

Using VMware vSphere Storage APIs for Array Integration with Dell PowerMax

Increasing operational efficiency with VMware and Dell PowerMax

Abstract

This paper discusses how VMware's vSphere Storage APIs for Array Integration, also known as VAAI, offloads various virtual machine operations from the host to the Dell PowerMax®.

October 2024

Revisions

| Date | Description |
|----------------|---|
| March 2023 | Initial release of new template |
| November 2023 | PowerMaxOS 10.1.0.0, vSphere 8.0U2 |
| March 2024 | Explain how to add multiple claim rules for XCOPY if multiple arrays are attached to the same ESXi host |
| September 2024 | PowerMaxOS 10.2.0.0, vSphere 8.0U3 |

Acknowledgments

Author: Drew Tonnesen, Dell Engineering

The information in this publication is provided “as is.” Dell Inc. makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any software described in this publication requires an applicable software license.

This document may contain certain words that are not consistent with Dell's current language guidelines. Dell plans to update the document over subsequent future releases to revise these words accordingly.

This document may contain language from third party content that is not under Dell's control and is not consistent with Dell's current guidelines for Dell's own content. When such third-party content is updated by the relevant third parties, this document will be revised accordingly.

Table of contents

| | |
|--|----|
| Revisions..... | 2 |
| Acknowledgments..... | 2 |
| Table of contents | 4 |
| Executive summary..... | 6 |
| 1 VAAI primitives | 8 |
| 1.1 Hardware-accelerated Full Copy or XCOPY | 8 |
| 1.2 Hardware-accelerated Block Zero | 9 |
| 1.3 Hardware-assisted locking | 11 |
| 1.3.1 ATS heartbeat..... | 13 |
| 1.4 UNMAP | 15 |
| 1.4.1 Datastore | 15 |
| 1.4.2 Guest OS..... | 16 |
| 1.5 VAAI and HYPERMAX OS/PowerMaxOS version support..... | 16 |
| 1.6 Compression/Deduplication/Data Reduction..... | 18 |
| 1.7 Host IO Limits | 18 |
| 1.8 Viewing VAAI support..... | 19 |
| 1.9 VAAI Statistics | 22 |
| 1.9.1 UNMAP Example..... | 23 |
| 1.9.2 XCOPY Example | 24 |
| 1.10 Enabling the Storage APIs for Array Integration | 26 |
| 1.10.1 Full Copy, Block Zero, and Hardware-assisted locking..... | 26 |
| 2 NVMe over Fabrics (NVMeoF)..... | 28 |
| 2.1 VAAI..... | 28 |
| 3 eNAS and PowerMax File | 29 |
| 3.1 Extended stats | 30 |
| 3.2 Nested clones | 30 |
| 4 Full Copy Detail | 32 |
| 4.1 Default copy size in vSphere 7 and 8 | 32 |
| 4.2 XCOPY claim rules in vSphere 7 and 8 | 33 |
| 4.2.1 Custom vendor rule | 34 |
| 4.2.2 Virtual Storage Integrator | 37 |
| 4.2.3 Thin vmdks | 39 |
| 4.2.4 Claim rule results | 39 |
| 4.3 SRDF and XCOPY | 42 |

| | | |
|-------|---|-----------|
| 4.4 | XCOPY with multiple arrays on an ESXi host | 43 |
| 4.5 | Caveats for using hardware-accelerated Full Copy | 43 |
| 4.5.1 | SRDF/Metro specific..... | 44 |
| 4.5.2 | Compression/Deduplication (Data Reduction) specific | 45 |
| 5 | Block Zero Detail | 46 |
| 5.1.1 | Eagerzeroedthick disks | 46 |
| 5.1.2 | Zeroedthick virtual disks | 48 |
| 5.2 | Caveats for using hardware-accelerated Block Zero | 48 |
| 6 | UNMAP Detail | 49 |
| 6.1 | Automated UNMAP | 49 |
| 6.1.1 | Reclamation priority..... | 51 |
| 6.1.2 | Changing reclamation values in vSphere 7 and 8..... | 55 |
| 6.1.3 | Reclamation rate changes in vSphere 8 | 55 |
| 6.1.4 | Reclamation hosts in 8.0U3..... | 56 |
| 6.1.5 | Automatic UNMAP support for SESparse | 56 |
| 6.2 | Recommendations..... | 56 |
| 6.3 | Manual UNMAP after VM deletion..... | 57 |
| 6.4 | UNMAP internals | 61 |
| 6.4.1 | Recommendation..... | 61 |
| 6.5 | Caveats for using UNMAP..... | 61 |
| 7 | Monitoring VAAI operations with ESXTOP | 63 |
| 7.1 | Monitoring VAAI with NFS | 66 |
| 8 | Performance | 68 |
| 8.1 | VDI..... | 68 |
| 9 | Conclusion..... | 69 |
| A | Technical support and resources | 70 |
| A.1 | Dell..... | 70 |
| A.2 | VMware..... | 70 |

Executive summary

VMware vSphere Storage APIs is a family of APIs used by third-party hardware, software, and storage providers, such as Dell, to develop components that enhance several vSphere features and solutions. This paper focuses on one component of those APIs known as VMware's Storage APIs for Array Integration (VAAI - also formerly known as vStorage APIs). Dell has four main API storage integrations as part of VAAI. These integrations are available with the HYPERMAX OS on the VMAX3/VMAX All Flash, and the PowerMaxOS on the PowerMax when running with VMware vSphere 7.0 and higher.¹ VAAI, by default, provides the ability to offload specific storage operations to the Dell PowerMax to increase both overall system performance and efficiency.² The four main VMware's Storage APIs supported by PowerMax are:

- **Full Copy** - This feature delivers hardware-accelerated copying of data by performing all duplication and migration operations on the array. Customers can achieve faster data movement via VMware Storage vMotion®, and virtual machine creation and deployment from templates and virtual machine cloning. The term Full Copy is used interchangeably with SCSI command that is issued for it, XCOPY.
- **Block Zero** - This feature delivers hardware-accelerated zero initialization, reducing common input/output tasks such as creating new virtual machines. This feature is especially beneficial when creating fault-tolerant (FT)-enabled virtual machines or when performing routine application-level Block Zeroing.
- **Hardware-assisted locking** - This feature delivers improved locking controls on Virtual Machine File System (VMFS), allowing far more virtual machines per datastore and shortened simultaneous block virtual machine boot times. This improves performance of common tasks such as virtual machine migration, powering many virtual machines on or off and creating a virtual machine from a template. This feature is covered only briefly in this paper.
- **UNMAP** - This feature enables the reclamation of blocks of thin-provisioned LUNs by informing the array that specific blocks are obsolete. VMware offers both Guest OS UNMAP and automated UNMAP at the VMFS level.

In addition to the four main APIs, VAAI includes an API for Thin Provisioning. Thin Provisioning is enabled by default. Thin Provisioning encompasses a couple different capabilities. First, the API allows ESXi to tell the array when there is reclaimable space on a thin provisioned LUN. This is essential for UNMAP. Second, the API supports out of space errors. This means the array can notify ESXi when it is running out of space. The out of space errors capability is sometimes referred to as “Stun & Resume” because when a VM runs out of space on the array, rather than crashing, it will enter a “stunned” state until the user increases space (or powers off the VM) on the array allowing the VM to “resume.”

To determine whether a device supports the Thin Provisioning API, query it with `esxcli`:

```
esxcli storage core device list
```

One of the columns is called “Thin Provisioning Status.” When the API is supported, this column will return “yes,” when not supported the status will be either “no” or “unknown.” In the screen below, [Figure 1](#), an array supported device is shown on the left, while an unsupported, local device is on the right.

¹ vSphere 6.x releases are no longer covered.

² Though only the PowerMax array is mentioned in the paper, VAAI is also supported on the other VMAX platforms. Refer to Table 1 for required code levels for the Dell platforms.

```
10.228.245.115 - PuTTY
naa.60000970000197700103533030303735
Display Name: EMC Fibre Channel Disk (naa.60000970000197700103533030303735)
Has Settable Display Name: true
Size: 102401
Device Type: Direct-Access
Multipath Plugin: NMP
Devfs Path: /vmfs/devices/disks/naa.60000970000197700103533030303735
Vendor: EMC
Model: SYMMETRIX
Revision: 5977
SCSI Level: 5
Is Pseudo: false
Status: on
Is RDM Capable: true
Is Local: false
Is Removable: false
Is SSD: true
Is VVOL PE: false
Is Offline: false
Is Perennially Reserved: false
Queue Full Sample Size: 0
Queue Full Threshold: 0
Thin Provisioning Status: yes
Attached Filters: VAAI_FILTER
VAAI Status: supported
Other UIDs: vml.02000800006000097000019770010353303030373553594d4d4554
Is Shared Clusterwide: true
Is Local SAS Device: false

10.228.245.115 - PuTTY
naa.600605b0049aca50177c7bd707efe3a4
Display Name: Local LSI Disk (naa.600605b0049aca50177c7bd707efe3a4)
Has Settable Display Name: true
Size: 952320
Device Type: Direct-Access
Multipath Plugin: NMP
Devfs Path: /vmfs/devices/disks/naa.600605b0049aca50177c7bd707efe3a4
Vendor: LSI
Model: MR9261-8i
Revision: 2.12
SCSI Level: 5
Is Pseudo: false
Status: on
Is RDM Capable: false
Is Local: true
Is Removable: false
Is SSD: false
Is VVOL PE: false
Is Offline: false
Is Perennially Reserved: false
Queue Full Sample Size: 0
Queue Full Threshold: 0
Thin Provisioning Status: unknown
Attached Filters:
VAAI Status: unsupported
Other UIDs: vml.0200000000600605b0049aca50177c7bd707efe3a445239323631
Is Shared Clusterwide: false
Is Local SAS Device: false
```

Figure 1. Thin Provisioning Status

When running the supported versions of HYPERMAX OS/PowerMaxOS and vSphere, VAAI functionality is enabled by default on any PowerMax array.

This paper will discuss the operation of and best use cases for utilizing VAAI features with the PowerMax. In doing so, it will demonstrate how to maximize efficiency and increase the effectiveness of a VMware vSphere environment deployed on a PowerMax.

This paper covers through the PowerMaxOS 10.2.0.0 release.

1 VAAI primitives

Storage APIs for Array Integration is an API for storage partners to leverage that permits certain functions to be delegated to the storage array, thus greatly enhancing the performance of those functions. This API is fully supported on the PowerMax. This array offload capability supports four primitives: hardware-accelerated Full Copy, hardware-accelerated Block Zero, hardware-assisted locking, and UNMAP.

Note: The primitives will not cause failures in the vSphere environment if conditions prevent their use. In such cases, VMware simply reverts back to the default software behavior. For example, if the Full Copy primitive is enabled and a user attempts to clone a VM to a local datastore which does not support hardware acceleration, VMware will revert to the default software copy. The primitives may also be enabled or disabled even in the middle of operations that are utilizing them. At the point of change, VMware will immediately begin using the opposite capability of what was previously being used – software to offload or offload to software – and finish the task with that methodology (assuming no further change is made).

1.1 Hardware-accelerated Full Copy or XCOPY

The time it takes to deploy or migrate a virtual machine is reduced by use of the Full Copy primitive, as the process is entirely executed on the storage array and not on the ESXi server. This minimizes overall traffic on the ESXi server. In addition to deploying new virtual machines from a template or through cloning, Full Copy is also utilized when doing a Storage vMotion. When a virtual machine is migrated between datastores on the same array the hot copy is performed entirely on the array.

Full Copy is implemented as an asynchronous process on the PowerMax. When using the PowerMax, the host simply passes information to the array about what data to copy or move and to what location it should be moved or copied. The progress of this transfer of information is what is seen in the vSphere Client. When the vCenter reports that the clone or Storage vMotion task is complete, it means the PowerMax has received all the information it needs to perform the task. The array, using the same mechanism as the Dell TimeFinder local replication software, completes the transfer of data either synchronously or asynchronously, depending on the state of the device. The end-user is completely unaware of this since the virtual machine is available during the Storage vMotion, or in the case of a clone immediately after the vCenter task is complete.

To ensure that the Full Copy process does not over-utilize array resources, thereby impacting critical operations, there is an upper ceiling that can be reached if many clones and/or Storage vMotions are performed at once; however, reaching these limits is very unlikely. If it does occur, vSphere seamlessly switches to software copy in mid-operation. VMware vSphere will retry the XCOPY command after a brief time, at which point if resources are available, the array copy will resume. Note that a clone or Storage vMotion can switch back and forth between software and hardware copies without any user intervention, and in fact without the end-user knowing. If the user wishes to monitor such activity, the section Monitoring VAAI operations with ESXTOP in this paper provides detail.

Not only does Full Copy save time, but it also saves server CPU cycles, memory, IP and SAN network bandwidth, and storage front-end controller I/O. This is due to the fact that the host is relieved of the normal function it would serve of having to read the data from the array and then write the data back down to the array. That activity requires CPU cycles and memory as well as significant network bandwidth. [Figure 2](#) provides a graphical representation of Full Copy.

Note: The terms Full Copy and XCOPY will be used interchangeably in this paper.

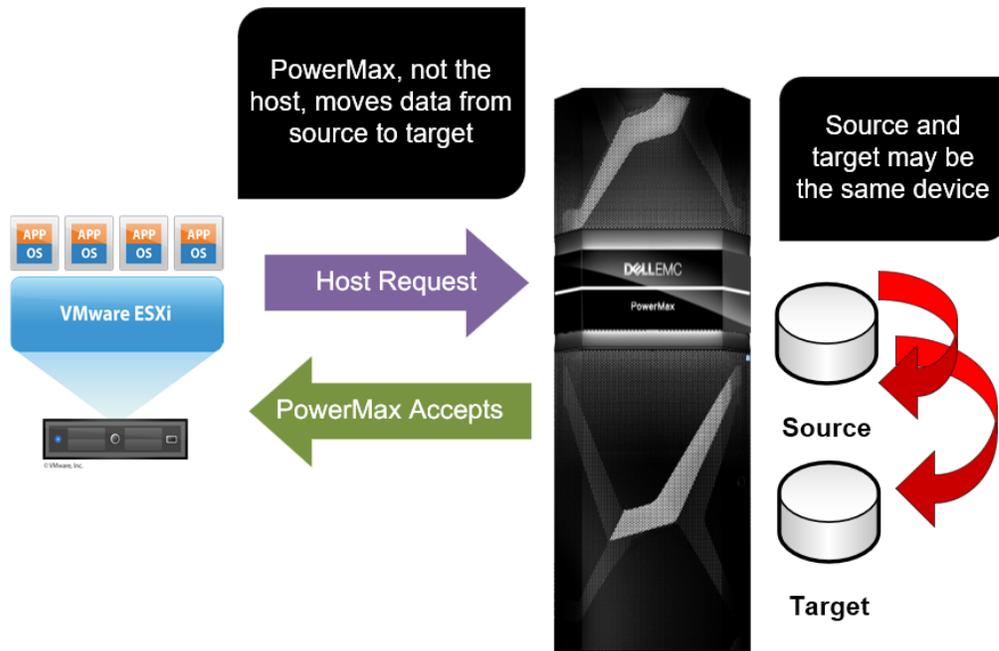


Figure 2. Hardware-accelerated Full Copy

1.2 Hardware-accelerated Block Zero

Using software to zero out a virtual disk is a slow and inefficient process. By passing off this task to the array, the typical back and forth from host to array is avoided and the full capability of the PowerMax can be utilized to accomplish the zeroing.

A typical use case for Block Zeroing is when creating virtual disks that are eagerzeroedthick (EZT) in format. Without the Block Zeroing primitive, the ESXi server must complete all the zero writes of the entire disk before it reports back that the disk zeroing is complete. For large disks this is time-consuming. When employing the Block Zeroing primitive as in [Figure 3](#), (also referred to as “write same” since it uses the WRITE SAME SCSI operation (0x93) to achieve its task) the disk array returns the cursor to the requesting service as though the process of writing the zeros has been completed. It then finishes the job of zeroing out those blocks asynchronously, without the need to hold the cursor until the job is done, as is the case with software zeroing.

On the PowerMax no zeros are actually written, or space reserved. This is because the PowerMax can be thought of as a single pool of storage, since new extents can be allocated from any thin pool on the array. Therefore, the only way that a thin device can run out of space is if the entire SRP runs out of space. It was deemed unnecessary, therefore, to reserve space on the PowerMax when using Block Zero. The WRITE SAME commands themselves, however, are processed regardless. Note, however, that because of the architecture of the PowerMax there is no performance impact even though the space is not pre-allocated.

Block Zero is also initiated with “zeroedthick” or “thin” virtual disks when a guest operating system writes to a previously unallocated portion of the disk. When a new write comes through, ESXi issues write same to the block(s) and then executes the write. The Block Zero operation is only performed once per block upon a first write, subsequent writes to the same block are not preceded by a Block Zero operation. Block Zero is also leveraged during an initial format of a VMFS volume to reserve space for VMFS metadata. Note that only a small portion of the device when formatted as a VMFS volume is actually reserved.

This primitive has a profound effect when deploying "eagerzeroedthick" virtual disks or writing to unallocated sectors of a "zeroedthick" or "thin" virtual disk. When enabled, the PowerMax will allocate the relevant tracks in the thin pool (if not already allocated) but flag the tracks as never written. Block Zero operations therefore result in allocated, but not written to, tracks. Tracks such as these will always return zeros to a host read and will be skipped in the case of a TimeFinder or SRDF.

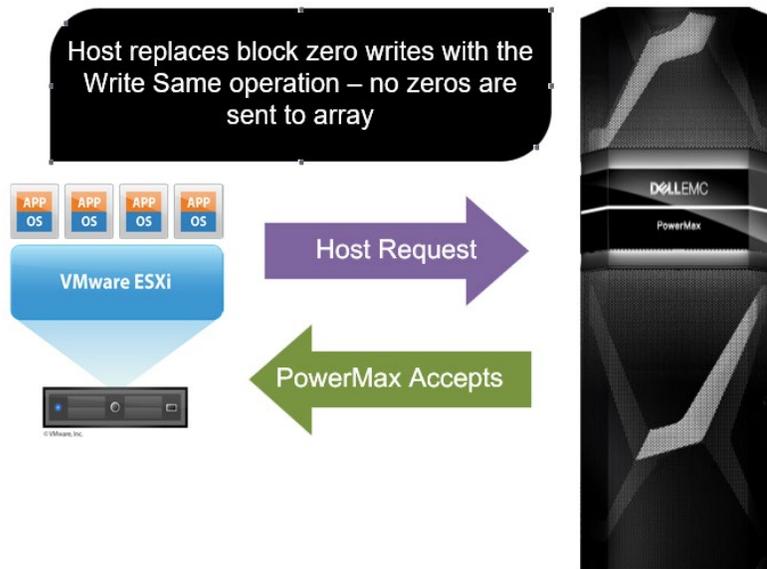


Figure 3. Hardware-accelerated Block Zero

Without Block Zero, VMware will send zeros and the PowerMax will allocate them. If the storage group where the datastore resides has data reduction enabled, however, the PowerMax will remove the zeros and deallocate the tracks when it is destaged from cache. The only condition where this does not hold true, is if the device is written to Storage Class Memory (SCM) as the PowerMax will not use data reduction on that media. In such circumstances, or if data reduction is not in use, the storage can be deallocated by using the volume reclaim functionality in Unisphere for PowerMax, the steps of which are shown in [Figure 4](#).

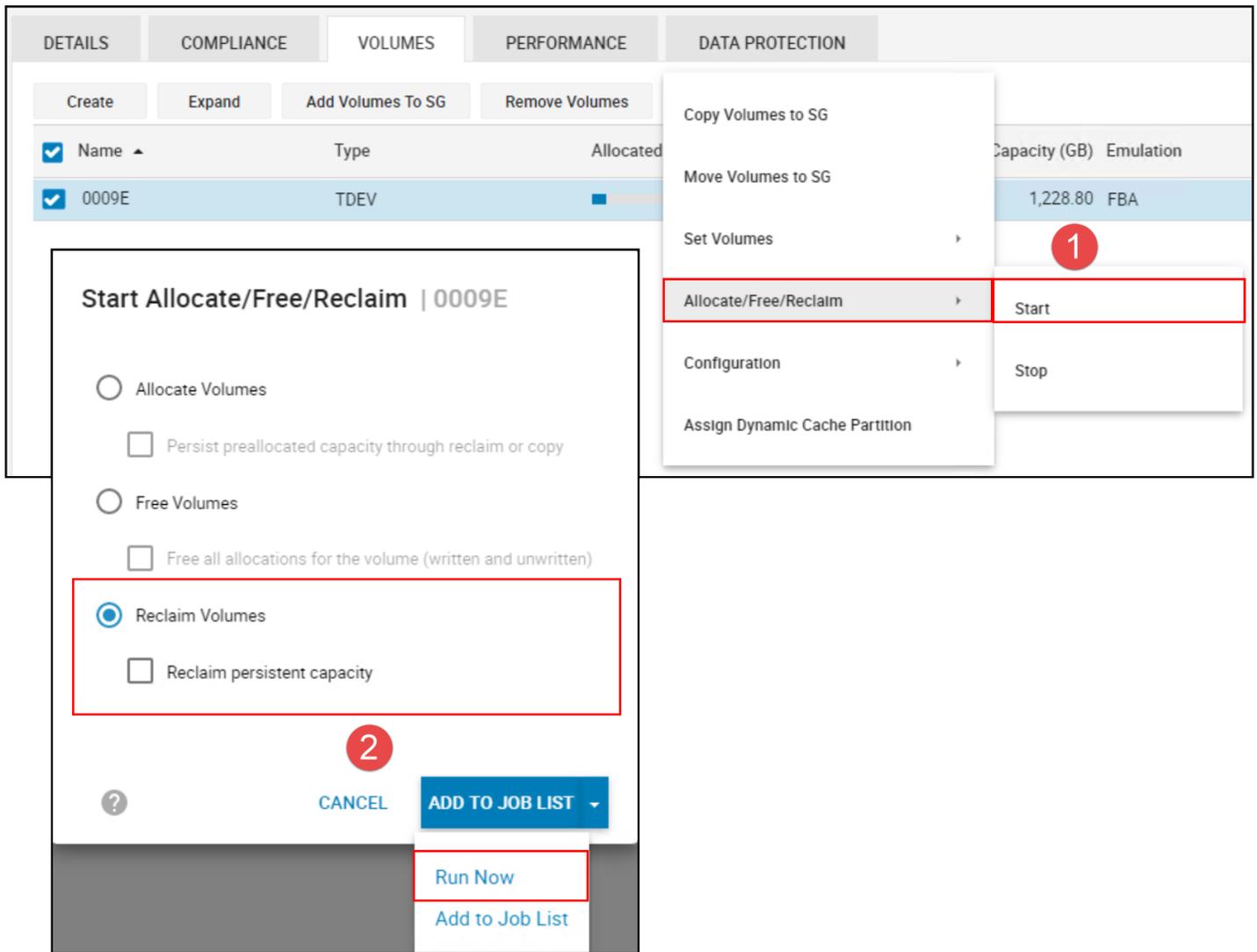


Figure 4. Reclaim zeroed out space

1.3 Hardware-assisted locking

As a clustered shared-storage manager, VMware's Virtual Machine File System (VMFS) needs to coordinate the access of multiple ESXi server hosts to parts of that shared storage. VMFS allocates part of the storage available to it for data describing virtual machines and their configurations, as well as the virtual disks that they access. Within a cluster of ESXi servers the virtual machines contained in the VMFS datastore can be loaded and run on any of the ESXi instances. They can also be moved between instances for load balancing and high availability.

