name	Target	Status	Start Time	Completion Time	Server
Resignature unresolved VMFS volume	esxi07.flashcloud.l...	Completed	11/6/2016 2:58:57 PM	11/6/2016 3:18:40 PM	vcenter60u2.flashcloud.local
Resignature unresolved VMFS volume	esxi07.flashcloud.l...	Completed	11/6/2016 2:58:57 PM	11/6/2016 3:18:40 PM	vcenter60u2.flashcloud.local
Resignature unresolved VMFS volume	esxi07.flashcloud.l...	Completed	11/6/2016 2:58:57 PM	11/6/2016 3:18:40 PM	vcenter60u2.flashcloud.local
Resignature unresolved VMFS volume	esxi07.flashcloud.l...	Completed	11/6/2016 2:58:57 PM	11/6/2016 3:18:40 PM	vcenter60u2.flashcloud.local
Resignature unresolved VMFS volume	esxi07.flashcloud.l...	Completed	11/6/2016 2:58:57 PM	11/6/2016 3:18:40 PM	vcenter60u2.flashcloud.local
Resignature unresolved VMFS volume	esxi07.flashcloud.l...	Completed	11/6/2016 2:58:57 PM	11/6/2016 3:18:41 PM	vcenter60u2.flashcloud.local
Resignature unresolved VMFS volume	esxi07.flashcloud.l...	Completed	11/6/2016 2:58:57 PM	11/6/2016 3:18:40 PM	vcenter60u2.flashcloud.local
Resignature unresolved VMFS volume	esxi07.flashcloud.l...	Completed	11/6/2016 2:58:57 PM	11/6/2016 3:18:41 PM	vcenter60u2.flashcloud.local

esxcfg-volumes improvements

If a VMFS volume is comprised of multiple extents, the “esxcfg-volumes –l”command can be used to show any unresolved VMFS extents. This command traditionally took quite a long time to complete. This has improved dramatically with VMFS-6 and ESXi 6.5. With eight unresolved volumes in 6.0 U2, the command took 1 minute and 53 seconds to complete. With ESXi 6.5 and VMFS-6, the same command on an identically configured volume takes less than a second to complete.

Unmounting and Mounting improvements

In this next example, the “MountVmfsVolumeEx_Task” and “UnmountVmfsVolumeEx_Task” PowerCLI cmdlets are used to mount and unmount the same eight VMFS volumes from three ESXi hosts simultaneously. These tests were once again run with VMFS-5 on ESXi 6.0 U2 and then again with VMFS-6 on ESXi 6.5.

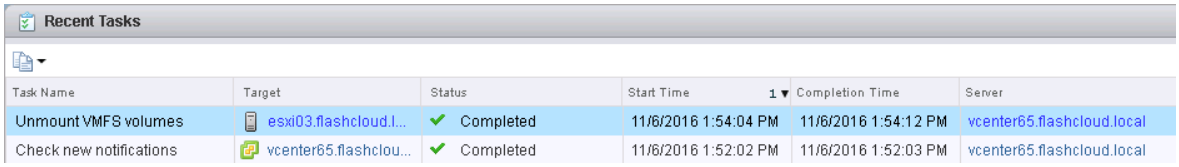
A sample snippet of the PowerCLI code here is shown here for completeness:

```
$esxihost = get-vmhost
$datastore = get-datastore mounttest*
$storageSystem0 = Get-View $esxihost[0].Extensiondata.ConfigManager.StorageSystem
$storageSystem1 = Get-View $esxihost[1].Extensiondata.ConfigManager.StorageSystem
$storageSystem2 = Get-View $esxihost[2].Extensiondata.ConfigManager.StorageSystem
$uuids = @()
foreach ($ds in $datastore)
{
    $uuids += $ds.ExtensionData.Info.vmfs.uuid
```

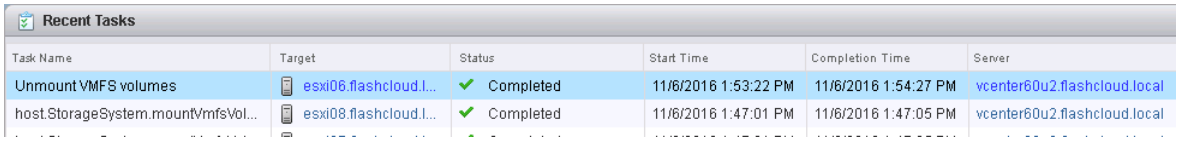
```
}
$StorageSystem0.UnmountVmfsVolumeEx_Task($uuids)
$StorageSystem1.UnmountVmfsVolumeEx_Task($uuids)
$StorageSystem2.UnmountVmfsVolumeEx_Task($uuids)
```

Across the board, unmount times were much better in ESXi 6.5 (8 seconds compared to 65 seconds). Mount times did not show any improvement, either on a single host or on multiple hosts.

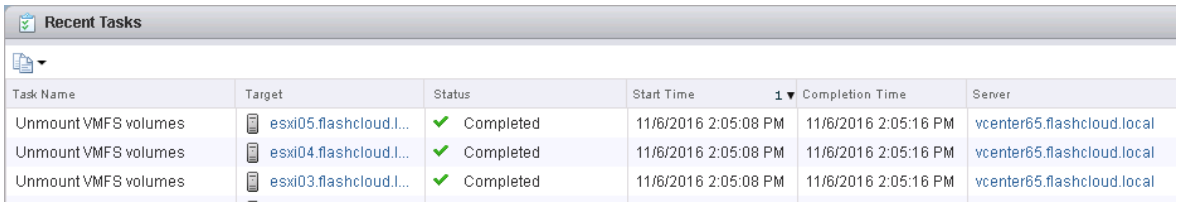
In the screenshots below, the first shows the single host unmount of eight VMFSvolumes on vCenter 6.5 taking 8 seconds:



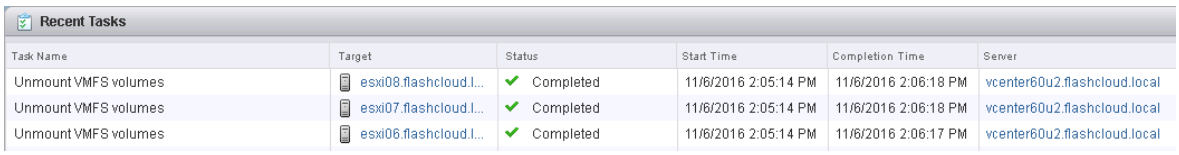
In vCenter 6.0 U2, the same unmount operation takes 1 minute, 5 seconds:



The same unmount operation is now run again, but this time the three ESXi hosts are simultaneous issuing unmount operations of the eight VMFS volumes. In the first example, we are showing vCenter 6.5, and the total time taken is 8 seconds:



In this example,we are using vCenter 6.0 U2 and the time taken is 1 minute, 4 seconds tocomplete the operations across the 3 hosts:



Upgrading from previous versions of VMFS to VMFS-6

Datastore filesystem upgrade from VMFS-5 (or previous versions) to VMFS-6 is not supported. Customers upgrading from older versions of vSphere to 6.5 release should continue to use VMFS-5 datastores (or older version) until they create new VMFS-6 datastores.

Since there is no direct 'in-place' upgrade of filesystem supported, customers should use Virtual Machine migration techniques such as Storage vMotion to move VMs from the old datastore to the new VMFS-6 datastore.

VMware has provided extensive documentation on how to migrate from earlier versions of VMFS to VMFS-6. Customers are strongly urged to refer to this official documentation when planning a migration to VMFS-6.

Hot Extend for VMDKs > 2TB

Prior to ESXi 6.5, virtual disks could only be extended if their size was below 2TB when the VM was powered on. If the size of a VMDK was 2TB or larger, or the expand operation caused it to exceed 2TB, the hot extend operation would fail. This required administrators to typically shut down the virtual machine to expand it beyond 2TB. The behavior has been changed in vSphere 6.5 and hot extend no longer has this limitation.

It is important to note that this does not require VMFS-6 or Virtual Machine HW version 13 to work. VMFS-5 will also support this functionality as long as ESXi is version at 6.5.

1.4 UNMAP

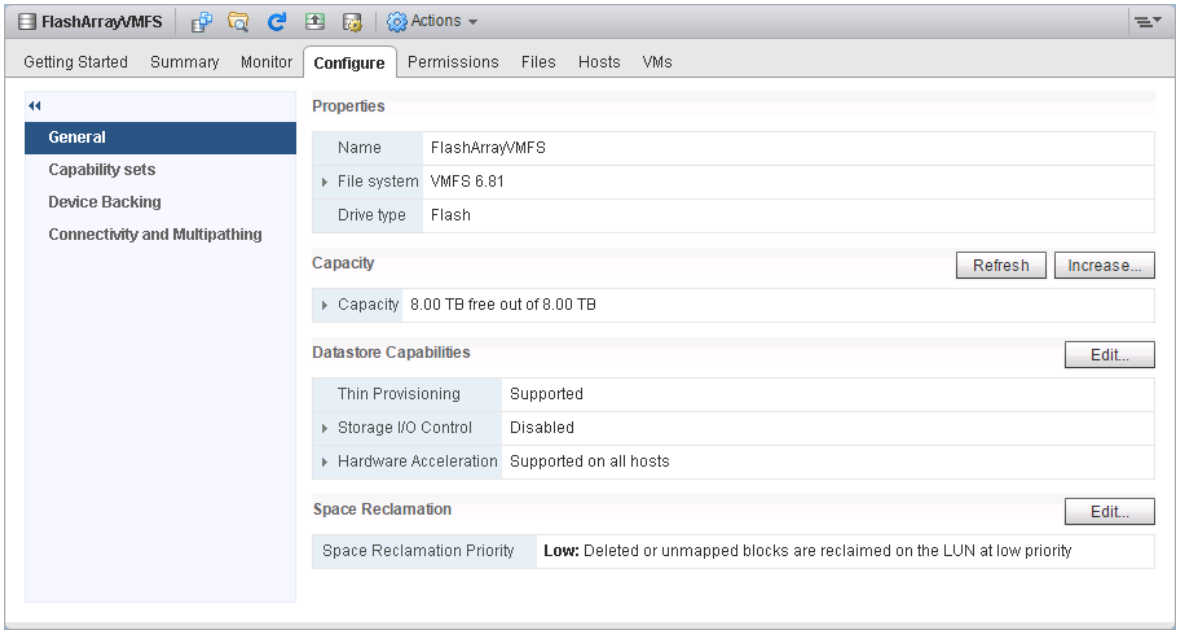
UNMAP

VAAI UNMAP was introduced in vSphere 5.0 to allow the ESXi host to inform the backing storage that files or VMs had be moved or deleted from a Thin Provisioned VMFS datastore. This allowed the backing storage to reclaim the freed blocks. There was no way of doing this previously, resulting in many customers with a considerable amount of stranded space on their Thin Provisioned VMFS datastores.

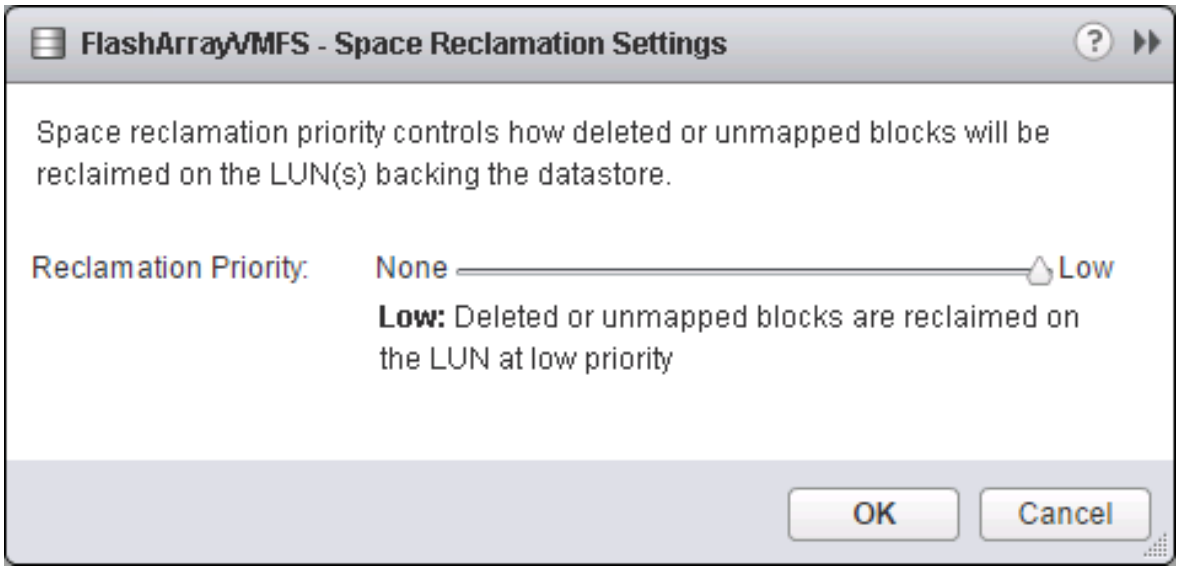
In vSphere 6.0, additional improvements to UNMAP were introduced which facilitated the reclaiming of stranded space from within a Guest OS. Effectively, if you have deleted files within a Guest OS, and your VM is thinly provisioned, you can tell the backing storage that you are no longer using these blocks. This allows the backing storage to reclaim this capacity for other uses.