VMware has implemented locking structures within the VMFS datastores that are used to prevent any virtual machine from being run on, or modified by, more than one ESXi host at a time. This locking metadata update operation is used by VMware whenever a virtual machine's state is being changed. This may be a result of the virtual machine being powered on or off, having its configuration modified, or being migrated from one ESXi server host to another with vMotion or Dynamic Resource Scheduling. The initial implementation of mutual exclusion for updates to these locking structures was built using SCSI RESERVE and RELEASE commands. This protocol claims sole access to an entire logical unit for the reserving host until it issues a release. Under the protection of a SCSI RESERVE, a server node could update metadata records on the device

without the risk of interference from any other host attempting to update the same records. This approach, which is shown in Figure 5, has significant impact on overall cluster performance since access to the device by other hosts is prevented while SCSI RESERVE is in effect. As ESXi clusters have grown in size, as well as in their frequency of modifications to the virtual machines they are running, the performance degradation from the use of SCSI RESERVE and RELEASE commands has become unacceptable.

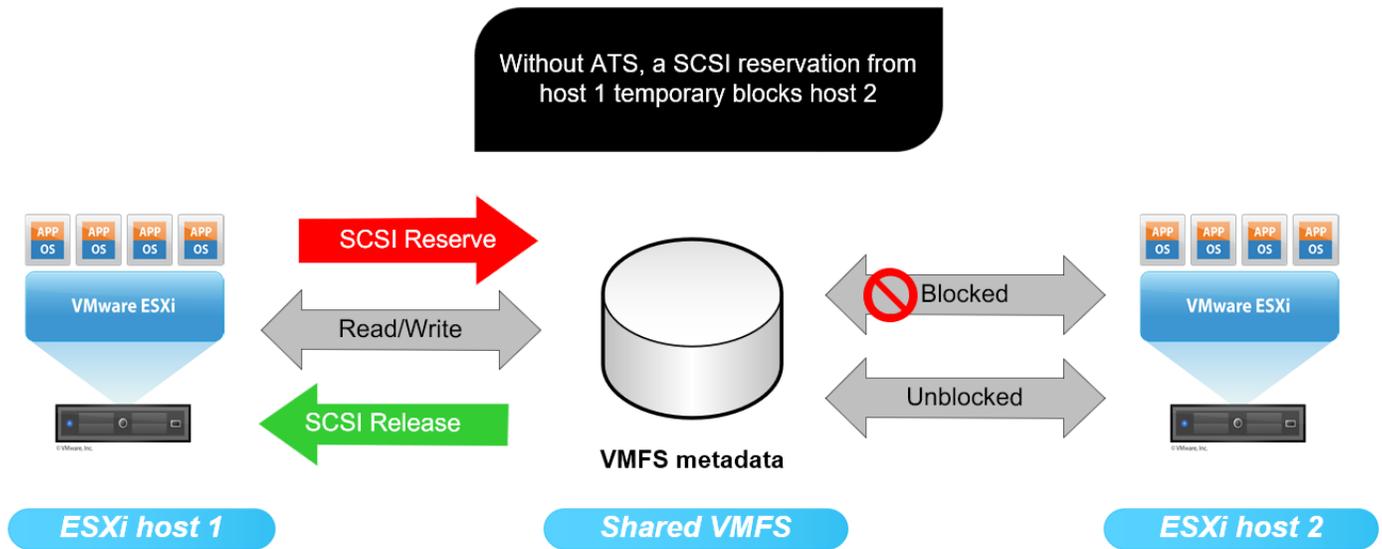


Figure 5. Traditional VMFS locking before hardware-assisted locking (ATS)

This led to the development of the third primitive for VAAI, hardware-assisted locking or ATS. This primitive provides a more granular means of protecting the VMFS metadata than SCSI reservations. Hardware-assisted locking leverages a storage array atomic test and set capability to enable a fine-grained block-level locking mechanism as shown in Figure 6. Firstly, hardware-assisted locking replaces the sequence of RESERVE, READ, WRITE, and RELEASE SCSI commands. It does this with a single SCSI request for an atomic read-modify-write operation conditional on the presumed availability of the target lock. Secondly, this new request only requires exclusivity on the targeted locked block, rather than on the entire VMFS volume containing the lock.

With ATS, access is based on a single block. If a block has been changed by host 1 since the last read from host 2, a re-read is enforced before a write is accepted from host 2

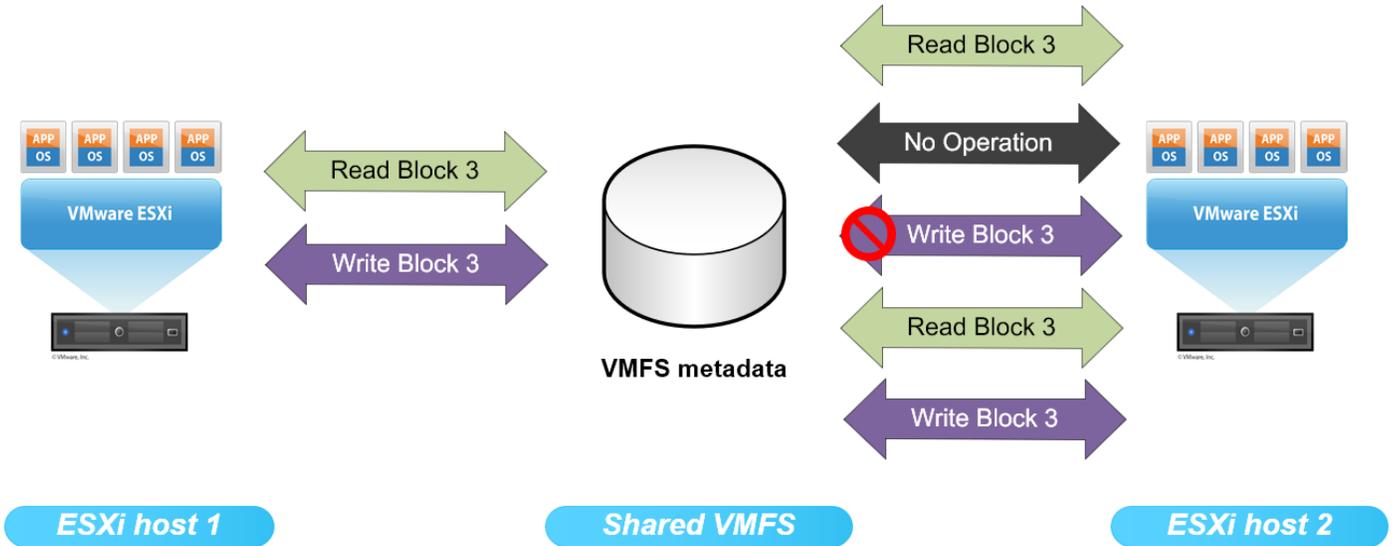


Figure 6. VMFS locking with hardware-assisted locking (ATS)

Although the non-hardware assisted SCSI reservation locking mechanism does not often result in performance degradation, the use of hardware-assisted locking provides a much more efficient means to avoid retries for getting a lock when many ESXi servers are sharing the same datastore. Hardware-assisted locking enables the offloading of the lock mechanism to the array and then the array does the locking at a very granular level. This permits significant scalability in a VMware cluster sharing a datastore without compromising the integrity of the VMFS shared storage-pool metadata.

1.3.1 ATS heartbeat

One of the types of locks that VMware performs is called the heartbeat. The heartbeat is used to synchronize operations between multiple hosts as well as check on the vitality of the host. In the initial implementation, VMware used SCSI reads and writes for the heartbeat, even on storage arrays that supported VAAI (ATS). Soon after, however, VMware defaulted to using ATS on supported arrays, offloading the heartbeat management. The change in behavior means there is a significant increase in the ATS traffic to the storage array. Under certain load conditions, therefore, ATS may fail with a false ATS miscompare or similar ATS error.³ Upon failure ESXi will then re-verify its access to the datastore which will lead to error messages in the vSphere Client similar to: **Lost access to datastore**. An ATS miscompare message can also be found in the /var/log/vmkernel.log file, though it is not the only ATS failure that may be recorded.

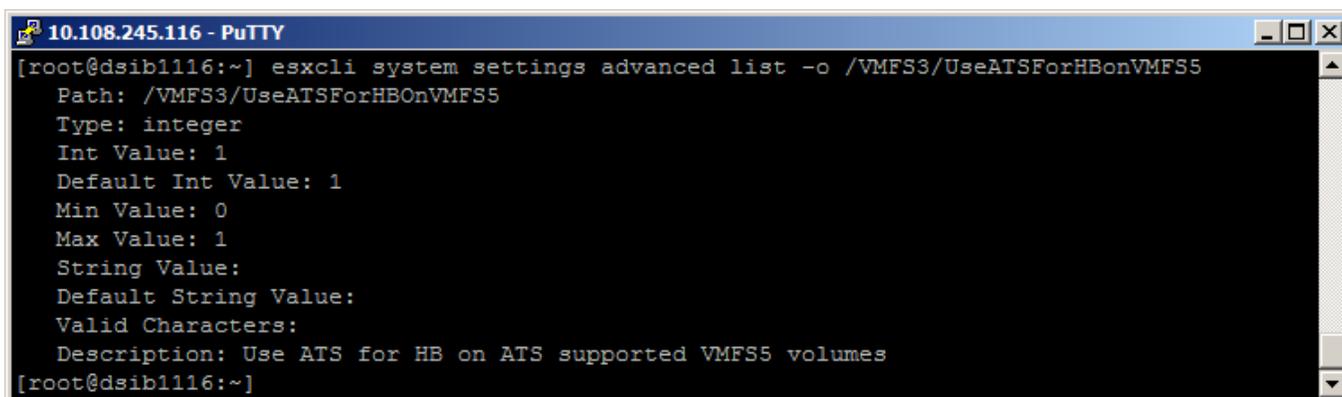
³ VMware KB article 2113956. Although the KB refers to an IBM storage system, the error can occur on any storage platform.

1.3.1.1 Disabling ATS heartbeat

In the unlikely event that ATS errors are persistently occurring in the environment, VMware and Dell recommend disabling ATS for the heartbeat mechanism. Disabling the function means ESXi will revert to the non-ATS functionality, or SCSI reads and writes. VMware indicates that the performance implications of reverting to previous functionality are not significant, thus, it is safe to do so.

Like almost all VAAI functions, disabling the ATS heartbeat is executed online. To view the current setting (Int Value) for the ATS heartbeat, execute the command in [Figure 7](#):

```
esxcli system settings advanced list -o /VMFS3/UseATSForHBonVMFS5
```



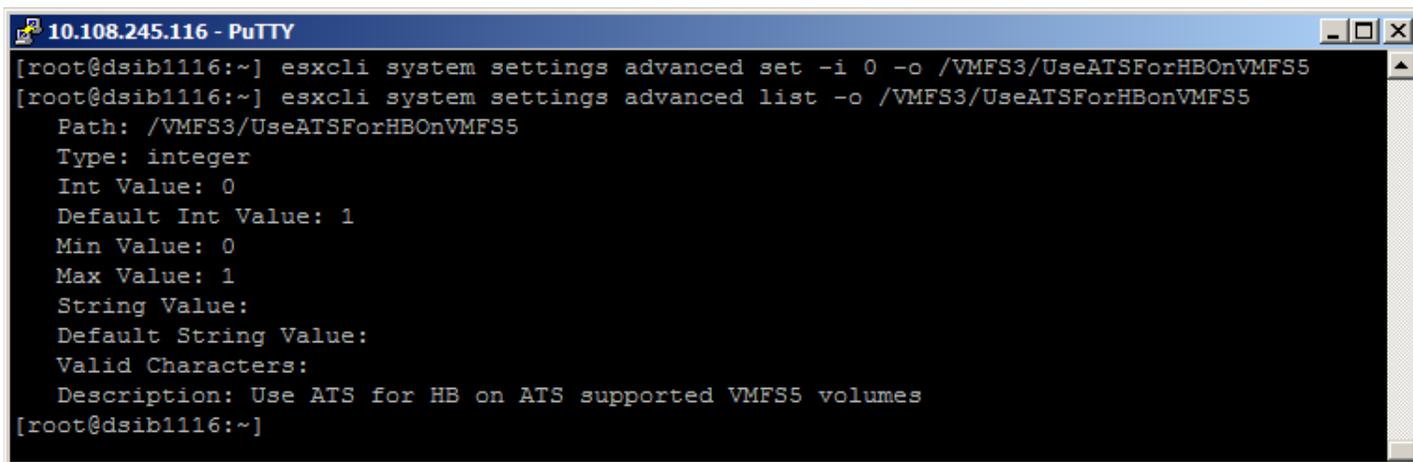
```
10.108.245.116 - PuTTY
[root@dsib1116:~] esxcli system settings advanced list -o /VMFS3/UseATSForHBonVMFS5
  Path: /VMFS3/UseATSForHBonVMFS5
  Type: integer
  Int Value: 1
  Default Int Value: 1
  Min Value: 0
  Max Value: 1
  String Value:
  Default String Value:
  Valid Characters:
  Description: Use ATS for HB on ATS supported VMFS5 volumes
[root@dsib1116:~]
```

Figure 7. ATS heartbeat setting

As noted in [Figure 7](#), the default value is on (1). In order to disable it, run the following command on each host that accesses the datastore in question. Note it will not return a response:

```
esxcli system settings advanced set -i 0 -o /VMFS3/UseATSForHBonVMFS5
```

The execution and result are shown in [Figure 8](#).



```
10.108.245.116 - PuTTY
[root@dsib1116:~] esxcli system settings advanced set -i 0 -o /VMFS3/UseATSForHBonVMFS5
[root@dsib1116:~] esxcli system settings advanced list -o /VMFS3/UseATSForHBonVMFS5
  Path: /VMFS3/UseATSForHBonVMFS5
  Type: integer
  Int Value: 0
  Default Int Value: 1
  Min Value: 0
  Max Value: 1
  String Value:
  Default String Value:
  Valid Characters:
  Description: Use ATS for HB on ATS supported VMFS5 volumes
[root@dsib1116:~]
```

Figure 8. Disabling ATS heartbeat setting

Note: Disabling the ATS heartbeat does not impact any other locking functions that are also handled by the VAAI ATS primitive. Its sole impact is on the creation or updating of the VMFS heartbeat. The ATS heartbeat is a subset of functionality of ATS and that the ATS primitive (*VMFS3.HardwareAcceleratedLocking*) itself should never be disabled.

1.4 UNMAP

The purpose behind UNMAP is the reclamation of space in the thin pool on the array. UNMAP can be utilized in two different scenarios. The first is to free space in a VMFS datastore – manually or automatically.⁴ The second is to free space in a thin vmdk that is part of a Guest OS in a VM. This is known as Guest OS, or in-guest UNMAP. It supports both VMFS and vVol datastores.

1.4.1 Datastore

As virtual machines are created and grow, then are moved, or deleted, space is stranded in the thin pool because the array does not know that the blocks where the data once resided is no longer needed. Prior to the integration of this new Storage API in vSphere and in the PowerMax, reclamation of space required zeroing out the unused data in the pool and then running a zero reclaim process on the array. With UNMAP support, vSphere tells the PowerMax what blocks can be unmapped or reclaimed. If those blocks span a complete extent or extents, they are deallocated and returned to the thin pool for reuse. If the range covers only some tracks in an extent, those tracks are marked as not written on the PowerMax, but the extent cannot be deallocated. This is still beneficial, however, as those tracks do not have to be retrieved from disk should a read request be performed. Instead, the PowerMax array sees the track as not written and immediately returns all zeros.

VMware will, of course, reuse existing space in the datastore even if the data has not been reclaimed on the array, however, testing has shown that VMware will use new space in the datastore before reusing existing. For example, a user creates a new VM in a 1 TB datastore which allocates 400 GB of actual space on the array. This datastore is not using automated UNMAP. The user then deletes the VM and creates a new one of the same size. The space allocation on the array now shows 800 GB because VMware used new space before old. Once all the new space is exhausted, VMware will reuse the old space occupied by that original 400 GB VM, but now the entire device is allocated when only a portion is needed. Automated UNMAP was designed just for this problem. In addition, by reclaiming the space in the thin pool on the PowerMax it becomes available to any device in the pool, not just the one backing the datastore. The use of the SCSI command UNMAP, whether issued manually or automatically, is the only way to guarantee that the full amount of space is reclaimed for the datastore.

Note: Automatic UNMAP does not necessarily remove all unused extents in a datastore for a variety of reasons. This is expected. If it is critical all possible unmappable extents are removed, no matter how small, an occasional manual UNMAP is recommended.

⁴ UNMAP in this scenario does not support vVol datastores as a vVol datastore is a logical construct and is comprised of many devices, not a single device like a VMFS datastore. It is not needed, however, as when an individual vmdk is removed, the vVol on the array is also deleted, thus freeing the space.

1.4.2 Guest OS

Guest OS or in-guest UNMAP is supported in both Windows and Linux. The following prerequisites are required to enable Guest OS UNMAP:

- Thin virtual disk (vmdk) – thick or eagerzero are not supported
- VM hardware version 13+
- If using Linux, the OS and file system must support the UNMAP command
- The advanced parameter **EnableBlockDelete** must be set to 1 if using VMFS 5 (vVols do not require the parameter). It is off by default.

```
esxcli system settings advanced set --int-value 1 --option  
/VMFS3/EnableBlockDelete
```

While Windows automatically issues UNMAP, Linux does not do so by default. Though `sg_unmap` and `fstrim` can be issued manually to free space, a simple parameter is available on file systems to reclaim space automatically. When mounting the file system that supports UNMAP (e.g., `ext4`), use the `discard` option and all deleted files will cause UNMAP to be issued. Here is an example.

```
mount /dev/sdb1 /u02 -o discard
```

1.5 VAAI and HYPERMAX OS/PowerMaxOS version support

Dell's implementation of VAAI is done within the HYPERMAX OS and PowerMaxOS software. As with any software, bugs are an inevitable byproduct. While Dell makes every effort to test each use case for VAAI, it is impossible to mimic every customer environment and every type of use of the VAAI primitives. To limit any detrimental impact customers might experience with VAAI, the following table, Table 1, is provided. It includes all current releases and any E Packs that must be applied at that level, beginning with the recommended base level of 5977 to the current PowerMaxOS 10.2.0.0 release. Using this table will ensure that a customer's array software release is at the proper level for VAAI interaction. It is essential to review both the Important Notes column and any footnotes.

Table 1. HYPERMAX OS/PowerMaxOS and VAAI support

| HYPERMAX OS/PowerMaxOS code level | E Pack required | VAAI primitives supported | Minimum version of vSphere required | Important Notes |
|---|-----------------|-----------------------------------|--|--|
| PowerMaxOS 10.2.0.0 (6079) | No | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 for FC-NVMe, U3 for TCP-NVMe | NVMeoF does not support the Full Copy (XCOPY) primitive. |
| PowerMaxOS 10.1.0.0 and 10.0.1.0 (6079) | No | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 for FC-NVMe, U3 for TCP-NVMe | NVMeoF does not support the Full Copy (XCOPY) primitive. |
| 5978.711.711 (5978 Q1 2021) | No | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 for FC-NVMe, U3 for TCP-NVMe | NVMeoF does not support Full Copy (XCOPY) or Block Zero (WRITE SAME) primitives. |
| 5978.669.669 | No | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 for FC-NVMe, U3 for TCP-NVMe | NVMeoF does not support Full Copy (XCOPY) or Block Zero (WRITE SAME) primitives. |
| 5978.479.479 | No | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 | |
| 5978.444.444 | No | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 | First release to support VAAI metrics. |
| 5978.221.221 | No | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 | |
| 5978.144.144 | No | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 | This is the first PowerMaxOS release for the new PowerMax platform. |
| 5977.1131.1131 | No | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 | Recommended SR on top of 5977.1125.1125. |
| 5977.1125.1125 (5977 2017 Q2 Release) | Recommended | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 | A number of E Packs are available on top of the base 5977.1125.1125 release. |
| 5977.952.892 (5977 2016 Q4 SR) | No | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 | All primitives supported with SRDF/Metro. Support for synchronous Full Copy on SRDF/Metro devices. |
| 5977.945.890 (5977 2016 Q3 SR) | No | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 | All primitives supported with SRDF/Metro. Support for synchronous Full Copy on SRDF/Metro devices. |

| | | | | |
|-----------------------------------|---------------------|-----------------------------------|-------------|---|
| | | | | First VMAX All Flash release with compression available. |
| 5977.814.786 | No | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 | All primitives supported with SRDF/Metro except Full Copy. |
| 5977.813.785 | No | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 | All primitives supported with SRDF/Metro except Full Copy. |
| 5977.811.784 (5977 2016 Q1 SR) | Recommended | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 | All primitives supported with SRDF/Metro except Full Copy. Refer to KB https://support.emc.com/kb/470672 for recommended HYPERMAX OS version. |
| 5977.691.684 (5977 2015 Q3 SR) | Yes for SRDF/ Metro | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 | Support for synchronous Full Copy on SRDF target devices. Only ATS supported on SRDF/Metro. |
| 5977.596.583 (5977 2015 Q1 SR) | No | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 | Recommended minimum code level for VMAX3. Performance improvements in all VAAI implementations. |
| 5977.498.472 (5977 2014 Q4 SR) | No | ATS, Block Zero, Full Copy, UNMAP | vSphere 7.0 | First HYPERMAX OS release with Full Copy support; eNAS supports VAAI for vSphere 5.0 and later with plug-in. |
| 5977.250.189 | No | ATS, Block Zero, UNMAP | vSphere 7.0 | VMAX3 GA; No Full Copy support. |

Note: Be aware that this table only displays information related to using vSphere with VAAI and HYPERMAX OS/PowerMaxOS. It is not making any statements about general vSphere support with HYPERMAX OS/PowerMaxOS. For support information consult E-Lab Navigator. It may also not contain every release available.

1.6 Compression/Deduplication/Data Reduction

Starting with HYPERMAX OS release 5977 2016 Q3 SR, Dell offers compression on VMAX All Flash arrays. Starting with PowerMaxOS 5978 on PowerMax, both compression and deduplication are supported and are referred to as Data Reduction. Compression and deduplication are not available on VMAX3 arrays and deduplication is not available on VMAX All Flash arrays. The VAAI primitives are fully supported with compression and deduplication and no user intervention is required when these features are employed.

1.7 Host IO Limits

Dell provides support for the setting of Host I/O Limits. This feature allows users to place limits on the front-end bandwidth and IOPS consumed by applications on PowerMax systems.

Limits are set at the storage group level. As users build masking views with these storage groups, limits for maximum front-end IOPS or MBs/s are evenly distributed across the directors within the associated masking view. The bandwidth and IOPS are monitored by the array to ensure that they do not exceed the user-specified maximum, throttling when necessary.

VAAI interacts with Host I/O Limits in that the commands issued by VMware to the array count toward either IOPS or MBs/s. From an IOPS perspective, these commands are counted as individual IOs and register as regular metrics based on front-end activity. From a bandwidth perspective, UNMAP and Full Copy commands are counted as individual IOs of modest size since the code does not know the actual extent sizes. But the address ranges that VMware passes with the Block Zero commands count against the Host I/O MBs/s limit. This activity will show up in the Host IO Limit MBs/sec metric, while not showing up in the regular Host MBs/sec metric.

In the following example, a 50 MB/s Host I/O Limit was set on a storage group with a single 500 GB device. A 250 GB eagerzeroedthick disk was created on the device. Figure 9 shows that the Write Same commands are being counted against the limit (yellow line), but the array is not registering it as regular IO (blue line). Note the time it takes to create the vmdk - more than 1 hour. The normal creation time is about 30 seconds. This example of course is exaggerated by using a very small size for the limit yet demonstrates how VAAI and Host I/O Limits interact.

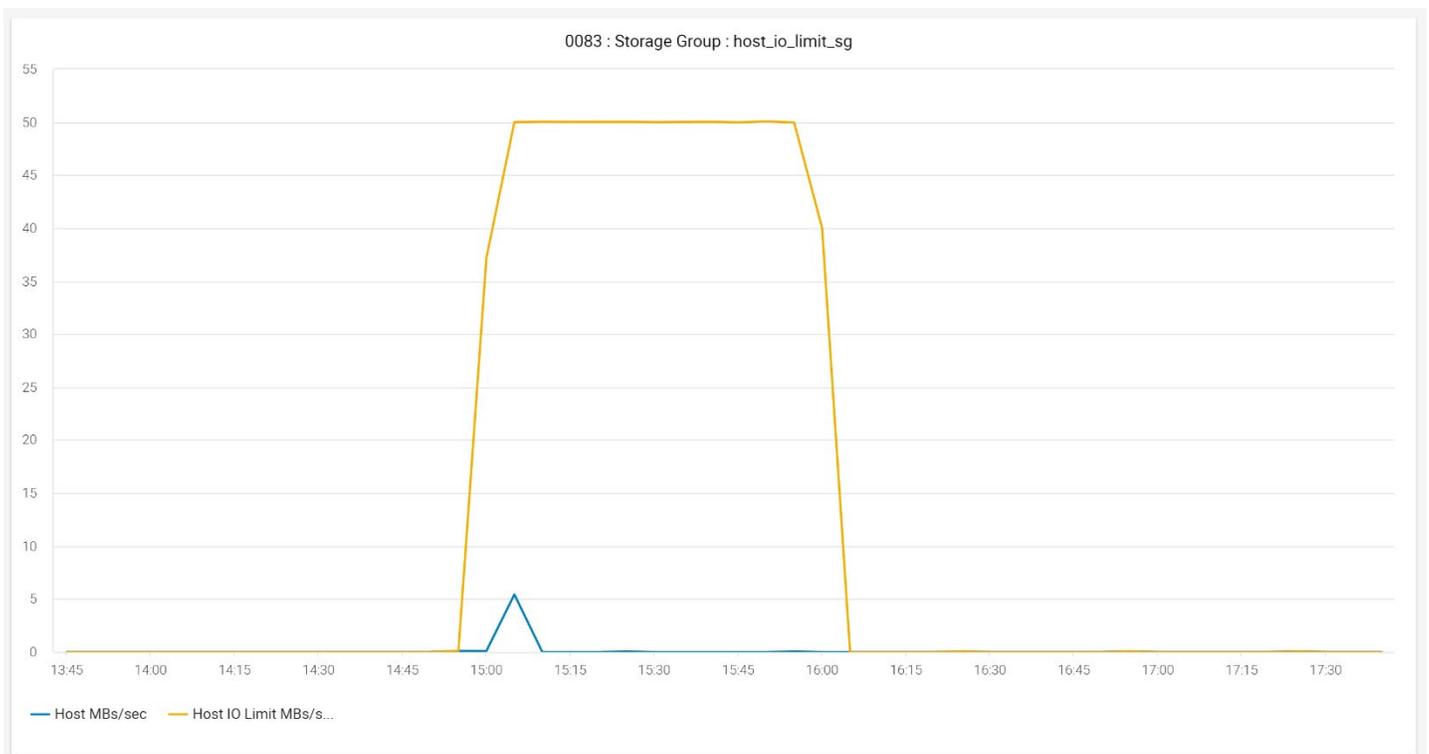


Figure 9. Host IO Limits with VAAI

1.8 Viewing VAAI support

Through the CLI or within the vSphere Client, one can see whether a particular device or datastore supports hardware acceleration or VAAI. From the CLI, using the Network Addressing Authority (NAA) identifier, the status for each VAAI primitive can be seen as in Figure 10 for vSphere 7.x in the red box and vSphere 8.x in the green box. Note in vSphere 8.x there is an additional row for cross-array XCOPY which is not currently supported on PowerMax.

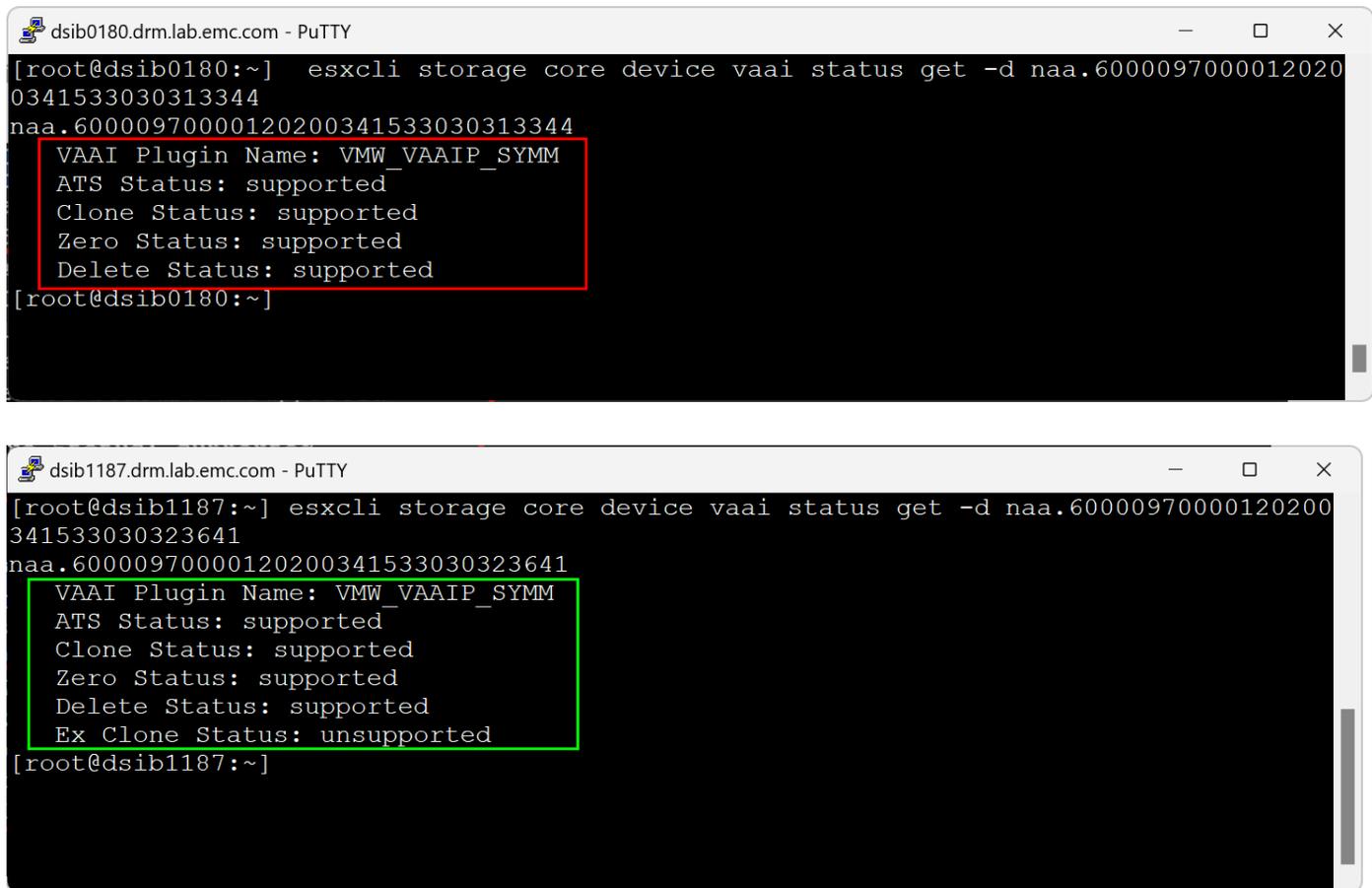


Figure 10. Viewing VAAI primitive support in vSphere 7.x and 8.x

It is important to remember that querying the device provides the status from the array's perspective and not vSphere's. This means that if a primitive is disabled within ESXi, the query will still show the primitives as supported by the array. As it is possible to disable primitives on the array itself, however, the query could return an unsupported value for one of those primitives. [Figure 11](#) shows when both UNMAP and XCOPY are disabled at the array level. Note that only Delete Status is reported as being unsupported and not Clone Status.

```

10.228.244.182 - PuTTY
[root@dsib0182:~] esxcli storage core device vaai status get -d naa.60000970000197600450533030303534
naa.60000970000197600450533030303534
  VAAI Plugin Name: VMW_VAAIP_SYMM
  ATS Status: supported
  Clone Status: supported
  Zero Status: supported
  Delete Status: unsupported
[root@dsib0182:~]
    
```

Figure 11. UNMAP support disabled on the array

If using eNAS or PowerMax File and thus NFS (see section eNAS and VAAI), to view VAAI support run the vmkfstools command shown in Figure 12.

```

dlqa0054.lss.emc.com - PuTTY
~ # vmkfstools -Ph /vmfs/volumes/enas_vmfs1
NFS-1.00 file system spanning 1 partitions.
File system label (if any): enas_vmfs1
Mode: public
Capacity 2.0 TB, 1.9 TB available, file block size 4 KB, max file size 16 TB
UUID: 482273a0-81e18c11-0000-000000000000
Partitions spanned (on "notDCS"):
  nfs:enas_vmfs1
NAS VAAI Supported: YES
Is Native Snapshot Capable: YES
~ #
    
```

Figure 12. Viewing VAAI primitive support for NFS

In the vSphere to view VAAI support, first navigate to the chosen datastore in the left-hand panel. Then select **Configure -> Hardware Acceleration**. The **Hardware Acceleration** column indicates the support status of the primitives for the device or datastore. There are three values for this column: “Supported,” “Not supported,” or “Unknown”⁵. This column is shown in Figure 13.

⁵ A particular device or datastore may be labeled as “Unknown” until it is successfully queried.

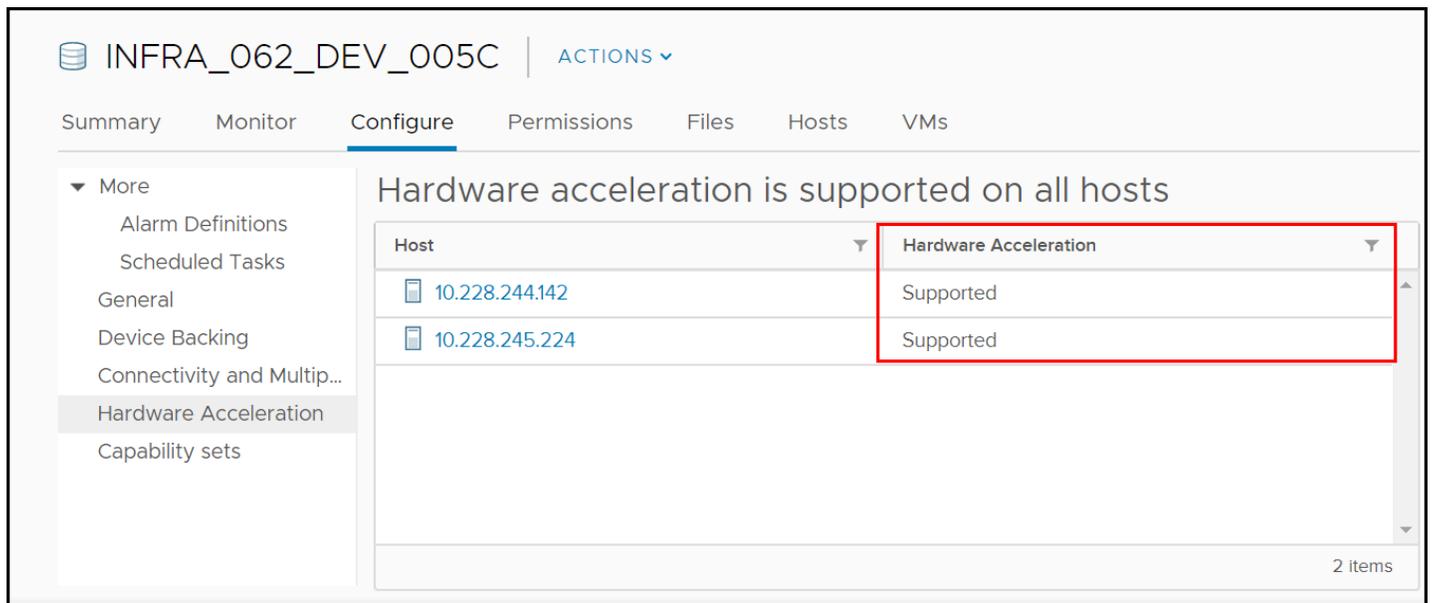


Figure 13. Viewing VAAI support in the vSphere Client

Note that this column, unlike the CLI, does not provide any detail about the individual primitives. For example, if hardware-accelerated locking is supported but not UNMAP, this column will still show “Supported.”

Note: The Hardware Acceleration column does not indicate whether the primitives are enabled or disabled, only whether they are supported.

1.9 VAAI Statistics

Prior to PowerMaxOS 5978.444.444 there is no user interface to view VAAI primitive activity on the array. For Full Copy, however, if the Solutions Enabler command `symdev show` is run against the source or target device involved in a hardware-accelerated operation, the parameter “Extent Based Clone” will either display “Source” or “Target.”

Beginning with PowerMaxOS 5978.444.444, Unisphere for PowerMax includes the following VAAI statistics at the thin device⁶ and storage group level:

- VAAI Total Command Count
- VAAI Total Time
- VAAI Unmap Command Count
- VAAI Unmap KB
- VAAI Unmap MB
- VAAI Write Same Command Count
- VAAI Write Same KB

⁶ These metrics are not available for vVols.

- VAAI Write Same MB
- VAAI XCopy Command Count
- VAAI XCopy KB
- VAAI XCopy MB

In the Performance tab of Unisphere, you can find the VAAI metrics under **All**. This is displayed in [Figure 14](#). The only VAAI metric that is a KPI is **VAAI Total Command Count**.

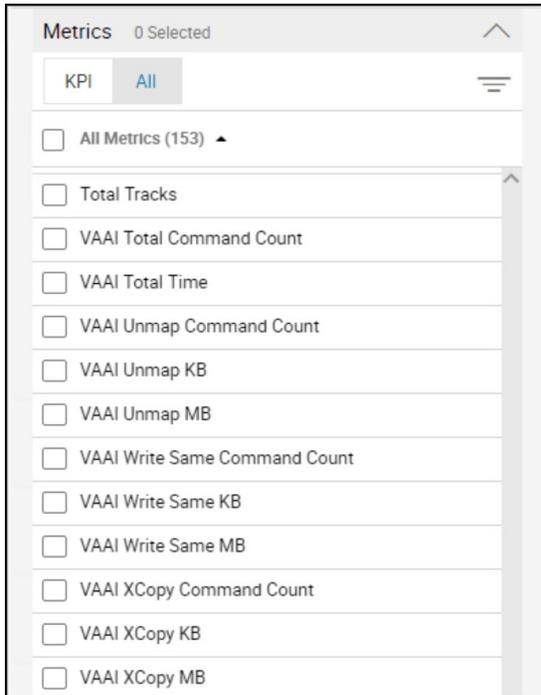


Figure 14. VAAI metrics in Unisphere

Note: More current versions of Unisphere reduced the number of VAAI metrics to eight, removing the MB measurements and renaming the KB measurements to “Throughput.”

The statistics available in Unisphere do not directly correlate with what VMware records for commands in *esxtop* so it is not beneficial to compare them; however, the Unisphere stats provide information not available in *esxtop* such as the amount of storage that was reclaimed during a recent auto UNMAP process.

1.9.1 UNMAP Example

In this example, two VMs were removed from a datastore with auto UNMAP enabled. VMware issued 3024 UNMAP commands, however the array only required ten commands to unmap the space on the array as seen in [Figure 15](#). Those ten commands reclaimed about 215 MB of space.

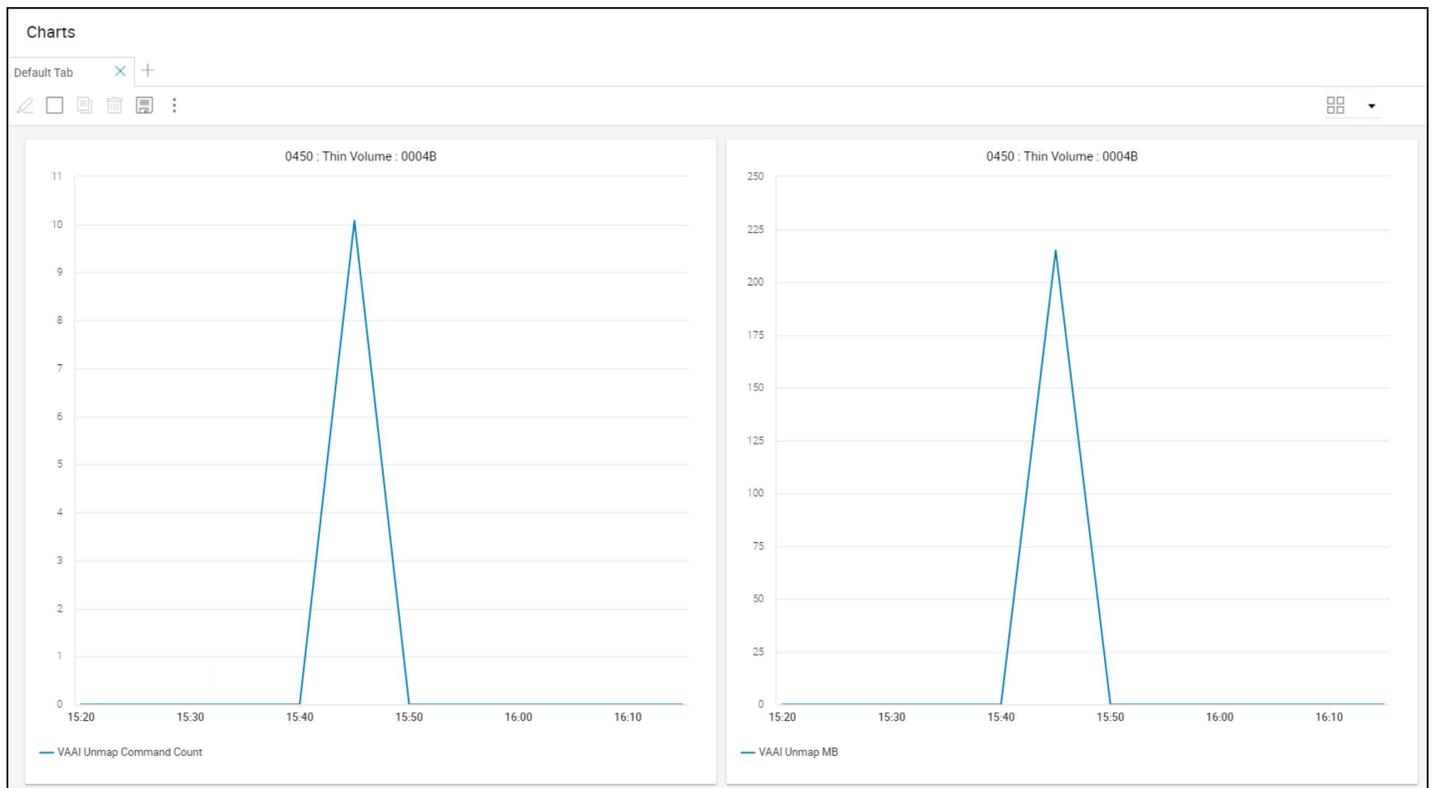


Figure 15. UNMAP statistics example

1.9.2 XCOPY Example

In the following example, a virtual machine was cloned within the same datastore which will engage XCOPY. Two clones were executed, the first using the default transfer size of 4 MB, and the second using the maximum transfer size (without a claim rule) of 16 MB. Note the values in [Figure 16](#).

```

10.228.244.180 - PuTTY
login as: root
Using keyboard-interactive authentication.
Password:
The time and date of this login have been sent to the system logs.

WARNING:
  All commands run on the ESXi shell are logged and may be included in
  support bundles. Do not provide passwords directly on the command line.
  Most tools can prompt for secrets or accept them from standard input.

VMware offers supported, powerful system administration tools. Please
see www.vmware.com/go/sysadmintools for details.

The ESXi Shell can be disabled by an administrative user. See the
vSphere Security documentation for more information.
[root@dsib0180:~] esxcfg-advcfg -g /DataMover/MaxHWTransferSize
Value of MaxHWTransferSize is 4096
[root@dsib0180:~] esxcfg-advcfg -s 16384 /DataMover/MaxHWTransferSize
Value of MaxHWTransferSize is 16384
[root@dsib0180:~]
    
```

Figure 16. XCOPY transfer size

Figure 17 is a screenshot of the results of the two clones. Note the first run in the left of each graph where there were 12 XCOPY commands issued for a total of ~48 MB. When the transfer size was changed to 16 MB and the clone re-run, notice that the command count was reduced to three because the copy size had been quadrupled. The size of the actual copied data, however, does not change. Remember, though, that the max transfer size is just that, a max, and that depending on the data being copied there may not always be perfect ratio between the original copy count and the one done after changing the transfer size.