Introducing Automated UNMAP Space Reclamation

In vSphere 6.5, there is now an automated UNMAP crawler mechanism for reclaiming dead or stranded space on VMFS datastores. This is a big change from previous versions of vSphere, where UNMAP was run manually. Now UNMAP will run continuously in the background. The vSphere UI can be used to see if Space Reclamation is running against a particular datastore as shown below:



There are currently two settings available via the vSphere UI for setting the reclamation priority.These are None and Low. With the default setting of Low, the expectation that any blocks that are no longer used will be reclaimed within 12 hours.



Reclaim priority is only on or off. Low implies reclaim is enabled, None is disabled.

TRIM Handling

TRIM is the ATA equivalent of SCSI UNMAP. A TRIM operation gets converted to UNMAP in the I/O stack, which is SCSI. However, there are some issues with TRIM getting converted into UNMAP. UNMAP work at certain block boundaries on VMFS, whereas TRIM does not have such restrictions.

While this should be fine on VMFS-6, which is now 4K aligned, certain TRIMs converted into UNMAPs may fail due to block alignment issues on previous versions of VMFS.

Linux Guest OS SPC-4 support

As mentioned earlier, there was some limited in-guest UNMAP support introduced in vSphere 6.0 to reclaim in-guest dead space natively. This was limited to Windows 2012 R2 initially, primarily because of the vSCSI version. Linux distributions check the SCSI version, and unless it is version 5 or greater, it does not send UNMAPs. With SPC-4 support introduced in vSphere 6.5, Linux Guest OS'es will now also be able to issue UNMAPs.

Auto UNMAP Limitations and Considerations

There are some limitations and considerations when it comes to using UNMAP, and which will prevent UNMAP from functioning. The first of these is the UNMAP granularity on the storage array itself. The granularity of the reclaim is set to 1MB chunk. Automatic UNMAP is not supported on arrays with UNMAP granularity greater than 1MB. Auto UNMAP feature support is footnoted in the VMware Hardware Compatibility Guide (HCL).

Another issue is related to block alignment. VMDK alignment is aligned on 1 MB block boundaries. However un-alignment may still occur within the guest OS filesystem. This may also prevent UNMAP from working correctly. A best practice is to align guest OS partitions to the 1MB granularity boundary.

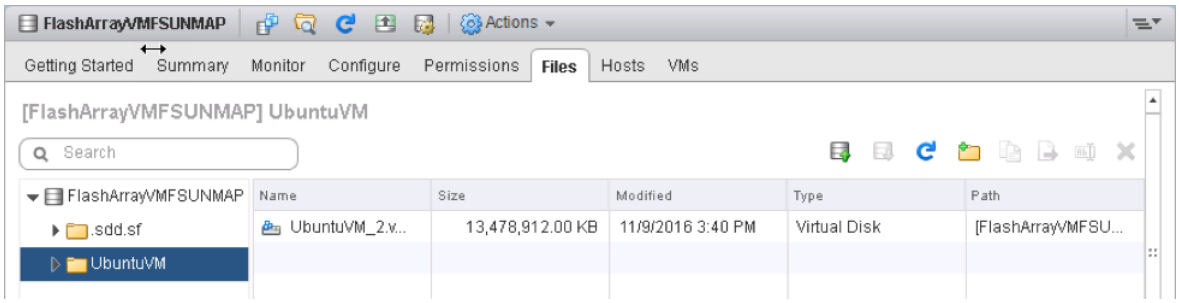
In-guest UNMAP in action

In vSphere 6.5, SCSI support was added to enable in-guest UNMAP with Linux-based virtual machines.

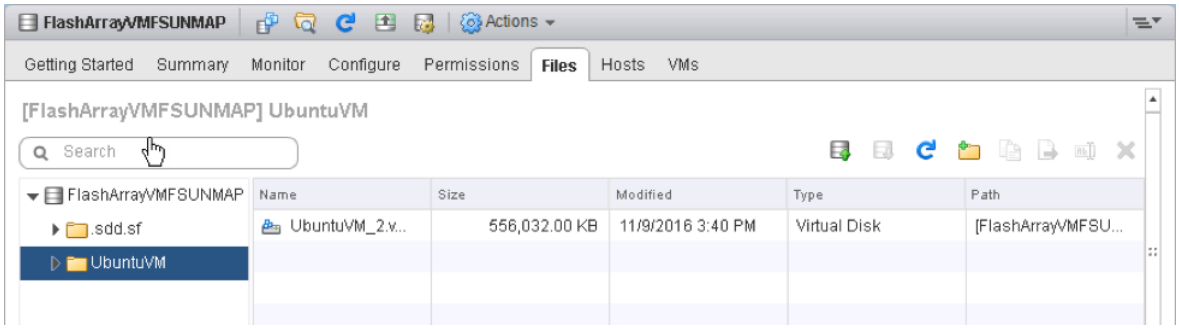
In the following scenario, an EXT4 filesystem was created and mounted in an Ubuntu guest with the discard option, as shown below. This virtual disk was also thinly provisioned.

```
$ sudo mount /dev/sdd /mnt/unmaptest -o discard
```

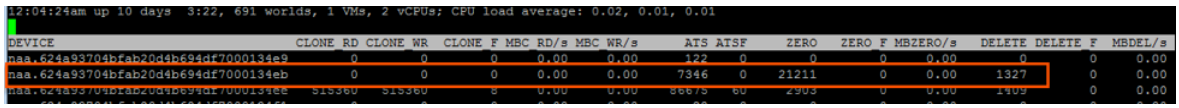
Four files were added to the filesystem of about 13.5 GB in aggregate size. The thin virtual disk grew to 13.5 GB after the file placement:



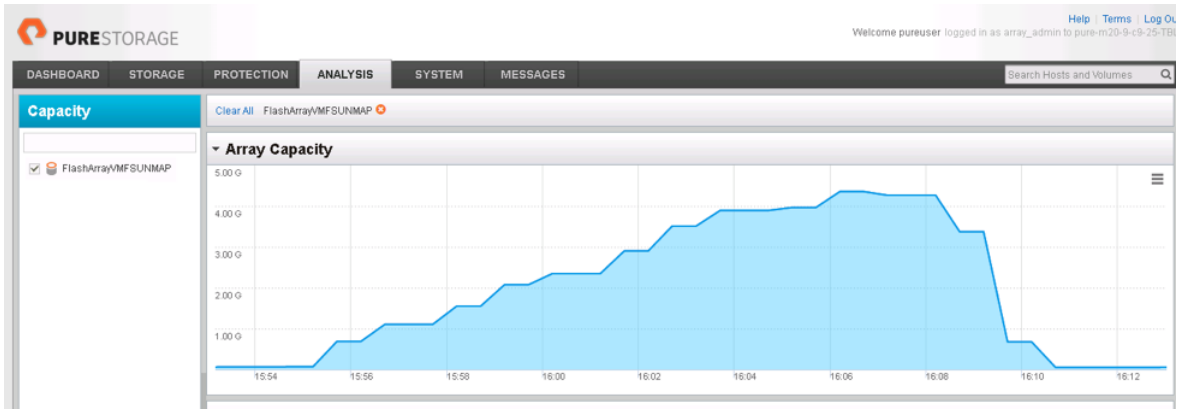
The files were subsequently removed with the "rm" command. Due to the discard option being used, the space was reclaimed. The virtual disk shrunk by 13 GB:



ESXi then issued its own UNMAP command. ESXTOP shows UNMAP being issued in the DELETE column:



Finally, if the array is also checked, one can see that the space is reclaimed on the VMFS volume that contains the thin virtual disk.



In vSphere 6.5,administrators can reclaim dead space in both the Guest OS, and on the datastores.

1.5 Storage I/O Control v2

Storage I/O Control v2

Storage I/O Control (SIOC) was initially introduced in vSphere 4.1 to provide I/O prioritization of virtual machines running on a cluster of ESXi hosts that had access to shared storage. It extended the familiar constructs of shares and limits, which existed for CPU and memory, to address storage utilization through a dynamic allocation of I/O queue slots across a cluster of ESXi servers. The purpose of SIOC is to address the 'noisy neighbor' problem, i.e. a low priority virtual machine impacting other higher priority virtual machines due to the nature of the application and its I/O running in that low priority VM.

vSphere 5.0 extended SIOC to provide cluster-wide I/O shares and limits for NFS datastores. This means that no single virtual machine should be able to create a bottleneck in any environment regardless of the type of shared storage used. SIOC automatically throttles a virtual machine which is consuming a disparate amount of I/O bandwidth when the configured **latency threshold** has been exceeded. To allow other virtual machines receive their fair share of I/O bandwidth on the same

datastore, a share based fairness mechanism has been created which now is supported on both NFS and VMFS.

vSphere 5.1 introduced a new SIOC feature called Stats Only Mode. When enabled, it doesn't enforce throttling but gathers statistics to assist Storage DRS. Storage DRS now has statistics in advance for new datastores being added to the datastore cluster & can get up to speed on the datastores profile/ capabilities much quicker than before.

Another 5.1 feature was Automatic Threshold Computation. The default latency threshold for SIOC is 30ms. Not all storage devices are created equal so this default was chosen as a sort of "catch-all". There are certain devices which will hit their natural contention point much earlier than others, for example All Flash Arrays, in which case the threshold should be lowered by the user. However, manually determining the correct latency can be difficult for users. This gave rise to the need for the latency threshold to get automatically determined at a correct level for each device. Using the I/O injector modeling of SIOC, peak throughput and corresponding latency of a datastore is measured. The latency threshold value at which Storage I/O Control will kick in is then set to 90% of this peak value (by default). vSphere administrators can change this 90% to another percentage value or they can still input a millisecond value if they so wish.

The default latency threshold for SIOC can be reduced to as low as 5ms.

SIOC V1 Overview

SIOC V1 is disabled by default. It needs to be enabled on a per datastore level, and it is only utilized when a specific level of latency has been reached. By default, the latency threshold for a datastore is set to 30ms, as mentioned earlier. If SIOC is triggered, disk shares (aggregated from all VMDKs using the datastore) are used to assign I/O queue slots on a per host basis to that datastore. In other words, SIOC limits the number of IOs that a host can issue. The more VMs/VMDKs that run on a particular host, the higher the number of shares, and thus the higher the number of IOs that that particular host can issue. The throttling is done by modifying the device queue depth of the various hosts sharing the datastore. When the period of contention passes, and latency returns to normal values, the device queue depths are allowed to return to default values on each host.