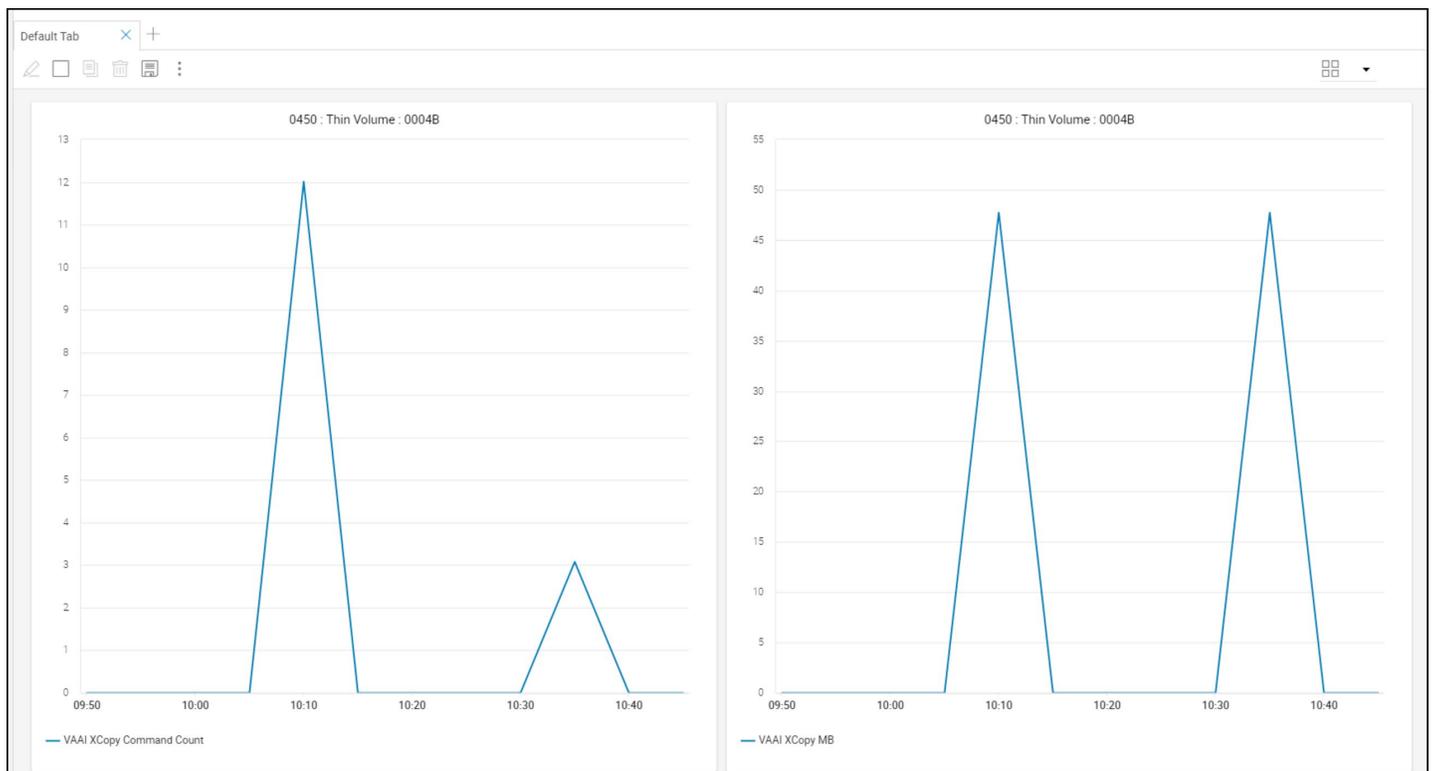


Figure 17. XCOPY statistics example

The new statistics are obviously most useful to tell when a device or devices in a storage group participate in VAAI activity. Knowing this information can help diagnose performance issues or in determining if VAAI commands are being issued.

1.10 Enabling the Storage APIs for Array Integration

Note: This section on enabling and disabling VAAI does not apply to VAAI with NFS. Once the NAS plug-in is installed, VAAI for NFS is enabled and cannot be disabled using any of the methods described in this paper.

1.10.1 Full Copy, Block Zero, and Hardware-assisted locking

The VAAI primitives are enabled by default on both the PowerMax using ESXi servers (properly licensed⁷) and should not require any user intervention. Three of the primitives, however, can be disabled through the ESXi server if desired, either through the CLI or GUI. Using the vSphere Client, Full Copy and Block Zero can be disabled or enabled independently by altering the respective settings, **DataMover.HardwareAcceleratedMove** and **DataMover.HardwareAcceleratedInit**, in the ESXi server advanced settings under DataMover as shown in Figure 18. Under normal circumstances, these settings should not be altered.

⁷ Refer to VMware documentation for required licensing to enable VAAI features.

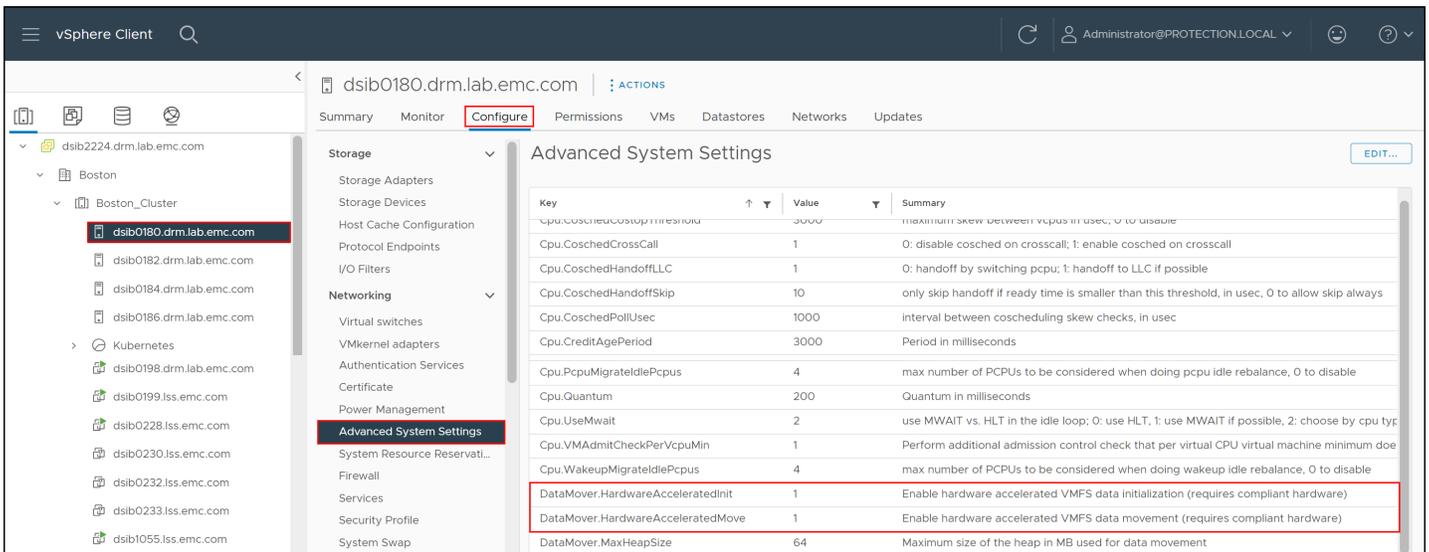


Figure 18. Enabling VAAI in the vSphere Client

Disabling or enabling the primitives is a dynamic process on the ESXi server and does not require the ESXi server to be in maintenance mode, nor does it necessitate a reboot of the server.

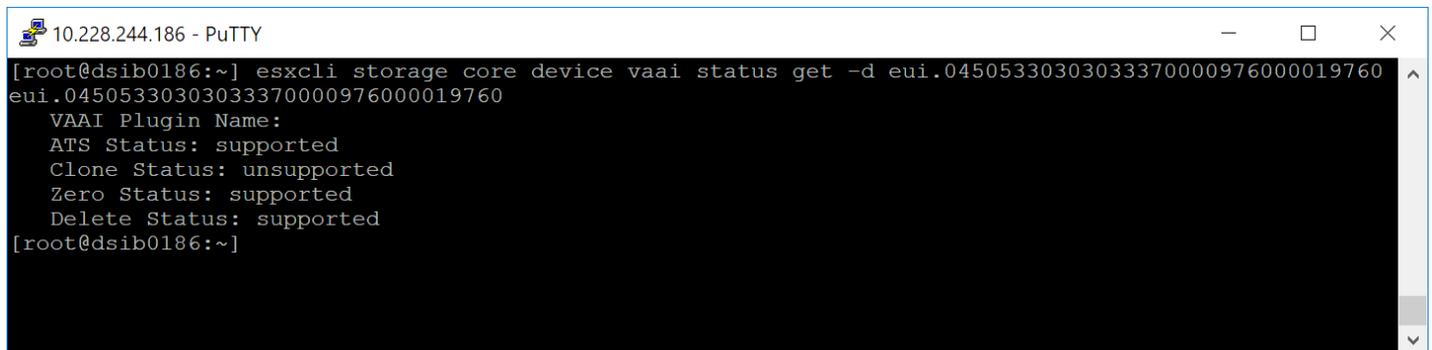
2 NVMe over Fabrics (NVMeoF)

NVMe or non-volatile memory express is a set of standards which define a PCI Express (PCIe) interface used to efficiently access data storage volumes on NVM. The standards were developed by a consortium of companies, including Dell. NVMe enables high concurrency, parallelism, and scalability. It replaces the other SCSI protocols which were never designed to work at the speed of non-volatile media memory. The PowerMax is an all-NVMe box so it benefits from this increase in speed, however, to take full advantage of NVMe the access to it must be extended over the storage network.

NVMe over fabrics, or NVMeoF, defines a common storage architecture for accessing the NVMe block storage protocol over a storage network. This means going from server to SAN, including a front-end interface to NVMe storage. There are any number of fabrics that make up NVMeoF including FC, RoCE, and IP. As FC is the predominant fabric for PowerMax customers, Dell introduced support for FC-NVMe with PowerMaxOS release 5978.444.444. VMware vSphere 7 supports FC-NVMe. In vSphere 7.0U3 VMware released support for TCP which Dell supports with PowerMaxOS 10.

2.1 VAAI

The future of NVMeoF is the replacement of all SCSI commands, but for now, VMware's implementation of FC-NVMe and NVMe/TCP is not a complete conversion to the new command set. There is still some command-set translation required between the new protocol and traditional SCSI. Unlike traditional SCSI, NVMeoF devices are not claimed by the VAAI Plug-in (see [Figure 19](#) where no name is shown). Instead, the command is translated directly, though the command number itself is different. For example, ATS Compare and Write in SCSI is the same in NVMeoF. UNMAP in SCSI becomes deallocate in NVMeoF and WRITE SAME or Block Zero becomes Write Zeroes. XCOPY has not been ported over, so VMware will not even attempt to send commands to the array and will use host-based copy.



```

10.228.244.186 - PuTTY
[root@dsib0186:~] esxcli storage core device vaa1 status get -d eui.045053303033370000976000019760
eui.04505330303033370000976000019760
VAAI Plugin Name:
ATS Status: supported
Clone Status: unsupported
Zero Status: supported
Delete Status: supported
[root@dsib0186:~]

```

Figure 19. VAAI support with NVMeoF

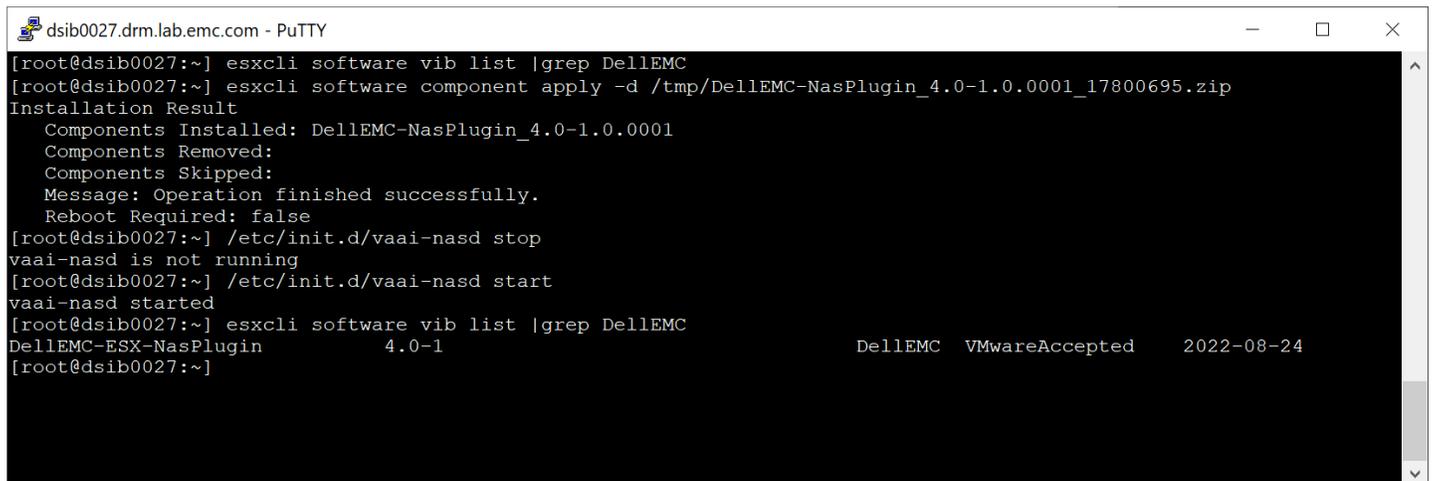
3 eNAS and PowerMax File

Dell offers Embedded NAS (eNAS) guest OS (GOS) on the PowerMax 2000/8000 and Embedded File GOS on the PowerMax 2500/8500. NAS on the PowerMax enables customers to leverage vital Tier 1 features including SLO-based provisioning, Host I/O Limits, and data reduction technologies. eNAS is comprised of virtual instances of VNX Software Data Movers and Control Stations that run on the PowerMax, while File is a new implementation that is being utilized across different arrays in the portfolio.

Unlike block storage, file storage does not support VAAI as part of the array code. All NFS file systems, regardless of vendor, require a NAS plug-in. Fortunately, the plug-in code is generic so that the same plug-in used for the Dell Unity platform can also be used for other arrays like PowerMax and PowerStore. Note that the plug-in works with either NFS version 3 or 4.1.

The VAAI features for NAS are not exactly equivalent to those available on block, though the idea is the same: to offload tasks to the array. Features on NAS include NFS Clone Offload, extended stats, space reservations, and snap of a snap. The NFS clone offload works much the same way as XCOPY as it offloads ESXi clone operations to the array.

There are two different versions of the plug-in depending on which ESXi version that is running: 4.0.1 for 7.0.1+ and 3.0.2 for 7.0+. Version 4.0.1 does not require a reboot after installation. To install the plug-in, download the NAS plug-in from Dell support. The plug-in is delivered as a VMware Installation Bundle (vib). The plug-in can be installed through VMware vCenter Update Manager or through the CLI as demonstrated in [Figure 20](#). Be sure to check for any existing NAS plug-in as it must be removed before installing the new one. Stop and start the `vaai-nasd` service if installing version 4.0.1 as below, otherwise reboot for the other versions.



```

dsib0027.drm.lab.emc.com - PuTTY
[root@dsib0027:~] esxcli software vib list |grep DellEMC
[root@dsib0027:~] esxcli software component apply -d /tmp/DellEMC-NasPlugin_4.0-1.0.0001_17800695.zip
Installation Result
  Components Installed:  DellEMC-NasPlugin_4.0-1.0.0001
  Components Removed:
  Components Skipped:
  Message: Operation finished successfully.
  Reboot Required: false
[root@dsib0027:~] /etc/init.d/vaai-nasd stop
vaai-nasd is not running
[root@dsib0027:~] /etc/init.d/vaai-nasd start
vaai-nasd started
[root@dsib0027:~] esxcli software vib list |grep DellEMC
DellEMC-ESX-NasPlugin          4.0-1                DellEMC  VMwareAccepted  2022-08-24
[root@dsib0027:~]

```

Figure 20. Installation of the NAS plug-in for VAAI

Once installed, the vSphere Client will show that VAAI is supported on the NFS datastores as in [Figure 21](#). If the plug-in is not installed, the column will show as “Unknown.”

| | Name | ↑ | Status | Type | Capacity | Free | Hardware Acceleration |
|--------------------------|------------------------|---|----------|---------|-------------|-------------|-----------------------|
| <input type="checkbox"/> | dsib0180_local | | ✓ Normal | VMFS 6 | 150.75 GB | 149.34 GB | Unknown |
| <input type="checkbox"/> | NFS_302_DS_1 | | ✓ Normal | NFS 4.1 | 1 TB | 1,022.49 GB | Supported |
| <input type="checkbox"/> | ORACLE_SID_302_DEV_134 | | ✓ Normal | VMFS 6 | 12 TB | 11.81 TB | Supported |
| <input type="checkbox"/> | TCP_DEV_120_SID_302 | | ✓ Normal | VMFS 6 | 1,023.75 GB | 1,022.33 GB | Supported |
| <input type="checkbox"/> | TCP_DEV_138_SID_302 | | ✓ Normal | VMFS 6 | 1,023.75 GB | 1,022.33 GB | Supported |
| <input type="checkbox"/> | VEEAM_SID_302_DEV_109 | | ✓ Normal | VMFS 6 | 8 TB | 7.02 TB | Supported |

Figure 21. NFS datastore with VAAI support

3.1 Extended stats

As mentioned, one of the other VAAI integrations for NFS is extended stats. Using `vmkfstools` the user can display the disk utilization for virtual machine disks configured on NFS datastores. The ***extendedstat*** argument provides disk details for the virtual disks. The command reports, in bytes, virtual disk size, used space, and unshared space. An example is shown in [Figure 22](#).

```

10.228.244.182 - PuTTY
[root@dsib0182:~] vmkfstools --extendedstat /vmfs/volumes/NFS_302_DS_1/dsib1092.lss.emc.com/dsib1092.lss.emc.com.vmdk
Capacity bytes: 42949672960
Used bytes: 42949672960
Unshared bytes: 42949672960
[root@dsib0182:~]
    
```

Figure 22. VAAI NFS extended stats feature

3.2 Nested clones

Another VAAI capability on NFS is the ability to create a thin-clone virtual machine from an existing thin-clone virtual machine. This functionality is referred to as nested clones. The functionality uses the snapshot architecture to instantaneously create lightweight virtual machine clones. The clone operations are off-loaded to the array. To take advantage of this feature with eNAS, enable the nested clone support property when you create the file system as highlighted in [Figure 23](#). It cannot be changed after the file system is created. File enables the capability automatically upon file system creation.

| | |
|--|---|
| Create from | <input checked="" type="radio"/> Storage Pool <input type="radio"/> Meta Volume |
| File System Name: | <input type="text"/> |
| Storage Pool: | symm_dsl 1.8 TB (1880960 MB) ▼ |
| Storage Capacity: | <input type="text"/> GB ▼ |
| Auto Extend Enabled: | <input type="checkbox"/> |
| Thin Enabled: | <input type="checkbox"/> |
| Slice Volumes: | <input type="checkbox"/> |
| Deduplication Enabled: | <input type="checkbox"/> |
| VMware VAAI nested clone support: | <input checked="" type="checkbox"/> (Must be selected at file system creation time) |
| Data Mover (R/W): | server_2 ▼ |
| Mount Point: | <input checked="" type="radio"/> Default <input type="radio"/> Custom |

Figure 23. Setting nested clone support on a file system

4 Full Copy Detail

By default, when the Full Copy primitive is supported, VMware informs the storage array to copy data in 4 MB chunks. For ESXi hosts that utilize more than PowerMax arrays, the default value should not be altered. However, in PowerMax only, or a combination of PowerMax and VNX environments for vSphere 7.x, and 8.x customers are advised to take advantage of VMware's ability to send larger chunks to the array for copy. In PowerMax only environments, there is a more advanced capability that can be used to increase the extent size even higher by the use of claim rules. The following two sections outline how to adjust the default copy size and how to add claim rules.

4.1 Default copy size in vSphere 7 and 8

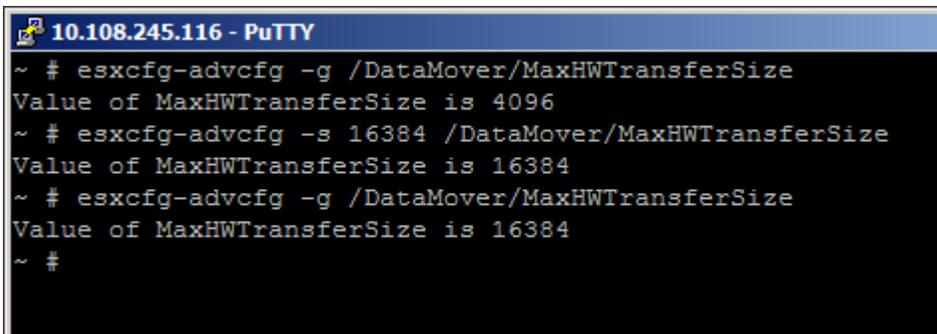
In vSphere 7 and 8 the default copy size can be incremented to a maximum value of 16 MB and should be set at that value for all PowerMax environments to take advantage of the array's ability to move larger chunks of storage at once.

The following commands show how to query the current copy (or transfer) size and then how to alter it.

```
esxcfg-advcfg -g /DataMover/MaxHWTransferSize
Value of MaxHWTransferSize is 4096

esxcfg-advcfg -s 16384 /DataMover/MaxHWTransferSize
Value of MaxHWTransferSize is 16384
```

Note that this change is per ESXi server and can only be done through the CLI (no GUI interface) as in [Figure 24](#):



```
10.108.245.116 - PuTTY
~ # esxcfg-advcfg -g /DataMover/MaxHWTransferSize
Value of MaxHWTransferSize is 4096
~ # esxcfg-advcfg -s 16384 /DataMover/MaxHWTransferSize
Value of MaxHWTransferSize is 16384
~ # esxcfg-advcfg -g /DataMover/MaxHWTransferSize
Value of MaxHWTransferSize is 16384
~ #
```

Figure 24. Changing the default copy size for Full Copy on the ESXi host

Alternatively, vSphere PowerCLI can be used at the vCenter level to change the parameter on all hosts as in [Figure 25](#).

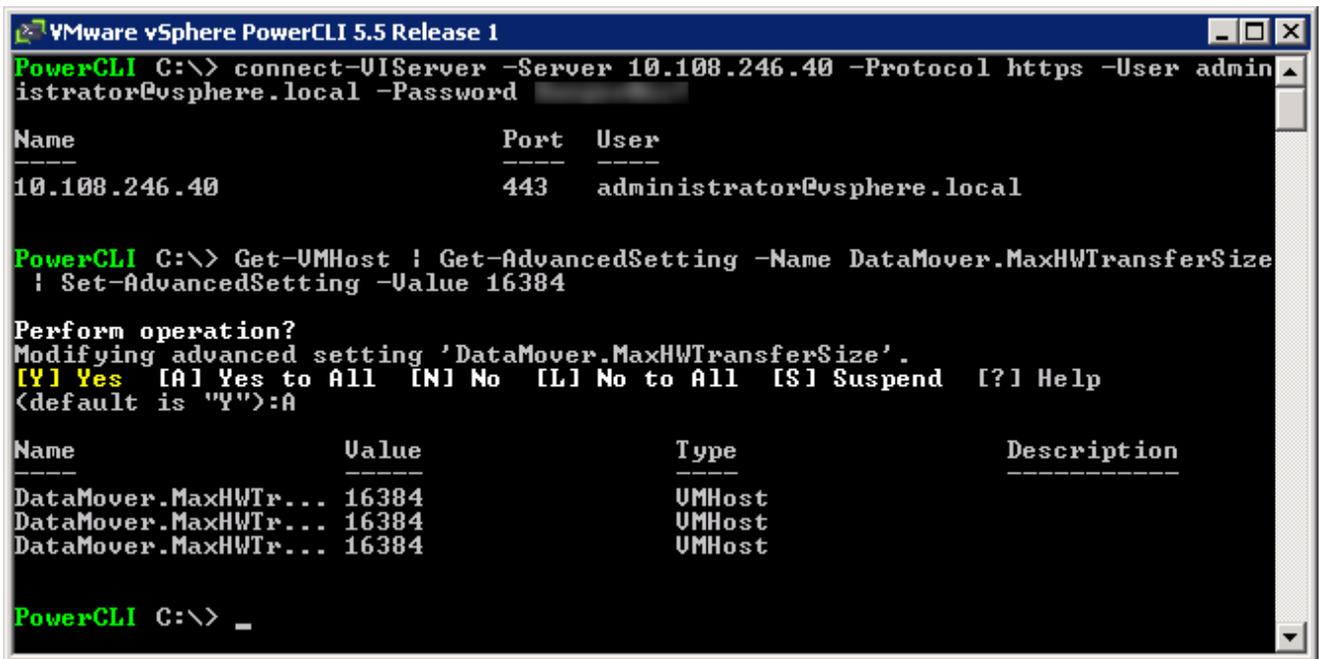


Figure 25. Changing the default copy size for Full Copy using vSphere PowerCLI

It is important to adjust the default transfer size in case claim rules are not in use or a condition occurs which renders them non-functional.

4.2 XCOPY claim rules in vSphere 7 and 8

In addition to increasing the default transfer size, VMware has enabled a different methodology by which the extent size can be increased – claim rules. These claim rules will take precedence over the MaxHWTransferSize.

Claim rules have always been used to implement VAAI for PowerMax. VAAI is a plug-in claim rule that says if a device is from a particular vendor, in this case a Dell PowerMax device, it should go through the VAAI filter. In other words, if VAAI can be used it will be used (various situations exist – detailed in this paper – where primitives like XCOPY will not be used). Each block storage device managed by a VAAI plug-in needs two claim rules, one that specifies the hardware acceleration filter and another that specifies the hardware acceleration plug-in for the device. These claim rules are where the change has been made (in addition to the code). What VMware added three new columns to the storage core claim rules to control the new behavior. They are:

- XCOPY Use Array Reported Values
- XCOPY Use Multiple Segments
- XCOPY Max Transfer Size

Figure 26 shows the output with the new columns and their default values.

```

10.108.245.116 - PuTTY
[root@dsib1116:~] esxcli storage core claimrule list --claimrule-class=VAAI
Rule Class Rule Class Matches
-----
VAAI 65429 Filter VAAIP_MASK vendor=MSFT model=Virtual HD false false 0
VAAI 65429 Filter VAAIP_MASK vendor=MSFT model=Virtual HD false false 0
VAAI 65430 Filter VAAIP_SYMM vendor=EMC model=SYMMETRIX false false 0
VAAI 65430 Filter VAAIP_SYMM vendor=EMC model=SYMMETRIX false false 0
VAAI 65431 Filter VAAIP_CX vendor=DGC model=* false false 0
VAAI 65431 Filter VAAIP_CX vendor=DGC model=* false false 0
VAAI 65432 Filter VAAIP_EQL vendor=EQLOGIC model=* false false 0
VAAI 65433 Filter VAAIP_NEIAPP vendor=NETAPP model=* false false 0
VAAI 65434 Filter VAAIP_HDS vendor=HITACHI model=* false false 0
VAAI 65434 Filter VAAIP_HDS vendor=HITACHI model=* false false 0
VAAI 65435 Filter VAAIP_LHN vendor=LEFTHAND model=* false false 0
VAAI 65435 Filter VAAIP_LHN vendor=LEFTHAND model=* false false 0
[root@dsib1116:~]

```

Figure 26. VAAI claim rule class

By altering the rules and changing these three new settings, one can adjust the max transfer size up to a maximum of 240 MB, rather than 16 MB. Though the columns exist in both the Filter and VAAI claim rule classes, only the VAAI class needs to be changed. The new required settings are:

- XCOPYY Use Array Reported Values = **true**
- XCOPYY Use Multiple Segments = **true**
- XCOPYY Max Transfer Size = **240**

It is not sufficient to simply change the transfer size because VMware requires that the **Use Array Reported Values** is set to true if the transfer size is set. When **Use Array Reported Values** is set it means that VMware will query the array to get the largest transfer size if the max transfer size is not set or if there is a problem with the value. Unfortunately, the PowerMax advertises a value higher than 240 MB so in the case of an issue with the transfer size VMware will default to the previous value set for MaxHWTransferSize; therefore, it is important to set that to 16 MB. In addition, for the PowerMax to accept up to 240 MB, multiple segments must be enabled since a single segment is limited to 30 MB (there are eight segments available). Therefore, when one modifies the claim rules, set the size to 240 MB and set both other columns to true (even though the **Use Array Reported Values** column is not used).

4.2.1 Custom vendor rule

The process to change the claim rules differs slightly between what is termed the **custom** rule versus the **generic** rule. Both, however, involve removing the existing VAAI rule class, loading that change, then adding it back with the new columns set. Following this a reboot of the server is needed.⁸ The custom claim rule plugin was introduced for PowerMax arrays back in vSphere 6.x and is called **VMW_VAAIP_SYMM**. Note that each storage vendor has their own custom plug-in. To change the new column values run the following commands. They will not return a response. Note that failure to load the claimrule before adding back the new one may result in a duplicate rule error.

Note: Refer to the VMware documentation to ensure the proper syntax is used for the claim rule commands

```
esxcli storage core claimrule remove --rule 65430 --claimrule-class=VAAI
```

⁸ There is a process by which this can be done with the server running which involves unmounting datastores and detaching devices; however, it is not recommended as testing has shown it to be inconsistent.

```
esxcli storage core claimrule load --claimrule-class=VAAI
```

```
esxcli storage core claimrule add -r 65430 -t vendor -V EMC -M SYMMETRIX -P
VMW_VAAIP_SYMM -c VAAI -a -s -m 240
```

To verify the changes are set before reboot, list the claim rules as in [Figure 27](#) below:

```
esxcli storage core claimrule list --claimrule-class=VAAI
```

| Rule Class | Rule ID | Claim Rule | XCOPY Use Array | Reported Values | XCOPY Use Multiple Segments | XCOPY Max Transfer Size |
|------------|---------|------------------|-----------------|-----------------|-----------------------------|-------------------------|
| VAAI | 65429 | Model=Virtual HD | false | false | false | 0 |
| VAAI | 65429 | Model=Virtual HD | false | false | false | 0 |
| VAAI | 65430 | Model=SYMMETRIX | true | true | true | 240 |
| VAAI | 65430 | Model=SYMMETRIX | true | true | true | 240 |
| VAAI | 65431 | Model=* | false | false | false | 0 |
| VAAI | 65431 | Model=* | false | false | false | 0 |
| VAAI | 65432 | Model=* | false | false | false | 0 |
| VAAI | 65432 | Model=* | false | false | false | 0 |
| VAAI | 65433 | Model=* | false | false | false | 0 |
| VAAI | 65433 | Model=* | false | false | false | 0 |
| VAAI | 65434 | Model=* | false | false | false | 0 |
| VAAI | 65434 | Model=* | false | false | false | 0 |
| VAAI | 65435 | Model=* | false | false | false | 0 |

Figure 27. Updated VAAI claim rule for XCOPY

The changes will not be permanent until a reboot of the server. Once the reboot is complete, re-run the command to ensure the changes appear. Dell always recommends using the maximum value of 240 MB. Using the maximum ensures that, if possible (the extent must be contiguous), a 240 MB extent will be used.

The new claim rule does not impact how VAAI functionality works. For instance, the following still hold true:

- It will revert to software copy when it cannot use XCOPY
- It can still be enabled/disabled through the GUI or CLI
- It will use the **DataMover/MaxHWTransferSize** setting when the claim rules are not changed or a non-PowerMax array is being used
- The maximum value is just that, a maximum. That is the largest extent VMware can send but does not guarantee all extents will be that size. It is common when XCOPY is activated that extent copy sizes will vary from under 1 MB to 240 MB and everything in between.

To return to the former functionality, simply remove the rule, and add back the claimrule without turning on any settings:

```
esxcli storage core claimrule remove --rule 65430 --claimrule-class=VAAI
```

```
esxcli storage core claimrule load --claimrule-class=VAAI
```

```
esxcli storage core claimrule add -r 65430 -t vendor -V EMC -M SYMMETRIX -P
VMW_VAAIP_SYMM -c VAAI
```

A reboot is again required.

4.2.1.1 Generic rule

Starting in vSphere 6.7, VMware began the process of moving away from the custom vendor VAAI plug-ins (VMW_VAAIP_SYMM) enumerated in the previous section. To that end, VMware offers a generic plug-in for VAAI named VMW_VAAIP_T10. This generic plug-in works the same way the custom plug-in does. The idea is to eventually deprecate all vendor custom plug-ins. Although the custom plug-in is still supported, for new servers which do not already have the claim rules setup for PowerMax, Dell recommends using the new generic plug-in. The steps are similar to the custom plug-in and are covered next.

4.2.1.2 VMW_VAAIP_T10

First, the existing custom vendor rule must be removed. Because VMware suggests using claim rule number 914, remove both the Filter and VAAI classes. This is different than what was done for the custom plug-in as the Filter class did not need to be removed. Note that if an attempt is made to add claim rule 914 before 65430 is removed, VMware will error and indicate 914 is a duplicate of 65430.

Note: The number 914 for the claim rule is arbitrary. If it is currently in use, a different number can be used, for example 950.

```
esxcli storage core claimrule remove --rule 65430 --claimrule-class=Filter
esxcli storage core claimrule remove --rule 65430 --claimrule-class=VAAI
```

Reload the filter and rule before adding the new ones to avoid duplication.

```
esxcli storage core claimrule load --claimrule-class=Filter
esxcli storage core claimrule load --claimrule-class=VAAI
```

Now add the new filter and rule.

```
esxcli storage core claimrule add -r 914 -t vendor -V EMC -M SYMMETRIX -P
VAAI_FILTER -c Filter
esxcli storage core claimrule add -r 914 -t vendor -V EMC -M SYMMETRIX -P
VMW_VAAIP_T10 -c VAAI -a -s -m 240
```

Reboot the ESXi host for the changes to take effect. Once the host comes back up, verify the changes. Be aware that because rule 65430 is still part of the ESXi build, VMware will add it back after the reboot; however, it is only listed as a “file” and not also “runtime” as is the new generic claim rule. Therefore, it cannot claim any devices. The new rule is seen in the red box, while the old rule is in blue in [Figure 28](#).

```
esxcli storage core claimrule list --claimrule-class=VAAI
```

| Rule | Class | Rule | Class | Type | Plugin | Matches | XCOPY Use | Transfer Size KiB |
|------|-------|---------|--------|------------------|------------------------------|---------|-----------|-------------------|
| VAAI | 914 | runtime | vendor | VMW_VAAIP_T10 | vendor=EMC model=SYMMETRIX | | | 245760 |
| VAAI | 914 | file | vendor | VMW_VAAIP_T10 | vendor=EMC model=SYMMETRIX | | | 245760 |
| VAAI | 65429 | runtime | vendor | VMW_VAAIP_MASK | vendor=MSFT model=Virtual HD | | | 0 |
| VAAI | 65429 | file | vendor | VMW_VAAIP_MASK | vendor=MSFT model=Virtual HD | | | 0 |
| VAAI | 65430 | file | vendor | VMW_VAAIP_SYMM | vendor=EMC model=SYMMETRIX | | | 0 |
| VAAI | 65431 | runtime | vendor | VMW_VAAIP_CX | vendor=DGC model=* | | | 0 |
| VAAI | 65431 | file | vendor | VMW_VAAIP_CX | vendor=DGC model=* | | | 0 |
| VAAI | 65432 | runtime | vendor | VMW_VAAIP_EQL | vendor=EQLOGIC model=* | | | 0 |
| VAAI | 65432 | file | vendor | VMW_VAAIP_EQL | vendor=EQLOGIC model=* | | | 0 |
| VAAI | 65433 | runtime | vendor | VMW_VAAIP_NETAPP | vendor=NETAPP model=LUN* | | | 0 |
| VAAI | 65433 | file | vendor | VMW_VAAIP_NETAPP | vendor=NETAPP model=LUN* | | | 0 |
| VAAI | 65434 | runtime | vendor | VMW_VAAIP_HDS | vendor=HITACHI model=* | | | 0 |
| VAAI | 65434 | file | vendor | VMW_VAAIP_HDS | vendor=HITACHI model=* | | | 0 |
| VAAI | 65435 | runtime | vendor | VMW_VAAIP_LHN | vendor=LEFTHAND model=* | | | 0 |
| VAAI | 65435 | file | vendor | VMW_VAAIP_LHN | vendor=LEFTHAND model=* | | | 0 |

Figure 28. Generic claim rule listing

Note: For Storage vMotion operations, VMware will use a maximum of 64 MB for the extent size. This is a VMware restriction and has nothing to do with the array. No change to the claim rules is required. Remember if VMware can use an extent size up to 240 MB it will, if not it will use something smaller. In the case of Storage vMotion it will not exceed 64 MB per VMware design.

4.2.1.3 Multiple storage arrays and claim rules

Because claim rules for VAAI are unique based on a combination of the vendor and model of array, it is possible to have multiple rules on a single ESXi host which do not conflict with one another. Each claim rule then is applied only to devices from that claim rule's associated vendor and model of array. Furthermore, a claim rule must be unique to vendor and model. As already explained, if a rule already exists for a storage array, attempting to add another will result in an error, forcing the user to first remove the existing rule. Rule numbers, therefore, are arbitrary and do not represent any hierarchy. While VMware uses the rule number '914' in their documentation, any available number can be utilized. In fact, by using the switch '-u,' instead of specifying a rule number, VMware will generate one for the user.

For example, if both PowerMax and PowerStore arrays are utilized by the same ESXi host, the following claim rules could be added. Note how PowerStore uses the 'u' switch so that VMware auto assigns the rule id.

```
esxcli storage core claimrule add -r 914 -t vendor -V EMC -M SYMMETRIX -P
VAAI_FILTER -c Filter
esxcli storage core claimrule add -r 914 -t vendor -V EMC -M SYMMETRIX -P
VMW_VAAIP_T10 -c VAAI -a -s -m 240

esxcli storage core claimrule add -u -t vendor -V DelleMC -M PowerStore -P
VAAI_FILTER -c Filter
esxcli storage core claimrule add -u -t vendor -V DelleMC -M PowerStore -P
VMW_VAAIP_T10 -c VAAI -a -s -m 240
```

Note that not all storage arrays support XCOPY claim rules so review vendor documentation prior to creation.

4.2.2 Virtual Storage Integrator

As an alternative to using CLI, the Virtual Storage Integrator (VSI) plug-in to vCenter can automatically create the claim rule for an ESXi host. If installed, navigate to **Cluster (right-click) -> Dell VSI -> Apply Host Best Practices...** in Figure 29.

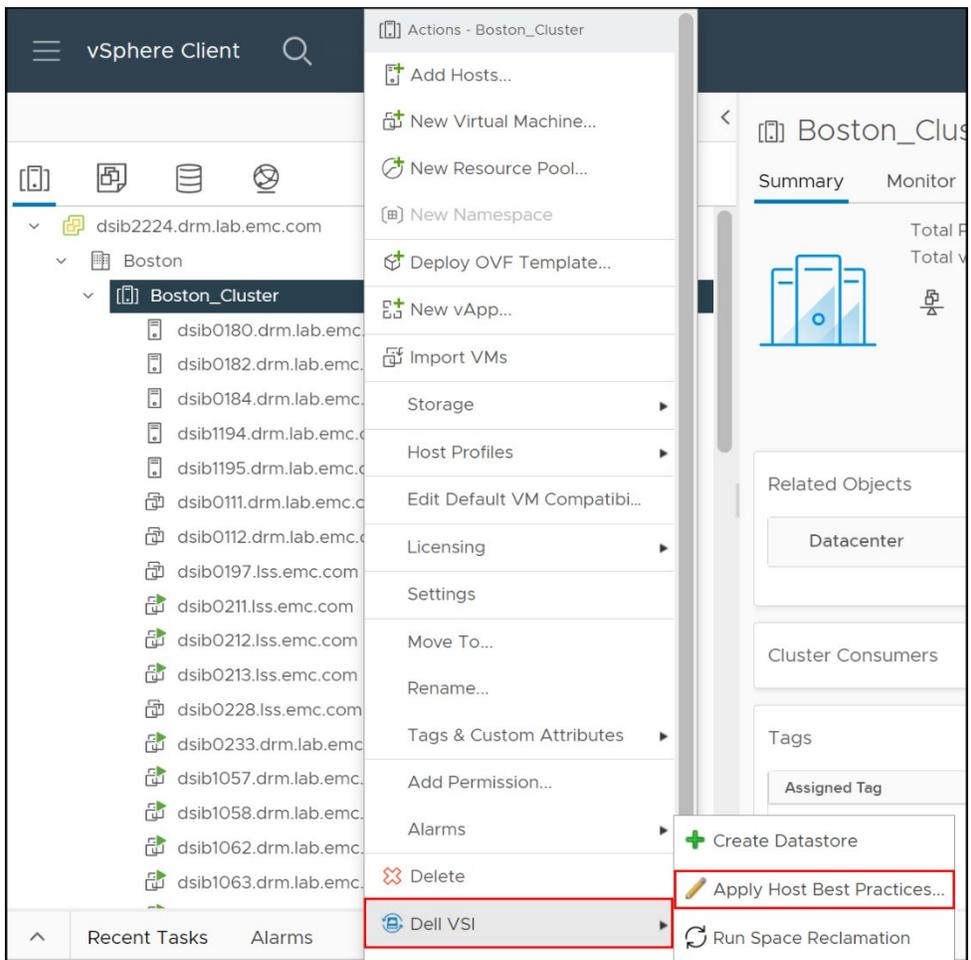


Figure 29. VSI Host Best Practices

In the next dialog in [Figure 30](#), select **PowerMax** as the storage platform from the drop-down box. Check the box next to the **Host Setting** for the **Create VAAI rule....** Click the **SAVE** button to complete. Reboot the hosts, when possible, for the change to take effect.

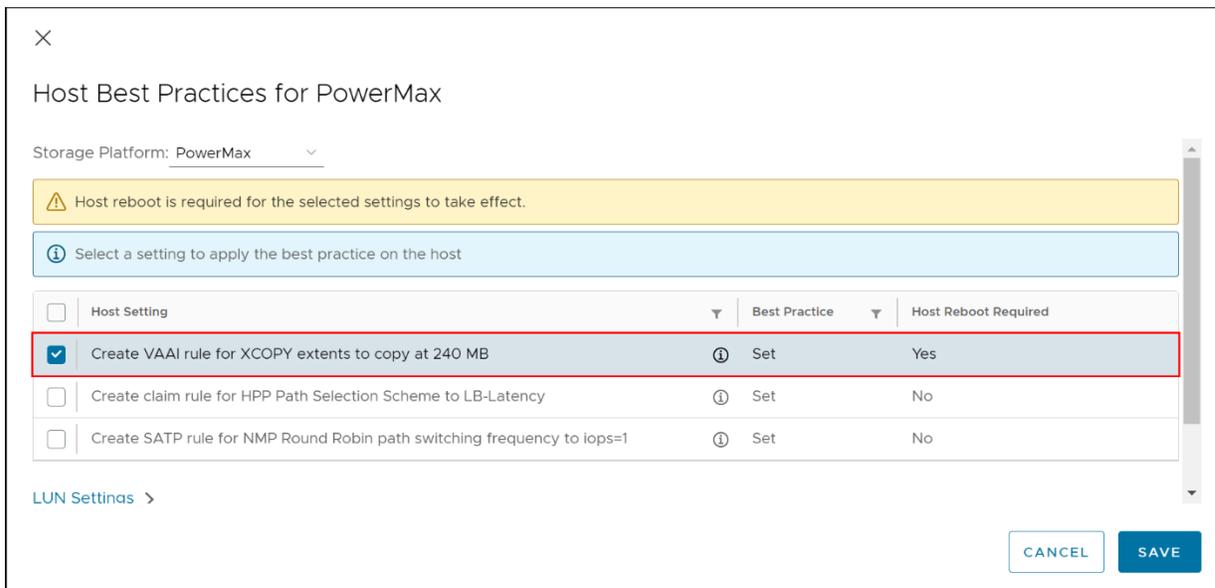


Figure 30. Creating XCOPY rule

Using VSI is the preferred method for adding the claim rule as it removes the complexity of CLI and can be run cluster wide. Note, however, that VSI hard-codes the rule number. If there are multiple arrays in use or the default rule of 914 cannot be used, do not use VSI to create the rule.

4.2.3 Thin vmdks

One area where the increased extent size has made no difference in the past is with thin vmdks as VMware always used 1 MB extents. In vSphere 7 and 8, however, VMware tries to consolidate extents and therefore when cloning will use the largest extent it can consolidate for each copy IO (software or hardware). This new behavior has resulted in two benefits on the PowerMax. The first is the obvious one that any task requiring XCOPY on thin vmdks will be faster than on previous vSphere versions. The second benefit is that a clone of a thin vmdk now behaves like a thick vmdk in terms of further cloning. In other words, if one clones a thin vmdk VM and then uses that clone as a template for further clones, those further clones will be created at a speed almost equal to a thick vmdk VM clone. This is the case whether that initial clone is created with or without XCOPY. The operation is a defragmentation of the thin vmdk. It is critical, therefore, if a thin vmdk VM is going to be a template for other VMs, that rather than converting the VM to template it is cloned to a template. This will ensure all additional deployments from the template will benefit from the consolidation of extents.

4.2.4 Claim rule results

The following graphs, [Figure 31](#) and [Figure 32](#), demonstrate the effectiveness of using claim rules for XCOPY when cloning 100 GB of data. Note that source and target refer to devices. XCOPY only works within an array, not between them.