SIOC V2 Introduction

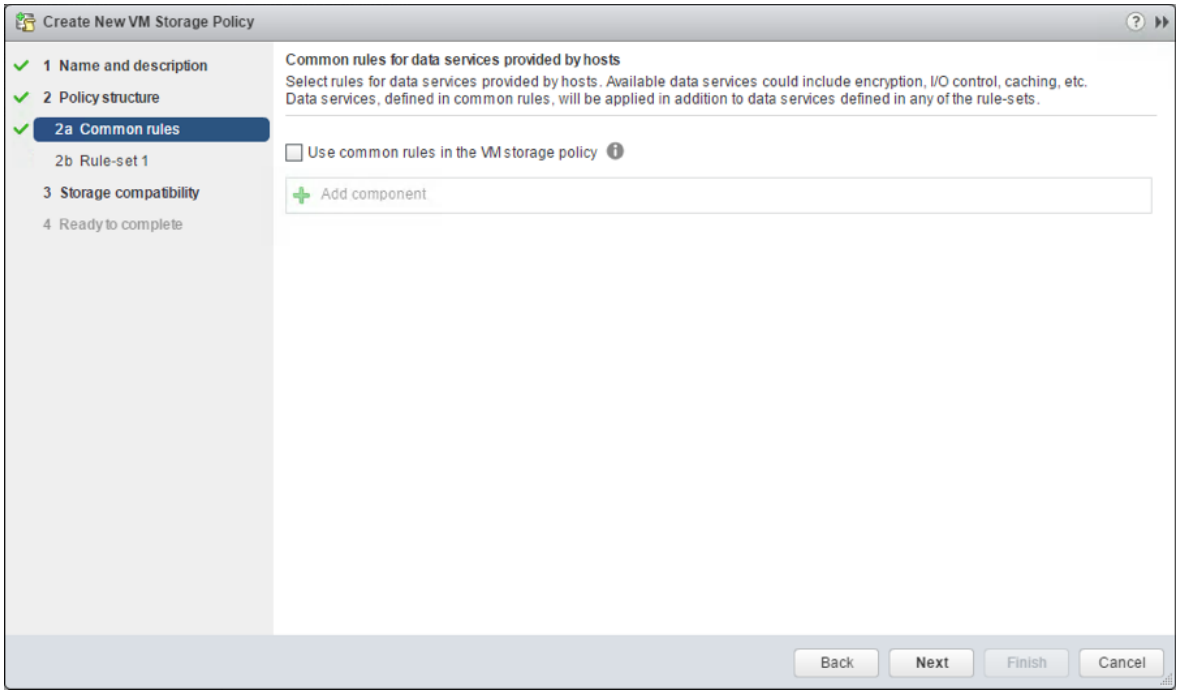
Before describing SIOC V2, it should be highlighted that SIOC V1 and SIOC V2 can co-exist on vSphere 6.5. This makes it much simpler when considering upgrades, or migrations between versions. With that in mind, SIOC V2 is considerably different from a user experience perspective when compared to V1. SIOCv2 is implemented using IO Filter framework Storage IO Control category. SIOC V2 can be managed using SPBM Policies. What this means is that you create a policy which contains your SIOC specifications, and these policies are then attached to virtual machines.

Creating an SIOC policy based

Creating an SIOC policy is done is exactly the same way as building a storage policy for VSAN or Virtual Volumes. Select the VM Storage Policy from the vSphere client home page, and from there select the option to create a new VM Storage Policy. VM Storage Policies in vSphere 6.5 has a new option called "Common Rules". These are used for configuring data services provided by hosts, such as Storage I/O Control and Encryption.

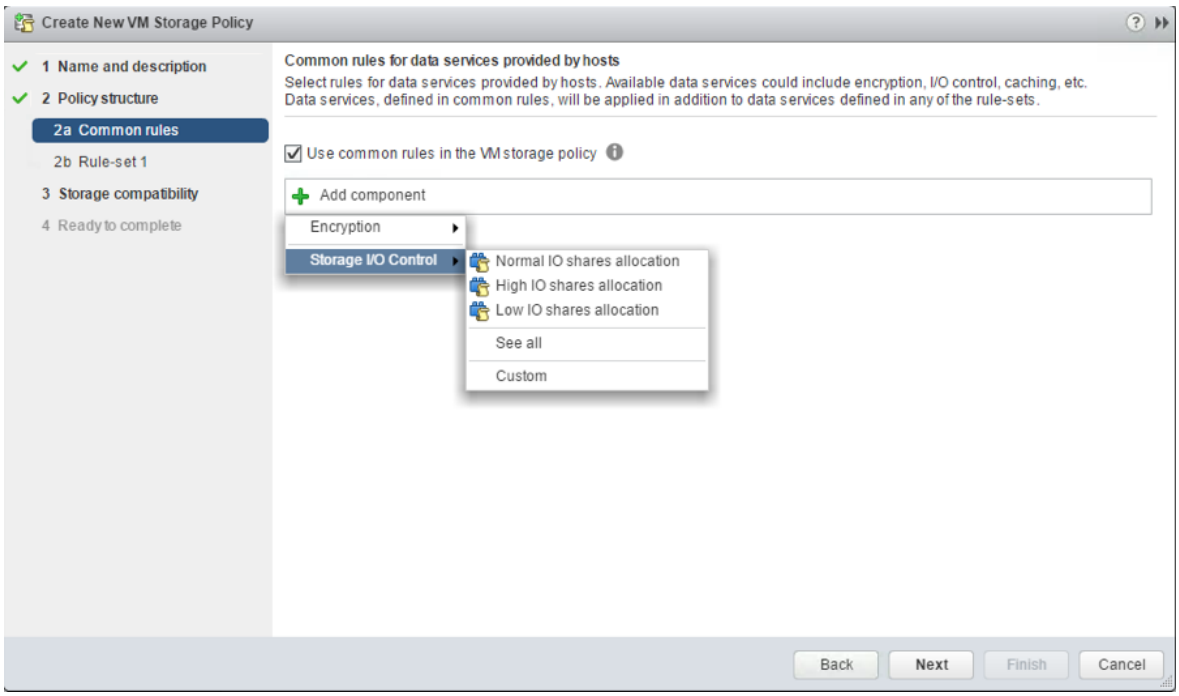
Use common rules in the VM storage policy

The first step is to click on the check box to enabled common rules. This will then allow you to add components, such as SIOC, to the policy.



Add Component – Storage I/O Control

In vSphere 6.5, there are two components available for common rules, Encryption and Storage I/O Control. Select Storage I/O Control in this case. Now you can select Normal, High, Low or Custom shares allocation.



This table describes the different Limits,Shares and Reservations associated with each setting:

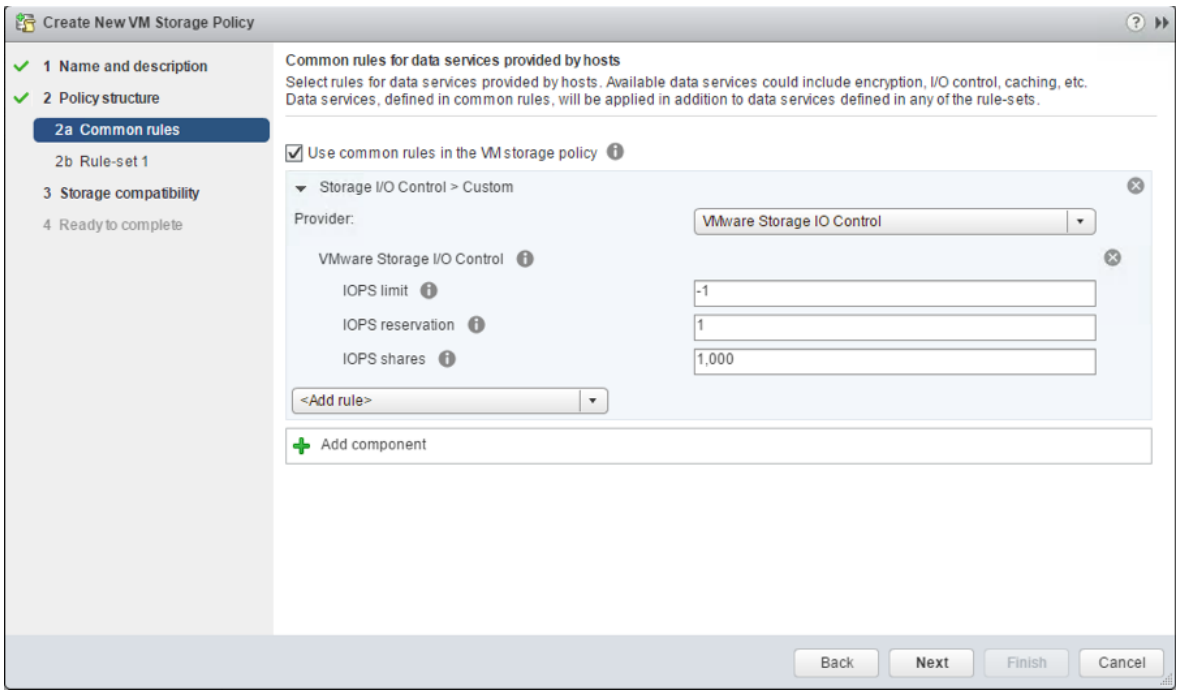
	HIGH	NORMAL	LOW

Limits	100,000	10,000	1,000
Reservation	100	50	10
Shares	2,000	1,000	500

When the policy has been created, it may be assigned to newly deployed VMs during provisioning,or to already existing VMs by assigning this new policy to the whole VM (or just an individual VMDK) by editing its settings. One thing to note is that IO Filter based IOPS does not look at the size of the IO. For example, there is no normalization so that a 64K IOP is not equal to 2 x 32K IOPS. It is a fixed value of IOPS irrespective of the size of the IO.

Custom Allocation

If neither of the values in the Normal, High, Low allocations is appropriate, there is the ability to create custom settings for these values. In a custom setting, IOPS limit and IOPS reservation are both set to -1, implying unlimited. These may be modified as required.



Advanced Options

SchedCostUnit

This is an advanced parameter that was created for SIOC V1 only. SIOC V2 does not have SchedCostUnit implemented. For V1, SchedCostUnit determines the unit size (normalized size) of an IO operation for scheduling, and it is currently a constant value of 32K. This constant value, however, may not satisfy different requirements from different customers. Some customers may want to set this unit size to 4K. Other customers may want to set it up to 256K.

To satisfy these different requirements, SchedCostUnit is now configurable. It defaults to an IO size value of 32K, and allowable values range between 4K to 256K.

The SchedCostUnit dictates how requests are counted. A request with size <= SchedCostUnit counts as a single I/O. Anything greater than SchedCostUnit will be counted as 2 or more requests.

For example, by changing the SchedCostUnit from 32K to 64K, the number of IOPS observed will halve. The size of the IO can be set using the:

```
"esxcli system settings advanced set -o /Disk/SchedCostUnit -i 65536"
```

and verified by using the"

```
"esxcli system settings advanced list -o /Disk/SchedCostUnit"
```

command. SIOC V2 counts guest IO directly. IOPS will be counted based on IO count, regardless of the IO size.

SchedReservationBurst

When limits are set on VMDKs, requests could have high average latency because the limit was enforced at a high (per request) granularity. This was due to the strict enforcement on a VM getting its share of IOs in interval of 1 second/L, where L is the user specified limit. The issue is more visible in fast storage, such as flash arrays. It was noted that SIOC V2 did not perform well when presented with a "bursty" workload on fast storage.

This SchedReservationBurst setting relaxes that constraint so a VM get its share of IOs at any time during a 1 second window, rather than enforce strict placement of IOs in intervals of 1/L. BURST option is turned-on by default.

SIOC V2 Limitations

In this initial release of SIOC V2 in vSphere 6.5, there is no support for vSAN or Virtual Volumes. SIOC v2 is only supported with VMs that run on VMFS and NFS datastores.

1.6 vSphere VM Encryption

vSphere VM Encryption

Introduction to vSphere VM Encryption

vSphere 6.5 introduces a new VM encryption mechanism. This encryption mechanism is implemented in the hypervisor, making vSphere VM encryption agnostic to the Guest OS. This not only encrypts the VMDK, but it also encrypts the VM Home directory contents, e.g. VMX file, metadata files, etc.

One other important point to highlight is that vSphere VM Encryption in vSphere 6.5 is also policy driven. As we have previously seen with SIOC v2, administrators need to select an appropriate policy that contains encryption and associate this policy with a virtual machine if encryption is desired. This policy driven approach also makes it very easy to ascertain whether or not encryption is associated with a particular VM from a vSphere level, rather than having to check each and every VM.

One other aspect to highlight is the key management, which is based on Key Management Interoperability Protocol (KMIP) version 1.1. VMware vCenter now contains a KMIP client, which works with many common KMIP key managers (KMS).

VM Encryption is datastore and virtual Hardware version agnostic. No in guest agents need to be installed or managed.

vSphere VM Encryption implementation

There are two levels of key encryption going on in vSphere VM Encryption. The first is when a VMDK has a policy with encryption associated with it. This key is what might be termed the "internal ESXi key", which is randomly generated on a per VM basis. This internal key is officially referred to as the DEK, or Data Encryption Key. These are the keys which encrypts all of the VM's files.

This key is then encrypted with the key from the key manager (KMS). This KMS key is referred to as the "tenant encryption key" or the KEK, short for Key Encryption Key. This is the master key that is stored in the KMS. It is responsible for encrypting keys.