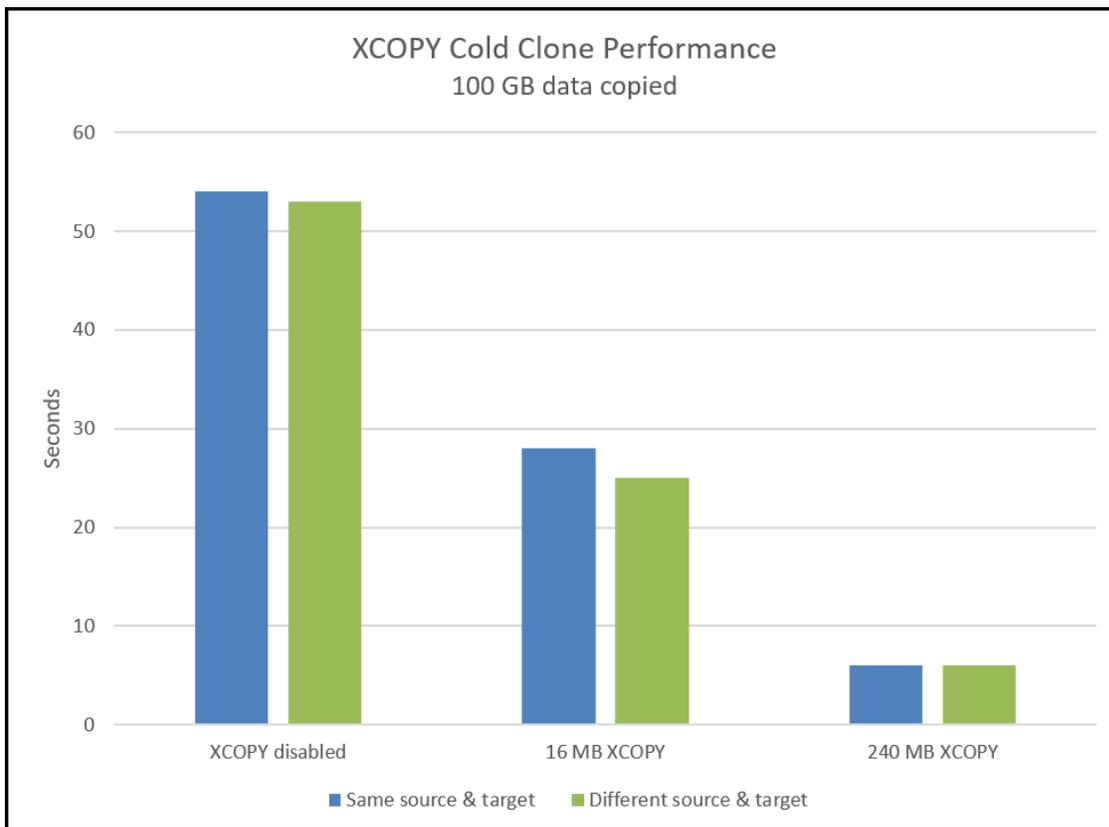


Figure 31. XCOPY cold clone performance on the PowerMax

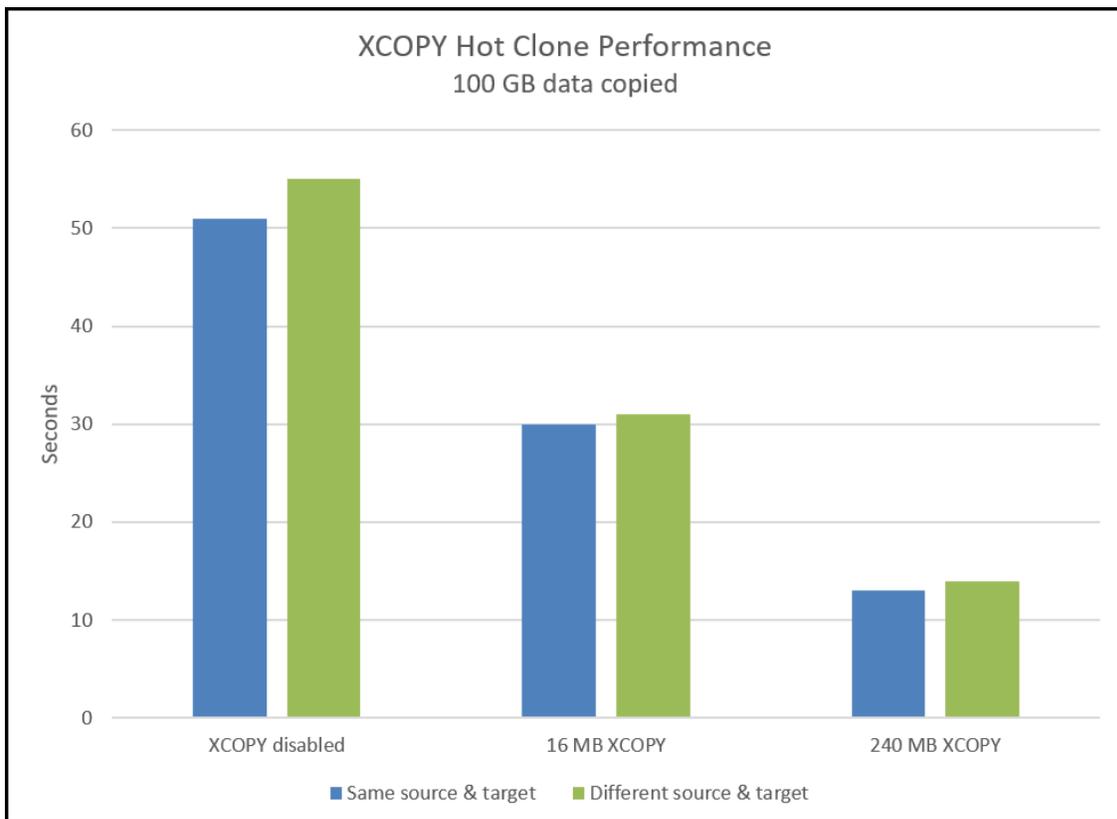


Figure 32. XCOPY hot clone performance on the PowerMax

Similarly, in [Figure 33](#), the benefits of XCOPY on a Storage vMotion are demonstrated. Here, 250 GB of data is moved between datastores. The benefit of the larger 64 MB extent size is apparent in the thin vmk results.

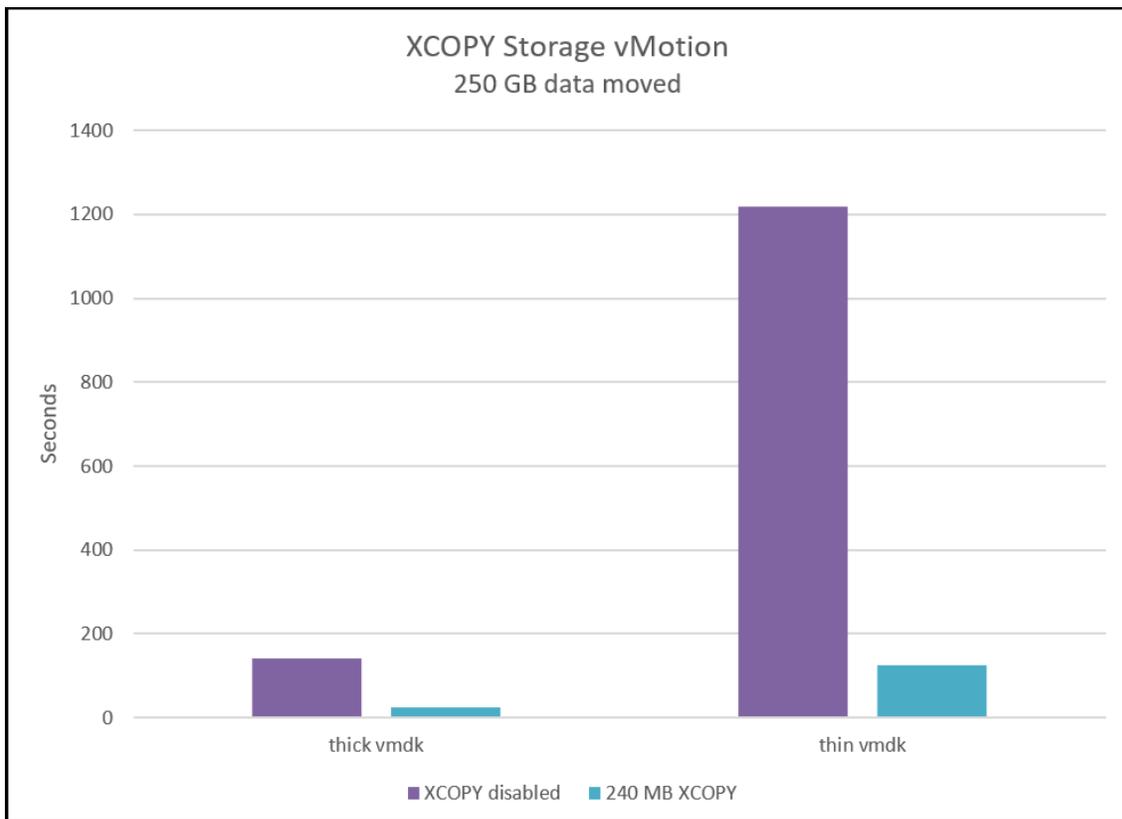


Figure 33. XCOPY SvMotion performance using claim rules with PowerMax

4.3 SRDF and XCOPY

As mentioned previously, XCOPY is implemented asynchronously on the PowerMax. Unfortunately, this implementation can cause a potential issue for customers when XCOPY is used in conjunction with SRDF/Metro or SRDF/Synchronous target devices. Therefore, Dell uses synchronous XCOPY for those target SRDF devices. The synchronous process ensures that a consistency group will not enter a “sync in progress” state. The two main reasons Dell implemented the synchronous copy instead of defaulting to software copy are:

1. Avoid any consistency issues with SRDF devices
2. Offload the copy process from the host to the array, thus saving CPU, memory, and bandwidth.

The performance of the synchronous implementation is, by its nature, more akin to VMware’s software copy than Dell’s asynchronous implementation. As such, while Dell synchronous copy will generally outpace VMware’s host copy (though not always), the improvement will not mirror those for asynchronous copy. It is, however, more desirable to offload the copy to the array to free the host and network resources for more important activities. If the time to complete an XCOPY task is more critical than the resources saved on the host, XCOPY can be disabled for SRDF devices on the array by Dell. For more detail on XCOPY performance considerations, please see Dell KB 533648. An example of performance is shown in [Figure 34](#).

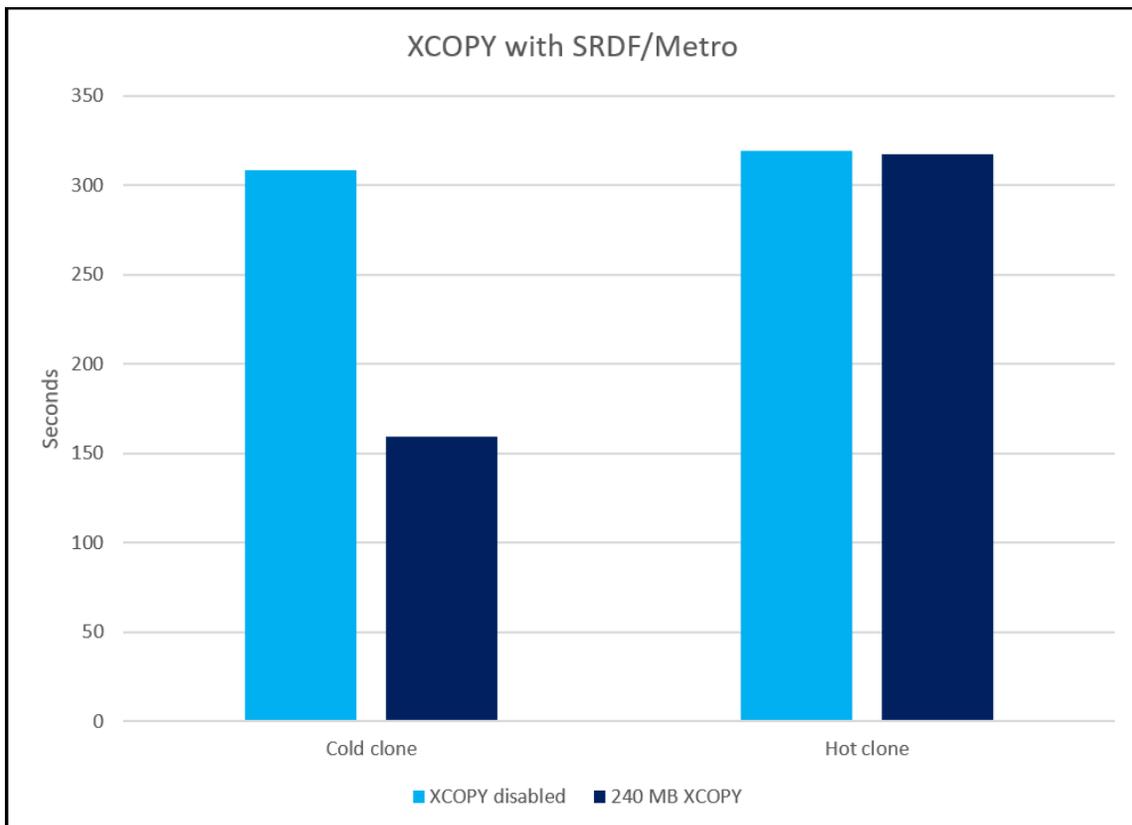


Figure 34. XCOPY with SRDF/Metro

4.4 XCOPY with multiple arrays on an ESXi host

One final scenario where XCOPY best practices for PowerMax may need to be altered, is if the ESXi hosts are zoned to not only a PowerMax, but another Dell array as well, e.g., Unity. Fortunately, as explained above, claim rules are array-specific, and as such their existence on an ESXi host has no bearing on devices presented from other arrays; however, on PowerMax it is always the best practice to also set the default hardware acceleration to its maximum of 16 MB in case there is an issue with the claim rules. When other arrays are present, though, this value should be left at the default of 4 MB as other Dell arrays have different best practices. For more detail, including other vSphere settings in multi-array scenarios, please refer to Dell KB 000020863.

4.5 Caveats for using hardware-accelerated Full Copy⁹

The following are general caveats that Dell provides when using this feature:

- VMware publishes a KB article (1021976) which details when Full Copy will not be issued. Such cases include VMs that have snapshots or cloning or moving VMs between datastores with different block sizes, among others.

⁹ Full Copy (XCOPY) can be disabled at the system level if the caveats are particularly problematic for a customer.

- One case not included is when the configuration parameter `disk.EnableUUID` is set to `TRUE` on Windows VMs. VMware Tools will set this parameter so be aware this will prevent XCOPY.
- Always set `MaxTransferSize` to 16 MB and use claim rules.
- Limit the number of simultaneous clones using Full Copy to three or four. This is not a strict limitation, but Dell believes this will ensure the best performance when offloading the copy process.
- A PowerMax device that has SAN Copy, TimeFinder and certain RecoverPoint sessions will not support Full Copy. Any cloning or Storage vMotion operation run on datastores backed by these volumes will automatically be diverted to the default VMware software copy. Note that the vSphere Client has no knowledge of these sessions and as such the “Hardware Accelerated” column in the vSphere Client will still indicate “Supported” for these devices or datastores.
- If an XCOPY session exists on a device, any TimeFinder operation will fail with the following error: *“Cannot use the device for this function because it is a Copy Session target.”* Once the XCOPY operation completes, the TimeFinder action will succeed.
- Full Copy is not supported for use with Open Replicator. VMware will revert to software copy in these cases.
- Full Copy is not supported for use with Federated Live Migration (FLM) target devices.
- Although SRDF is supported with Full Copy, certain RDF operations, such as an RDF failover, will be blocked until the PowerMax has completed copying the data from a clone or Storage vMotion.
- When the available capacity of a Storage Resource Pool (SRP) reaches the Reserved Capacity value, active or new Full Copy requests can take a long time to complete and may cause VMware tasks to timeout. For workarounds and resolutions please see Dell KB article 503348.
- Full Copy cannot be used between devices that are presented to a host using different initiators. For example, take the following scenario:
 - A host has four initiators
 - The user creates two initiator groups, each with two initiators
 - The user creates two storage groups and places the source device in one, the target device in the other
 - The user creates a single port group
 - The user creates two masking views, one with the source device storage group and one of the initiator groups, and the other masking view with the other initiator group and the storage group with the target device.
 - The user attempts a Storage vMotion of a VM on the source device to the target device. Full Copy is rejected, and VMware reverts to software copy.

4.5.1 SRDF/Metro specific

The following are caveats for SRDF/Metro when using Full Copy. In SRDF/Metro configurations the use of Full Copy does depend on whether the site is the one supplying the external WWN identity.

- Full Copy will not be used between a non-SRDF/Metro device and an SRDF/Metro device when the device WWN is different from the external device WWN. Typically, but not always, this means the non-biased site (recall even when using witness, there is a bias

site). In such cases Full Copy is only supported when operations are between SRDF/Metro devices or within a single SRDF/Metro device; otherwise, software copy is used.

- As Full Copy is a synchronous process on SRDF/Metro and SRDF/S devices, which ensures consistency, performance will be closer to software copy than asynchronous copy.
- While an SRDF/Metro pair is in a suspended state, Full Copy reverts to asynchronous copy for the target R1. It is important to remember, however, that Full Copy is still bound by the restriction that the copy must take place on the same array. For example, assume SRDF/Metro pair AA, BB is on array 001, 002. In this configuration device AA is the R1 on 001 and its WWN is the external identity for device BB, the R2 on 002. If the pair is suspended and the bias is switched such that BB becomes the R1, the external identity remains that of AA from array 001 (i.e., it appears as a device from 001). The device, however, is presented from array 002 and therefore Full Copy will only be used in operations within the device itself or between devices on array 002.

4.5.2 Compression/Deduplication (Data Reduction) specific

The following are caveats when using Full Copy with devices in storage groups with compression or compression/deduplication enabled. Note that deduplication is only available on the PowerMax platform and only in conjunction with compression. It is not possible to enable one without the other. As such, references to compression below also include the PowerMax.

- When Full Copy is used on a device in a storage group with compression enabled, or between storage groups with compression enabled, Full Copy will uncompress the data during the copy and then re-compress on the target.
- When Full Copy is used on a device in a storage group with compression enabled, to a storage group without compression enabled, Full Copy will uncompress the data during the copy and leave the data uncompressed on the target.
- When Full Copy is used on a device in a storage group without compression enabled to a storage group with compression enabled, the data will remain uncompressed on the target and be subject to the normal compression algorithms.
- As Full Copy is a background process, deduplication of data, if any, may be delayed until after all tracks have been copied. Full clone copies have been shown to deduplicate at close to 100% in enabled storage groups.

5 Block Zero Detail

Hardware-accelerated Block Zero provides benefits and performance increases in a variety of use cases. This paper will cover the two most encountered scenarios. The first scenario discussed will be concerned with deploying fully allocated virtual machines (virtual machines using the “eagerzeroedthick” virtual disk allocation mechanism) on the PowerMax. The second use case discusses the performance gains achieved using Block Zero when performing I/O in a virtual machine with virtual disks using the zeroedthick allocation format on the PowerMax.

5.1.1 Eagerzeroedthick disks

In certain situations, virtual disks are deployed using the “eagerzeroedthick” allocation mechanism to fully zero out the disk at creation. A VMware feature that requires EZT disks is VMware Fault Tolerance. Fault Tolerance requires that all virtual disks in use by protected virtual machines use this allocation mechanism. In other cases, virtual disks are created using the “thin” or “zeroedthick” allocations mechanisms where space is not fully allocated on creation of the virtual disk. Both these virtual disks will not zero out the space until the guest OS writes data to a previously unallocated block. This behavior inherently will cause a performance drop when allocating new space as the kernel must first write zeros to initialize the new space before the virtual machine can write actual data. Therefore, the “eagerzeroedthick” allocation mechanism is used to avoid performance degradations caused by the on-the-fly zeroing of “thin” or “zeroedthick” virtual disks.

Since “eagerzeroedthick” virtual machines are entirely zeroed at the initial creation, it can take a great deal of time to write these zeros, especially in virtual machines with large virtual disks. Furthermore, this creation operation consumes SAN bandwidth during execution. With the introduction of Block Zero, the zeroing is offloaded to the array, which saves not only on SAN bandwidth but also takes advantage of the increased processing power inherent on the PowerMax. In the case of the PowerMax the zeros are not actually written at all.

Note: As the PowerMax does not actually reserve space, it seems logical to conclude that there is little difference between using an eagerzeroedthick and a zeroedthick virtual disk on that platform. While this is true from a storage performance perspective, it is important to remember that if VMware or an application requires the use of an eagerzeroedthick virtual disk (e.g., Fault Tolerant VM or Oracle RAC), that disk type must be selected when creating the virtual machine.

In this use case, new eagerzeroedthick virtual disks of three sizes 100 GB, 500 GB, and 1000 GB were deployed with hardware-accelerated Block Zero (WRITE SAME) disabled and enabled. As shown by the graph in [Figure 35](#), the noted improvements are significant.

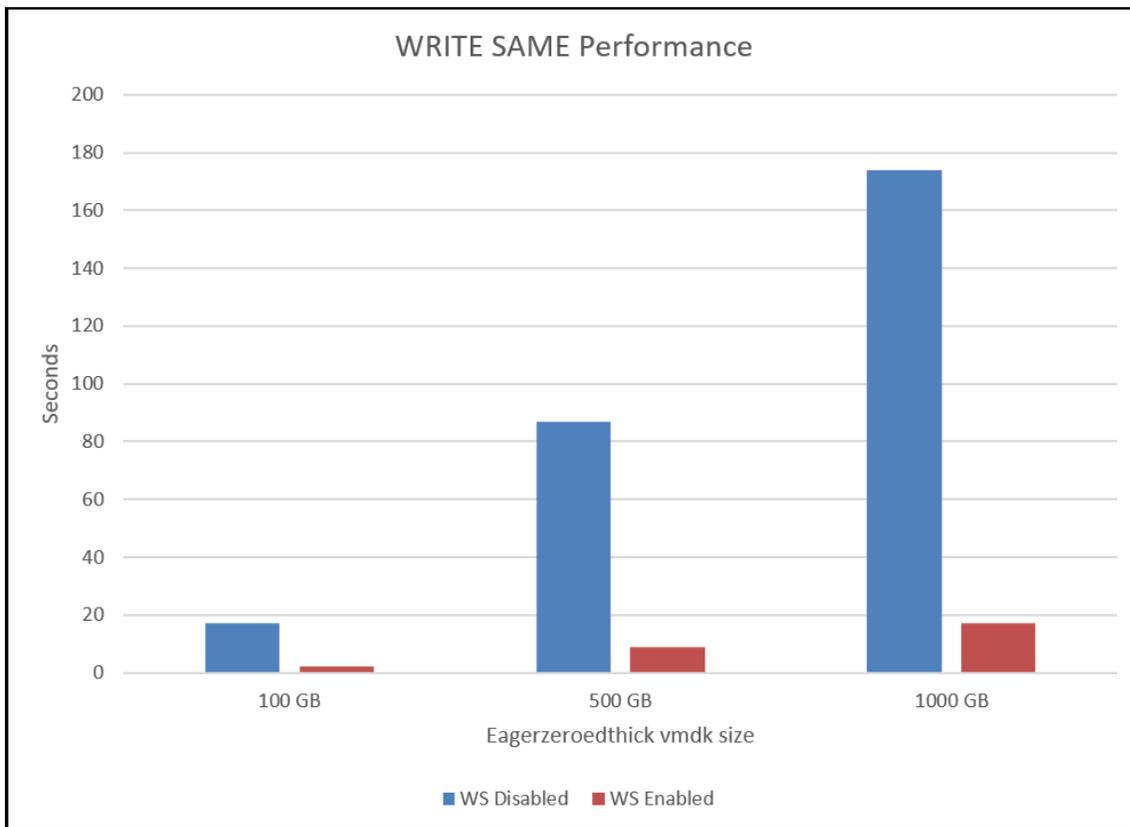


Figure 35. Fully allocated virtual disk deployment times using Block Zero

In addition to time savings, there is a substantial difference in throughput during the creation of the virtual disks when Block Zero is off and on. Figure 36 shows the ESXi performance graph of the write rate in KB per second (throughput) of the target PowerMax device from the ESXi server creating the virtual disk. A dramatic difference can be noted between when Block Zero is off and on. A virtual disk is created with hardware acceleration off and takes about three minutes and has a write rate exceeding 6,000,000 KB/s. After the virtual disk creation completes, hardware-accelerated Block Zero is enabled and a second identical virtual disk is created on the same target PowerMax device. The throughput required to deploy this is so low that it is unseen on the same scale as the first virtual disk creation. The throughput spikes at around 4,000 KB/s for around one second — a 1500x decrease in required throughput to create the virtual disk!



Figure 36. Throughput difference deploying an EZT virtual disk with Block Zero disabled and enabled

5.1.2 Zeroedthick virtual disks

By default, new virtual disks are created using the “zeroedthick” allocation mechanism. This is also the recommended disk format for use with PowerMax to improve storage efficiencies by not writing zeros on creation. In this allocation scheme, the storage required for the virtual disks is reserved in the datastore but the VMware kernel does not initialize all the blocks with zeros. The blocks are initialized with zeros by the ESXi kernel in response to I/O activities to previously uninitialized blocks by the guest operating system. Consequently, as discussed in the previous use case, there is overhead when writing to unallocated regions of the virtual disk as the ESXi kernel first must write zeros to the previously unallocated space before the guest can complete any new write operations. Prior to the VAAI primitives, to avoid this overhead, “eagerzeroedthick” virtual disks were used to ensure optimal performance. Unfortunately, especially in the case of Virtual Provisioning, “eagerzeroedthick” virtual disks wasted inordinate amounts of space in the thin pool due to the zeros written at creation. A decision therefore needed to be made between consuming more capacity with “eagerzeroedthick” virtual disks and accepting the inherent performance overhead associated with “zeroedthick” virtual disks. With the advent of Block Zero, however, that performance overhead is negated as the zeros are no longer written to initialize blocks before the guest OS writes to them.

5.2 Caveats for using hardware-accelerated Block Zero

The following are general caveats that Dell provides when using this feature:

- Block Zero is currently not supported for use with Open Replicator. VMware will revert to traditional writes in these cases.
- Block Zero commands will take longer to execute on devices in SRDF relationships.

6 UNMAP Detail

The VMware use cases for UNMAP¹⁰ revolve around the time it takes to reclaim storage space after the deletion of virtual disks, virtual machines, or the execution of a Storage vMotion. All of these tasks create empty space on the datastore but not on the array device. While Dell recommends the use of automated UNMAP for VMFS 6 datastores, there are use cases where customers wish to use manual UNMAP. Below is a discussion of both options, automated and manual.

6.1 Automated UNMAP

VMware offers automated UNMAP capability at the VMFS 6 datastore level (VMFS 5 is not supported). The automation is enabled by a background process (UNMAP crawler) that keeps track of which LBAs can be unmapped and then issues those UNMAPs at various intervals to maintain the reclaim rate. It is an asynchronous process. By default, the reclaim rate for UNMAP is 26 MB/s and is the recommended setting on PowerMax as will be explained.

Note: UNMAP commands can be sent by any ESXi host and concurrently with any other ESXi host. If, for example, two VMs on the same datastore are deleted, each residing on a different ESXi host, each of those hosts can send UNMAP commands for their respective VMs. The rate of reclaim, by default 26 MB/second, is enforced at the host level, not the datastore level. Thus, in this example the reclaim rate could run as high as 52 MB/s for that datastore. Therefore, increasing the reclaim rate above the default value can have a multiplying effect.

UNMAP is enabled by default when a VMFS 6 datastore is created through the vSphere Client wizard. There are two parameters available, one of which can be modified: Space Reclamation Granularity and Space Reclamation Priority. The first parameter, Space Reclamation Granularity, designates at what size VMware will reclaim storage. It cannot be modified from the 1 MB value. The second parameter, Space Reclamation Priority, dictates how aggressive the UNMAP background process will be when reclaiming storage. This parameter only has two values: None (off), and Low. It is set to Low by default. These parameters are seen in [Figure 37](#).

¹⁰ The UNMAP process is also known as Dead Space Reclamation and for the purposes of this paper the terms reclaim or unmapping should also be considered synonymous with the UNMAP process.

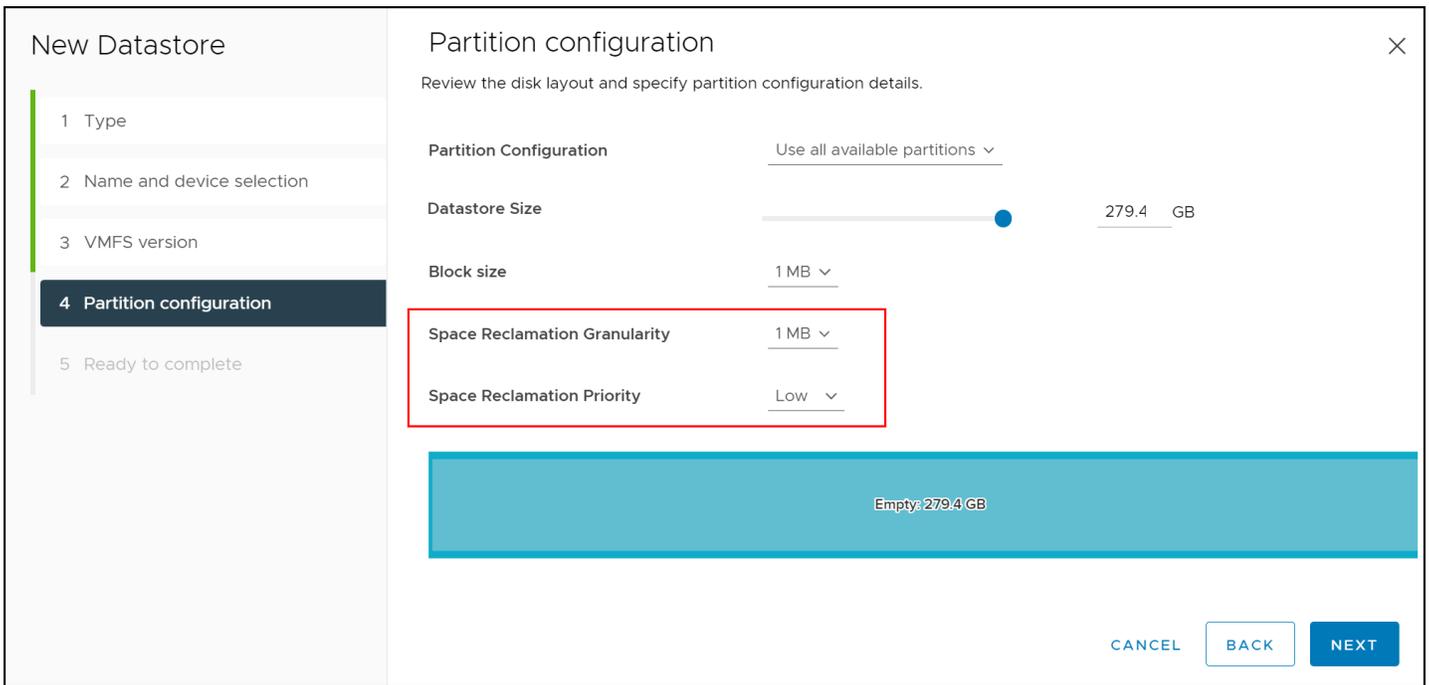


Figure 37. Enabling automated UNMAP on a VMFS 6 datastore

If it becomes necessary to either enable or disable UNMAP on a VMFS 6 datastore after creation, the setting can be modified through the vSphere Client or CLI. Within the datastore view in the Client, navigate to the Configure tab and the General sub-tab. On this page, find a section on Space Reclamation where the Edit button can be used to make the desired change as in [Figure 38](#).

The screenshot shows the configuration page for a VMFS 6 datastore. The 'Space Reclamation' section is expanded, showing 'Space reclamation' is 'Enabled at Low priority: Deleted or unmapped blocks are reclaimed on the LUN at low priority'. A modal dialog titled 'Space Reclamation Settings' is open, showing options to 'Enable automatic space reclamation at fixed rate' (with a 'Reclamation rate' of 100 MB/s) or 'Disable automatic space reclamation' (selected). A red arrow points to the 'EDIT...' button in the 'Space Reclamation' section.

Figure 38. Modifying automated UNMAP on a VMFS 6 datastore

Changing the parameter is also possible through CLI, demonstrated in Figure 39.

```

10.108.245.143 - PuTTY
[root@dsib1143:~] esxcli storage vmfs reclaim config set --volume-label VMFS6_UNMAP_TEST --reclaim-priority none
[root@dsib1143:~] esxcli storage vmfs reclaim config get --volume-label VMFS6_UNMAP_TEST
    Reclaim Granularity: 1048576 Bytes
    Reclaim Priority: none
[root@dsib1143:~]
    
```

Figure 39. Modifying automated UNMAP through CLI

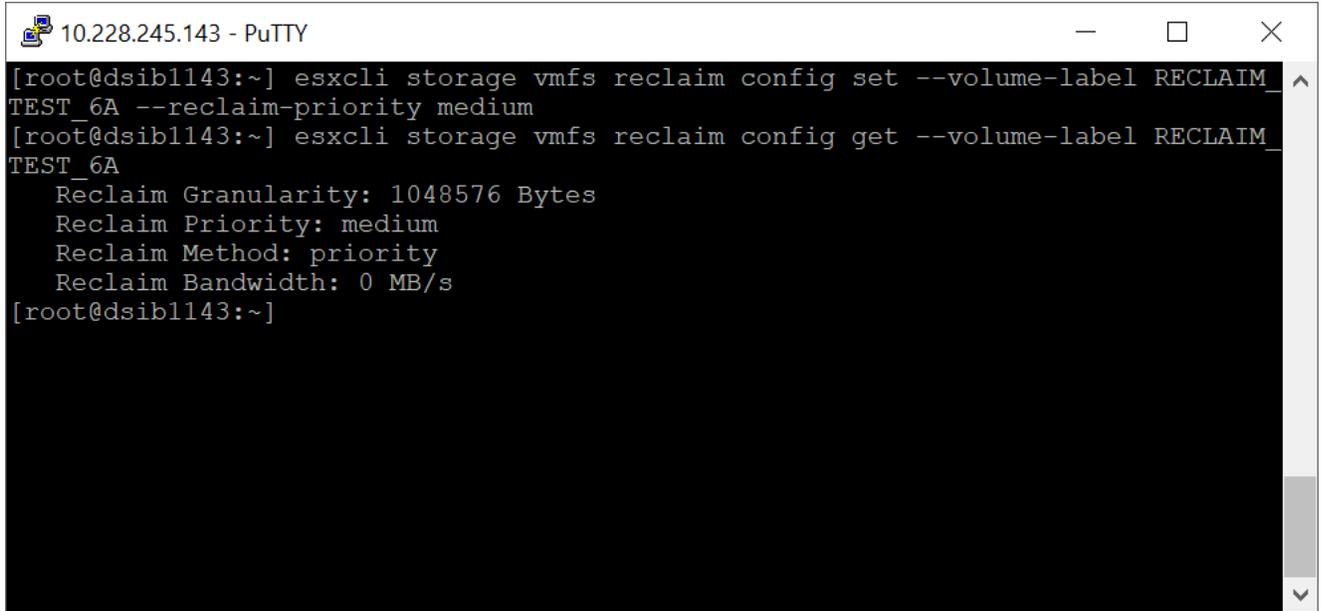
6.1.1 Reclamation priority

VMware offers the ability to not only change the reclamation priority that is used for UNMAP, but also change the method from priority to a fixed of bandwidth if desired. Priority can be changed from the default of low to medium or high, each having their own bandwidth setting. Medium will

translate into 77 MB/s while high will be 256 MB/s.¹¹ To change the priority, however, only the CLI is offered.

By default, the “Low” priority setting shown in [Figure 37](#) translates into reclaiming at a rate of 26 MB/s. To change it to medium priority, issue the following command seen in [Figure 40](#):

```
esxcli storage vmfs reclaim config set -volume-label RECLAIM_TEST_6A -
reclaim-priority medium
```



```
10.228.245.143 - PuTTY
[root@dsib1143:~] esxcli storage vmfs reclaim config set --volume-label RECLAIM_TEST_6A --reclaim-priority medium
[root@dsib1143:~] esxcli storage vmfs reclaim config get --volume-label RECLAIM_TEST_6A
Reclaim Granularity: 1048576 Bytes
Reclaim Priority: medium
Reclaim Method: priority
Reclaim Bandwidth: 0 MB/s
[root@dsib1143:~]
```

[Figure 40. Modifying reclaim priority](#)

To take advantage of the power of all flash storage arrays, VMware also offers the ability to set a fixed rate of bandwidth. The user can set the value anywhere from 100 MB/s to a maximum of 2000 MB/s in multiples of one hundred. This capability, unlike priority, is offered in the GUI. In the following example in [Figure 41](#), the reclaim method is changed to “fixed” at a 1000 MB/s bandwidth. In this interface the priority cannot be changed as previously mentioned.

¹¹ There is a bug in vSphere 7 where the bandwidth shows as 0 MB/s. This is fixed in vSphere 8.

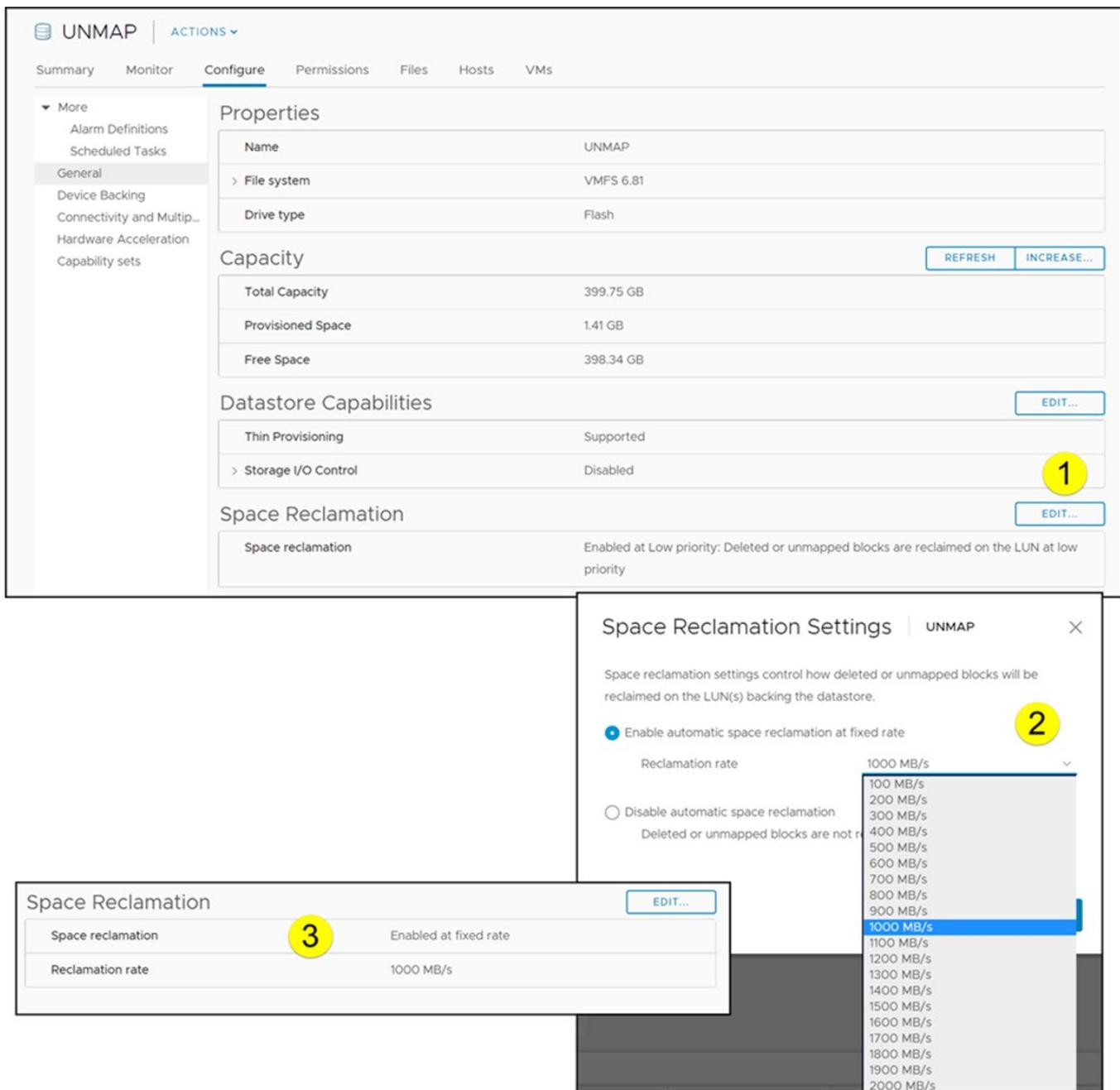


Figure 41. Modifying reclaim method to fixed

The CLI can also be used for the fixed method. For example, to change the reclaim rate to 200 MB/s on the datastore “RECLAIM_TEST_6A” issue the following:

```
esxcli storage vmfs reclaim config set -volume-label RECLAIM_TEST_6A -reclaim-method fixed -b 200
```

Below is the actual output in Figure 42, along with the `config get` command which retrieves the current value.

```

10.228.245.143 - PuTTY
[root@dsib1143:/vmfs/volumes] esxcli storage vmfs reclaim config set --volume-label RECLAIM_TEST_6A --reclaim-method fixed -b 200
[root@dsib1143:/vmfs/volumes] esxcli storage vmfs reclaim config get --volume-label RECLAIM_TEST_6A
  Reclaim Granularity: 1048576 Bytes
  Reclaim Priority: none
  Reclaim Method: fixed
  Reclaim Bandwidth: 200 MB/s
[root@dsib1143:/vmfs/volumes]
    
```

Figure 42. Changing reclaim priority

VMware has indicated that they do not advise changing the reclaim granularity.

Note: While an active UNMAP operation is underway, changing the bandwidth value will take longer than normal to complete.

One may notice that in [Figure 40](#), the reclaim bandwidth shows as 0 MB/s, despite knowing the actual value is 75 MB/s. VMware will not show the correct value, however, unless the reclaim method is first set to fixed, then changed back to priority as in [Figure 43](#).

```

10.228.245.143 - PuTTY
[root@dsib1143:~] esxcli storage vmfs reclaim config set --volume-label RECLAIM_TEST_6A --reclaim-priority medium
[root@dsib1143:~] esxcli storage vmfs reclaim config get --volume-label RECLAIM_TEST_6A
  Reclaim Granularity: 1048576 Bytes
  Reclaim Priority: medium
  Reclaim Method: priority
  Reclaim Bandwidth: 0 MB/s
[root@dsib1143:~] esxcli storage vmfs reclaim config set --volume-label RECLAIM_TEST_6A --reclaim-method fixed --reclaim-bandwidth 100
[root@dsib1143:~] esxcli storage vmfs reclaim config set --volume-label RECLAIM_TEST_6A --reclaim-priority medium
[root@dsib1143:~] esxcli storage vmfs reclaim config get --volume-label RECLAIM_TEST_6A
  Reclaim Granularity: 1048576 Bytes
  Reclaim Priority: medium
  Reclaim Method: priority
  Reclaim Bandwidth: 75 MB/s
[root@dsib1143:~]
    
```

Figure 43. Showing the actual reclaim bandwidth of medium priority

6.1.2 Changing reclamation values in vSphere 7 and 8

There is an additional requirement when changing the UNMAP method and bandwidth, or simply enabling/disabling UNMAP in the vSphere Client. After the change is made to the datastore on an ESXi host, the datastore must be unmounted and remounted on every other ESXi host that accesses the datastore for the change to propagate. If using CLI to make the change, the new value will take effect immediately on the initial ESXi host, but not on the other hosts until the unmount/remount process is complete. Although the unmount/remount process is the only one supported by VMware, changing the value on each ESXi host that accesses the datastore through CLI may also accomplish the same goal. This is not the case for the vSphere Client, however.

6.1.3 Reclamation rate changes in vSphere 8

While the default of low priority for UNMAP is the preferred value for automation, nevertheless some customers with extremely active arrays have found the value of 26 MB/s to produce too much load on the system. To help address these customer environments in vSphere 8, VMware took a two-fold approach.

First, VMware added a new floor to the fixed values. In vSphere 7 the lowest setting is 100 MB/s but has been reduced further in vSphere 8 to 10 MB/s as seen in [Figure 44](#).

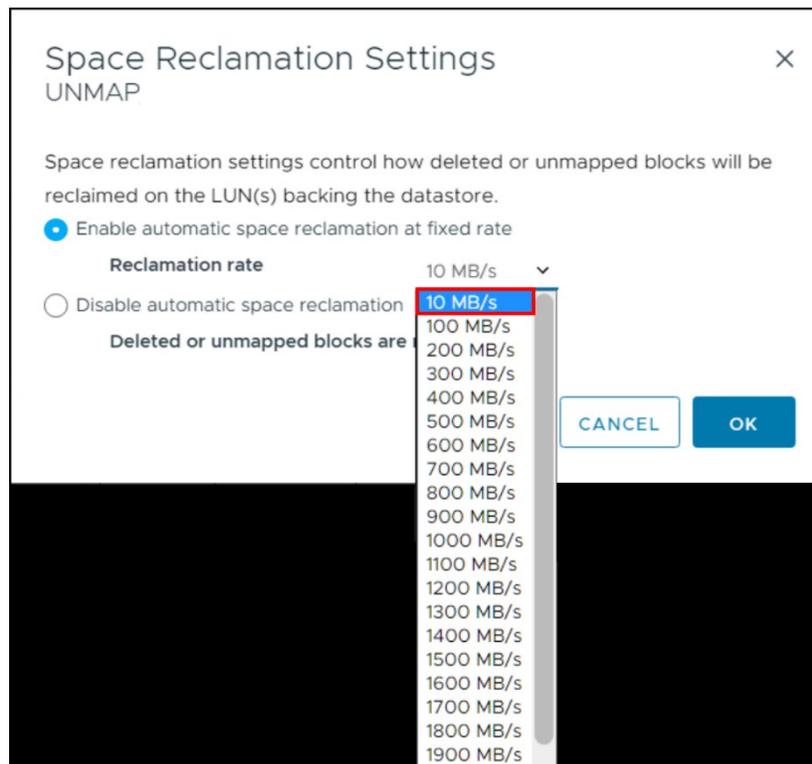


Figure 44. vSphere 8 minimum fixed rate

The reduction in size may allow some customers to continue to use automated UNMAP on extremely active arrays. It is important to remember, however, that the time to UNMAP will be increased when the rate is lowered. To use 10 MB/s customers must follow the unmount/remount datastore procedure explain in [Changing reclamation values in vSphere 7](#) since it is not possible to set a fixed value on a datastore at creation time, only to set the priority to low or none.

The second change VMware made was to add a new, low priority scheduling queue just for UNMAP. The queue functions as follows: If an array is not active, the set rate will be honored; however, on a heavily active array, the queue might result in a reduced rate. By throttling the rate,

VMware can respond to those arrays under heavy load which cannot process the UNMAP commands quickly enough. Note that the queue is under the control of VMware and it is not adjustable.