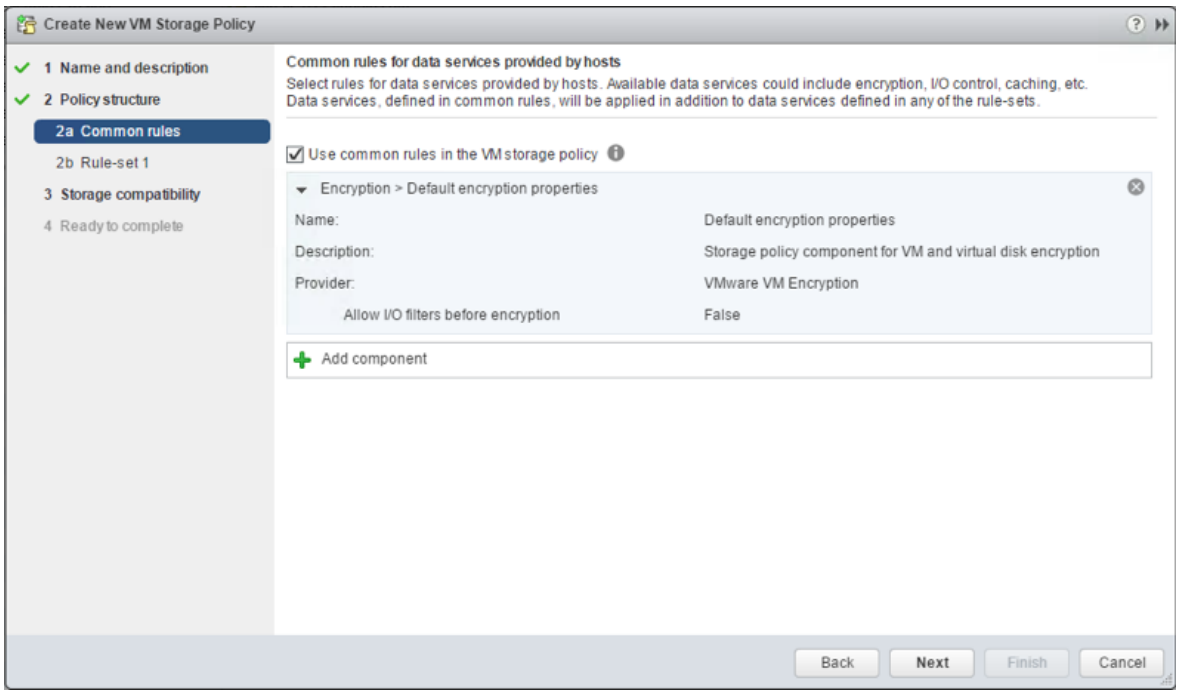
Now when the VM is powered on, the vCenter server retrieves the "tenant encryption key" from the KMS, and this is sent to the encryption module on the ESXi host to unlock the "internal ESXi key", allowing the VM to run.

Creating and Applying vSphere VM Encryption policy

Creating a policy is much similar to what we have seen in the Storage I/O Control section seen earlier in this white paper. Once again, common rules must be enabled. This will then allow you to add components, such as Encryption, to the policy.

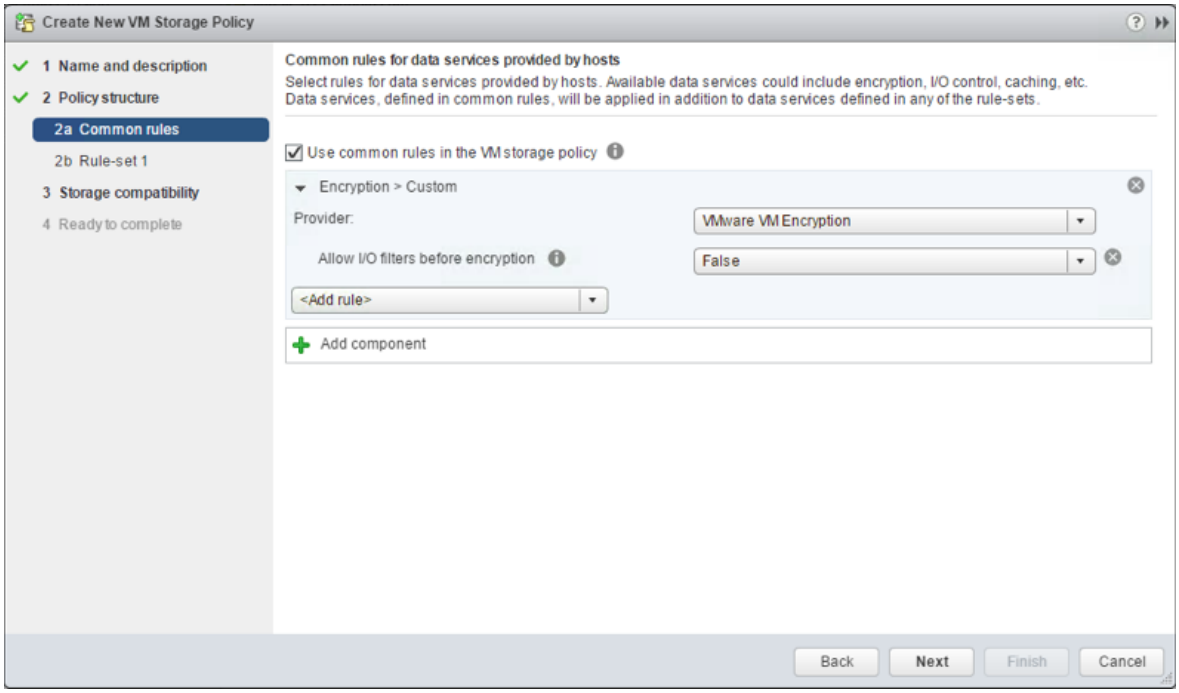
Default Encryption Policy

The default encryption policy is quite simple. The Provider is VMware VM Encryption, and the setting "Allow I/O filters before encryption" is set to false. This means that any filters that are also applied to the VM I/O are applied after encryption has taken place.



Custom Encryption Policy

The only setting that can be modified in the custom encryption policy at this time is the allow I/O filters before encryption.



VM Encryption Limitations and Considerations

There are some limitations and considerations associated with vSphere VM Encryption in vSphere 6.5, some of which are highlighted here.

Since VMware does not own the KMS, customers are urged to engage with the KMS provider for conversations around backup, DR, recovery, etc. of their KMS. It is critical that some plan is in place to retrieve the encryption keys in the event of a failure, or you may render your VMs unusable.

Administrators should not encrypt their vCenter server. This will lead to a “chicken-and-egg” situation where you need vCenter to boot so it can get the KEK from the KMS to unencrypt its files, but it will not be able to boot as its files are encrypted. Remember, vCenter server does not manage encryption. It is only a client of the KMS.

With VM Home encrypted, only administrators that have encryption privileges will be able to access the console of the virtual machine. One misconception with encryption of files in VM Home is that the folder itself is not encrypted. Only a few sensitive files in the VM Home folder are encrypted. Some (non-sensitive) VM files and log files are not encrypted. However, core dumps are encrypted on ESXi hosts with encrypted VMs.

Encrypted virtual machines cannot be exported to an OVF, nor can an encrypted VM be suspended.

If planning to use vSphere VM Encryption, it is strongly recommended that you refer to the official documentation.

1.7 New Virtual Storage Hardware

New Virtual Storage Hardware

VSCSI support for SPC-4

Prior to vSphere 6.5, virtual disks only supported SCSI Primary Command specification SPC-2. This provided a limited command set. With the introduction of SPC-4, virtual disks now support a much improved command set. In particular, this introduces support for UNMAP operations from within the Guest OS, as discussed earlier in this paper. However, SPC-4 also introduces support for the WRITE SAME primitive. Historically, Linux OSes included workarounds for WRITE SAME not being supported on VMware virtual disks, but with vSphere 6.5 and SPC-4 support, a different device model is now reported to the Guest OS which will enable Linux Guest OSes to use WRITE SAME on VMware virtual disks.

Virtual NVMe Device

Virtual NVMe device is a new virtual storage HBA which is designed for providing lower IO overhead and scalable IOs for all flash SAN/vSAN storages. NVMe devices are increasingly becoming the primary storage interface for flash-based storages due to its well-designed, low overhead storage protocol and its ability to support scalable IO on multi-core processors. As recent Linux and Windows guest OSes provide much lower I/O stack overhead by skipping SCSI I/O stacks, and leveraging multiple queues with NVMe devices, virtual NVMe device allows VMs to take advantage of such in-guest IO stack improvements. Virtual NVMe device provides 30-50% lower CPU cost per I/O and 30-80% higher IOPS compared to virtual SATA device on local PCIe SSD devices.

Here is the supported configuration information of virtual NVMe device.

- Supports NVMe Specification v1.0e mandatory admin and I/O commands
- Maximum 15 namespaces per controller: Each namespace is mapped to a virtual disk. Enumerated as nvme0:0, …, nvme0:15
- Maximum 4 controllers per VM: Enumerated as nvme0, nvme1, …, nvme3.
- Maximum 16 queues (1 admin + 15 I/O queues) and 16 interrupts.
- Maximum 256 queue depth (4K in-flight commands per controller)
- Interoperability with all existing vSphere features, except SMP-FT.

1.8 NFS 4.1

NFS 4.1

Hardware Acceleration/VAAI-NAS Improvements

One of the major NFS 4.1 client enhancements in vSphere 6.5 is to introduce support for hardware acceleration. In other words, certain operations may now be offloaded to the storage array. This comes in the form of a plugin to the ESXi host that is developed/provided by the storage array partner. Refer to your NAS storage array vendor for further information.

Kerberos IPv6 Support

There is now full IPv6 support with Kerberos in vSphere 6.5.

Kerberos AES Encryption Support

NFS 4.1 Kerberos adds AES encryption support in vSphere 6.5. The following Advanced Encryption Standards (AES) are now supported:

- AES256-CTS-HMAC-SHA1-96
- AES128-CTS-HMAC-SHA1-96

The DES-CBC-MD5 encryption type is not supported with NFSv4.1 in vSphere 6.5.

Kerberos Integrity SEC_KRB5I Support

Kerberos Integrity is a new feature in 6.5. vSphere 6.5 introduces Kerberos Integrity SEC_KRB5I. This feature uses checksum to protect NFS data.

1.9 ISCSI Improvements

ISCSI Improvements

ISCSI Routing

The Software iSCSI implementation in ESXi 6.5 now supports having the iSCSI initiator and the iSCSI target residing in different network subnets. This essentially means that we now support the routing of iSCSI connections and sessions.

Port Binding

Software iSCSI can leverage separate gateways per VMkernel in vSphere 6.5. Customers with such a configuration can use port binding to reach targets in different subnets.

ISCSI and NSX interoperability

Another new feature in vSphere 6.5 is that iSCSI can now work with the NSX “opaque” switch, as well as the traditional stand-alone vSwitch (VSS) and the Distributed vSwitch (DVS). This is true for both the Software iSCSI adapters and dependent iSCSI adapters.

UEFI iSCSI Boot

VMware now supports UEFI (Unified Extensible Firmware Interface) iSCSI Boot on Dell 13th generation servers with Intel x540 dual port Network Interface Card (NIC). On the System BIOS select Network Settings, followed by UEFI iSCSI Device Settings. In the Connection Settings, you need to populate initiator and target settings, as well as any appropriate VLAN and CHAP settings if required. This device will now appear in the list of options in the UEFI Boot Settings. The NIC Configuration must then have its Legacy Boot Protocol set to *iSCSI Primary*, and also be populated with initiator and target settings. You can now install your ESXi image and use any of the LUNs from the iSCSI target to install to. Subsequent reboots will boot from the ESXi image on the iSCSI LUN.

1.10 Acknowledgements

Acknowledgements

The author would like to acknowledge the assistance of many members of the core storage team at VMware for their assistance in putting this paper together. In no particular order, the author is grateful for contributions from Jeff Glasson, Richard Harry, Ravi Cherukupalli, Komal Desai, Bryan Branstetter, Dimitris Skourtis, Prasad Jangam, Rohan Pasalkar, Mike Foley, Salil Suri, Swapneel Kekre, Arun Lakshmipathy, Deepak Parthasarathy, Rahul Dev, Jinpyo Kim, Suds Jain, Narasimha Krishnakumar, Shashank Rajvanshi and Paudie O'Riordan.

1.11 A Special Thank You

A Special Thank You

A special word of thanks to Cody Hosterman of Pure Storage who assisted us with the setup and testing of many of the new vSphere 6.5 Core Storage features. Thank you so much Cody.

1.12 About The Author

About The Author

Cormac Hogan works in the Office of the CTO in the Storage and Availability Business Unit (SABU) at VMware. He has previously held roles in VMware's Technical Marketing and Technical Support organizations. He has written a number of storage related white papers and have given numerous presentations on storage best practices and vSphere storage products and features. He is also the co-author of the "Essential vSAN" book published by VMware Press. Cormac blogs about many different storage and virtualization topics at http://cormachogan.com