6.1.4 Reclamation hosts in 8.0U3

VMware introduced an additional UNMAP feature in 8.0U3 that allows customers to throttle how many ESXi hosts are permitted to issue UNMAP. As explained in the section Automated UNMAP, VMware does not restrict the number of ESXi hosts that can issue UNMAP commands against the same datastore, regardless of the priority or fixed max. In 8.0U3, the user can now set a max value for reclaim hosts, thus ensuring the fixed or priority setting is honored. In [Figure 45](#), the max value is adjusted down to a single host from the default of 128 (the max size of a vSphere cluster).

```
[root@dsib1001:~] esxcli storage vmfs reclaim config set -l 1473_DS_1 -m priority -p low -n 1
[root@dsib1001:~] esxcli storage vmfs reclaim config get -l 1473_DS_1
Reclaim Granularity: 1048576 Bytes
Reclaim Priority: low
Reclaim Method: priority
Reclaim Bandwidth: 26 MB/s
Reclaim Max Hosts: 1
[root@dsib1001:~]
```

Figure 45. Changing reclaim max hosts

6.1.5 Automatic UNMAP support for SESparse

For VM's with a SESparse snapshot (SESparse is the default snapshot format on VMFS 6), the space reclamation is also automatic. Note that automatic UNMAP for SESparse will not be triggered until 2 GB of cleared space is achieved.

6.2 Recommendations

Dell conducted numerous tests on UNMAP capabilities but found only minor differences in how quickly data is reclaimed on the array. This is consistent with manual UNMAP block options. For example, in [Figure 46](#) there are two reclaim tasks shown. Each one is to UNMAP 100 GB of storage from the datastore RECLAIM_TEST_6A. The first line is using a fixed method at 1000 MB/s. The second is a priority method at medium (75 MB/s). Each reclaim took the same amount of time and VMware performed the same UNMAP commands.

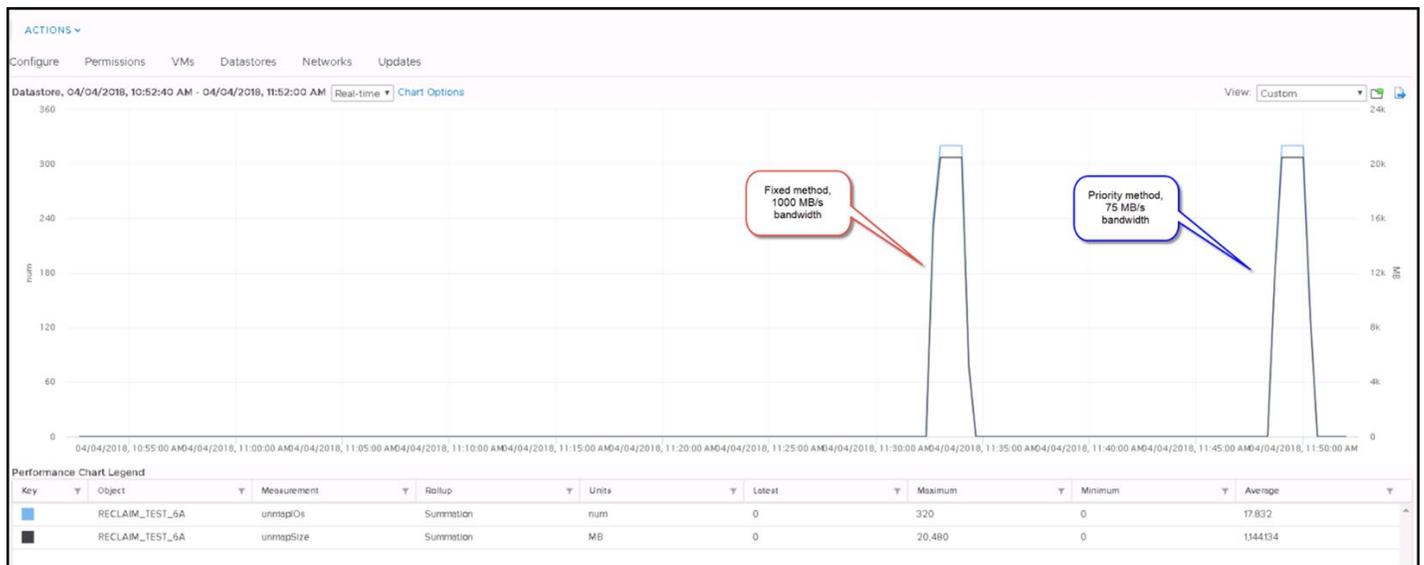


Figure 46. UNMAP IOPs and size

Because this is only an example and data will be stored in different extent sizes depending on many factors, it is possible larger fixed values will perform better than the priority method, despite these findings. However, at higher bandwidths and priority, if the array is already operating at close to peak, performance can be impacted; therefore, the recommendation is to continue to use the priority method at the low setting, or the default. VMware keeps the UNMAP rate set to low priority to minimize the impact on VM IO, without regard to the type of storage backing the datastores.

The following is a discussion of use cases for manual UNMAP.

6.3 Manual UNMAP after VM deletion

This example will demonstrate the manual reclamation of a datastore’s storage using ESXi CLI. The environment for the test consists of:

- A single device (4FD), in a storage group (data reduction enabled), presented to the ESXi cluster.
- 1 TB datastore, NO_AUTO_UNMAP, created on that device, with auto unmap disabled.
- 1 Windows VM, NO_AUTO_UNMAP_VM_1, with 600 GB of storage. Almost all of that storage is comprised of files on NTFS. There is minimal free space.

In Figure 47, VMware reports that the VM is occupying 601.42 GB. Unisphere for PowerMax has a similar allocation in Figure 48.

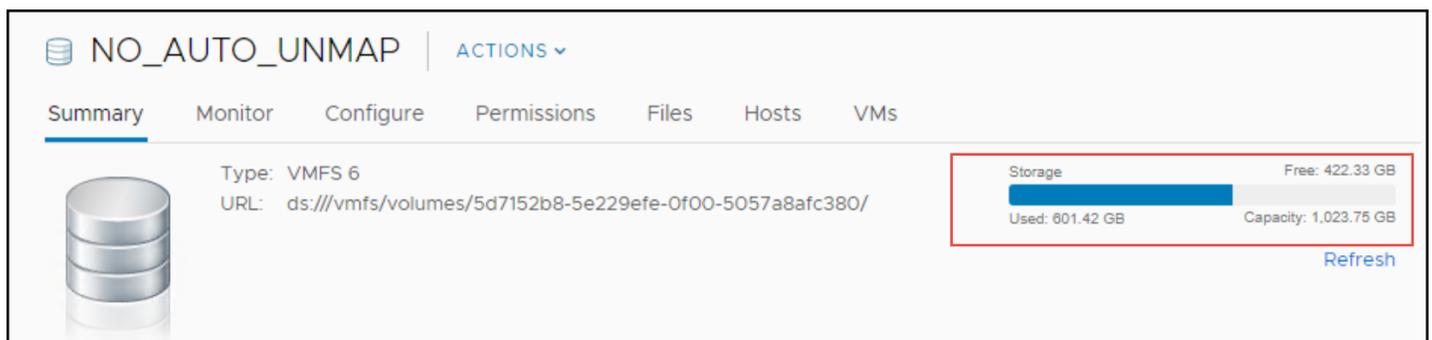


Figure 47. Datastore usage in vCenter

| Name | Type | Allocated (%) | Capacity (GB) | Emulation | Status |
|-------|------|---------------|---------------|-----------|--------|
| 004FD | TDEV | 62% | 1,024.00 | FBA | Ready |

Figure 48. Obtaining device allocation in Dell Unisphere for PowerMax

More detail about how PowerMax has allocated the storage is available through SYMCLI. Note in Figure 49 how tracks are in different pools, depending on how they can be compressed. In addition, the allocated tracks are different from exclusive tracks since some data in this VM is already available on the array and thus has been deduplicated. So, while the VM would normally consume 600 GB of storage, it only needs 421 GB¹² on the PowerMax.

```

10.228.246.116 - PuTTY
dsib2116:~ # symcfg -sid 357 list -tdev -dev 4FD -detail

Symmetrix ID: 000197600357

Enabled Capacity (Tracks) : 430489080
Bound Capacity (Tracks) : 8388615

SYMMETRIX THIN DEVICES
-----
Sym  Bound  Flags  Total  Pool  Pool  Exclusive  Comp
     Pool Name  ESPT  Tracks  Subs  Allocated  Allocated  Tracks  Ratio
-----
004FD -      FS.B   8388615  -    0 0    3452190  1.0:1
      DG1_F_F   -.--   -      -    4696662  56      -      -
      DG1_F_1   -.--   -      -    19429    0      -      -
      DG1_F_6   -.--   -      -    99304    1      -      -
      DG1_F_2   -.--   -      -    11082    0      -      -
      DG1_F_8   -.--   -      -    47154    1      -      -
      DG1_F_9   -.--   -      -    19797    0      -      -
Total
Tracks 8388615 -    4893428  58    3452190

Legend:
Flags: (E)mulation : A = AS400, F = FBA, 8 = CKD3380, 9 = CKD3390
        (S)hared Tracks : S = Shared Tracks Present, . = No Shared Tracks
        (P)ersistent Allocs : A = All, S = Some, . = None
        S(T)atus : B = Bound, I = Binding, U = Unbinding, A = Allocating,
                  D = Deallocating, R = Reclaiming, C = Compressing,
                  N = Uncompressing, F = FreeingAll, . = Unbound
dsib2116:~ #
    
```

Figure 49. Allocation detail with compression pools

¹² The total is calculated by multiplying the number of exclusive tracks by the track size of 128 KB.

Once the VM is deleted or relocated with Storage vMotion, VMware will recognize that the datastore is empty in [Figure 50](#), but as automatic UNMAP is not enabled, the allocation on the array remains as it did in [Figure 49](#).

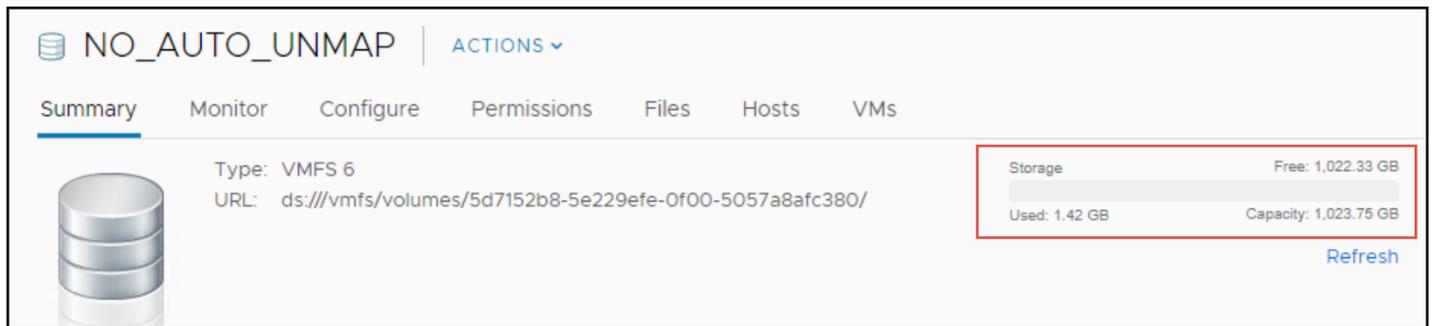


Figure 50. Datastore usage after VM deletion

At this point space can be reclaimed using a manual UNMAP. One thing to understand about manual reclaim is that VMware cannot do a “precision” UNMAP like automatic reclamation. If auto unmap was enabled on this datastore, VMware would only send commands for those LBAs which used to represent the VM. With manual reclamation, VMware must iterate all the segments of the entire datastore, reclaiming what it can. Therefore, the time it takes to reclaim will be dependent both on the size of the device and the amount of storage to reclaim, with the device size being the key factor in timing.

In the above example, the following script in [Figure 51](#) was run which records the time before and after the UNMAP command. Note we do not recommend changing the reclaim blocks from the default of two hundred, which is 200 MB reclaimed for each iteration. To reclaim the 421 GB on a 1 TB datastore took 58 seconds.

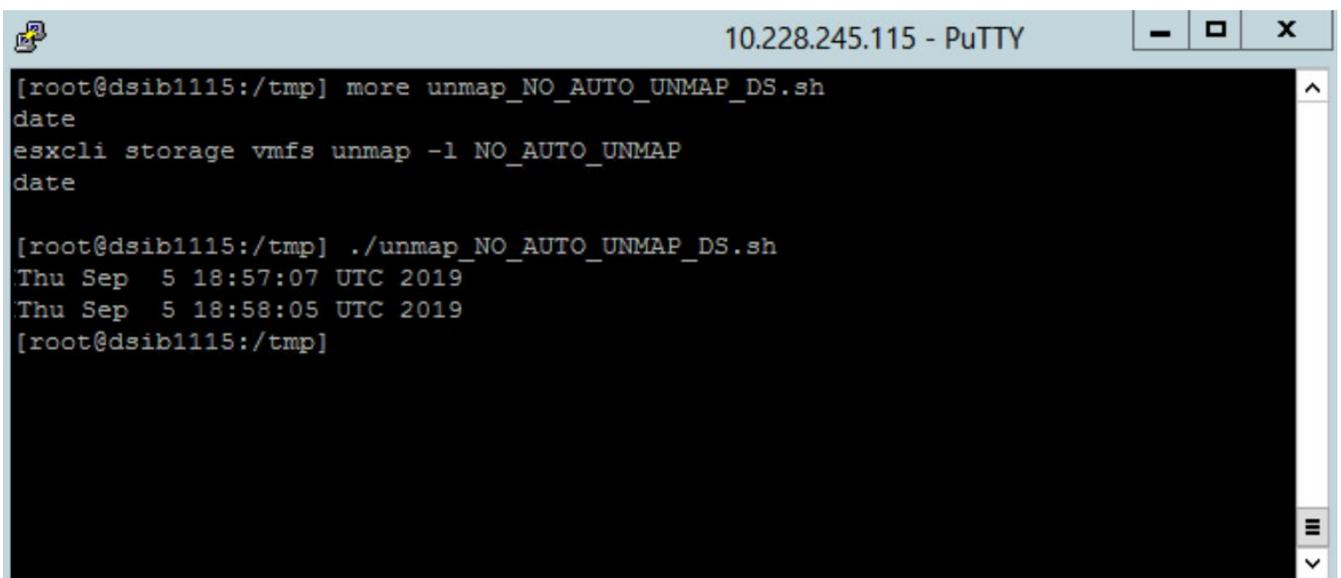


Figure 51. Manual UNMAP command

Here is the device allocation after the run in [Figure 52](#).

```

10.228.246.116 - PuTTY
dsib2116:~ # symcfg -sid 357 list -tdev -dev 4FD -detail

Symmetrix ID: 000197600357

Enabled Capacity (Tracks) : 430489080
Bound Capacity (Tracks) : 8388615

-----
S Y M M E T R I X   T H I N   D E V I C E S
-----
Sym      Bound   Flags      Total      Pool      Pool      Exclusive      Comp
Pool Name ESPT      Tracks     Subs      Allocated  Allocated  Allocated      Ratio
              Tracks  (%)      Tracks  (%)      Tracks
-----
004FD   -         FS.B       8388615    -         0    0         310  16.0:1
          DGl_F_F   -.--       -         -         213  0         -    -
          DGl_F_1   -.--       -         -         286  0         -    -

Total
Tracks                8388615    -         499  0         310

Legend:
Flags: (E)mulation : A = AS400, F = FBA, 8 = CKD3380, 9 = CKD3390
(S)hared Tracks : S = Shared Tracks Present, . = No Shared Tracks
(P)ersistent Allocs : A = All, S = Some, . = None
S(T)atus      : B = Bound, I = Binding, U = Unbinding, A = Allocating,
                D = Deallocating, R = Reclaiming, C = Compressing,
                N = Uncompressing, F = FreeingAll, . = Unbound
dsib2116:~ #
    
```

Figure 52. Device allocation after a manual reclaim

So, all storage (save datastore metadata or any blocks that could not be reclaimed) has been returned to the SRP and is available for other devices. Note what happens in [Figure 53](#) if the script is run again, even though there is nothing left to reclaim. The run takes almost as long as the first reclaim, 46 seconds. This is because VMware still must examine all the segments even though there is nothing left to reclaim. Interestingly, if compared to the first run, it shows that the actual UNMAP process took about 12 seconds. Hence, device size is particularly important to consider when running a manual UNMAP. Extremely large datastores will take time and impact ongoing IO on the datastore, and therefore it is advisable to execute any manual reclamations during maintenance windows.

```

10.228.245.115 - PuTTY
[root@dsib1115:/tmp] more unmap_NO_AUTO_UNMAP_DS.sh
date
esxcli storage vmfs unmap -l NO_AUTO_UNMAP
date

[root@dsib1115:/tmp] ./unmap_NO_AUTO_UNMAP_DS.sh
Fri Sep 6 19:10:07 UTC 2019
Fri Sep 6 19:10:53 UTC 2019
[root@dsib1115:/tmp]
    
```

Figure 53. Running an additional reclamation

6.4 UNMAP internals

Dead space is reclaimed in multiple iterations. A user can provide the number of blocks (the default is two hundred and recommended) to be reclaimed during each iteration. This specified block count controls the size of each temp file that VMware creates as part of the process. Since vSphere defaults to 1 MB block datastores, this would mean that if no number is passed, VMware will unmap 200 MB per iteration. VMware still issues UNMAP to all free blocks, even if those blocks have never been written to. VMware will iterate serially through the creation of the temp file as many times as needed to issue UNMAP to all free blocks in the datastore.

6.4.1 Recommendation

When employing manual UNMAP, Dell recommends using VMware’s default block size number (two hundred) based on internal testing, though fully supports any value VMware does. Below is an example of running manual UNMAP on a PowerMax with ESXi 7, using the default block size. The test was conducted using a single 2 TB datastore, with auto unmap disabled. A 500 GB vmdk, completely filled, was cloned once, twice, and three times. The vmdks were then deleted and a manual UNMAP run against each datastore. The results are reported in [Figure 54](#).

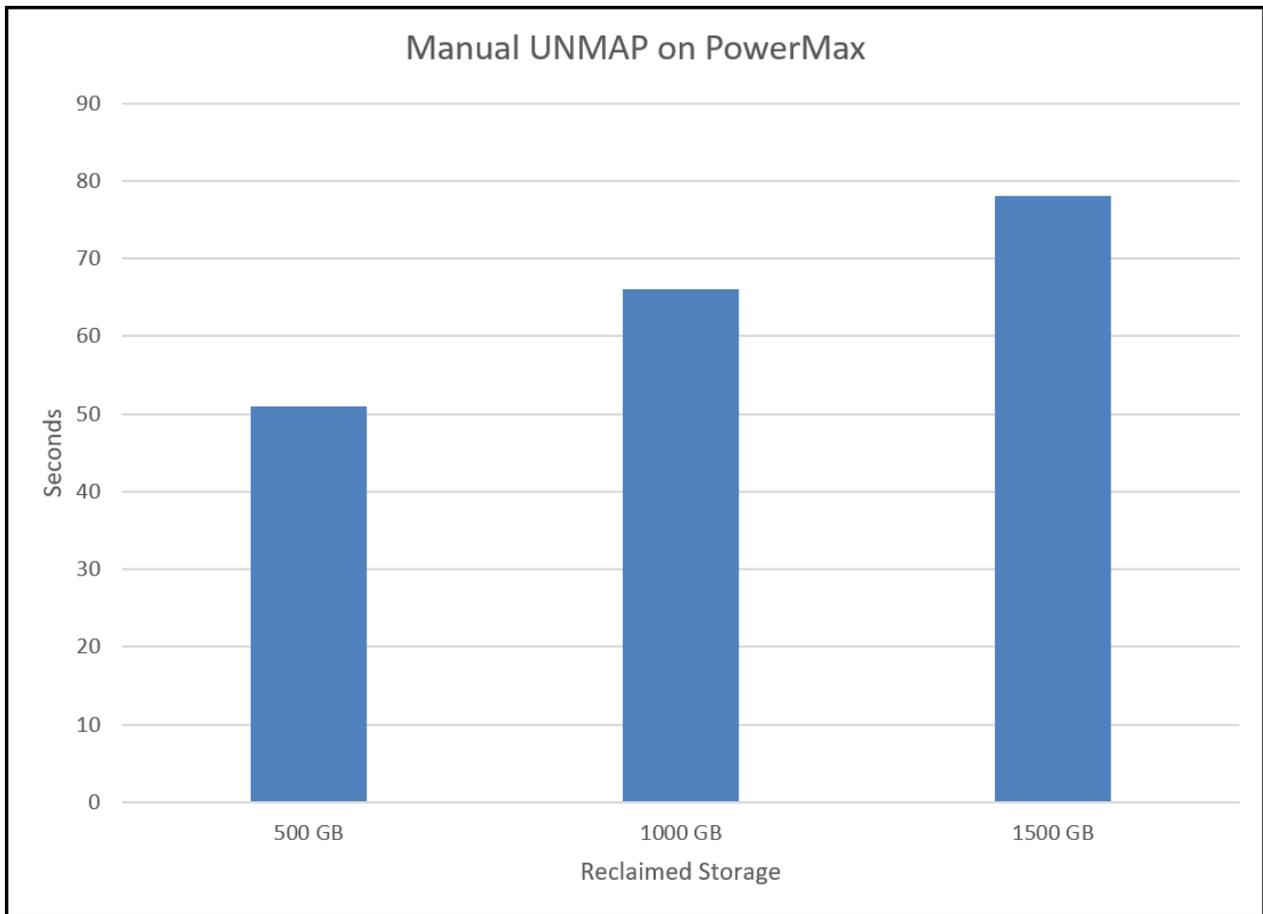


Figure 54. Manual UNMAP – PowerMax and ESXi 7

6.5 Caveats for using UNMAP

The following are some general caveats that Dell provides when using this feature:

- UNMAP is supported on SRDF devices, however, though the R1 receives bulk commands from VMware, those commands must be broken down on the array before sending over to the R2 device. This means UNMAP, in general, increases SRDF traffic.
- UNMAP is an online activity and can be run on datastores with active VMs, however manually reclaiming substantial amounts of storage at one time can impact device and thus VM performance. Therefore, Dell recommends running any manual reclaim either during maintenance windows or during a time of lower business activity.
- When an UNMAP process is run on a device (datastore or GuestOS), the PowerMax prioritizes releasing the storage on the device over the thin pool. This means it is possible that even if the device reports the space as reclaimed, the array may still be freeing space from the thin pool and SRP. This is as designed since it ensures that control of the device in VMware is returned to the user as soon as possible so they can continue to work. Normally the asynchronous nature of UNMAP is not an issue, however if the thin pool or SRP as a whole is approaching capacity, it may be necessary to monitor the release of storage in the SRP after a manual UNMAP before re-using the space the device now shows as free. To avoid this problem in an environment where the SRP is approaching capacity, Dell does not advise performing many large, manual UNMAP procedures of datastores at one time. In addition, if using VMFS6, automated UNMAP is strongly recommended as the freeing of extents occurs in smaller amounts, and only as needed.
- Beginning with vSphere 8.0U2, VMware supports manual UNMAP on config vVols. Config vVols are formatted with VMFS6, and therefore are a datastore. Note, however, that automated UNMAP is not yet supported.
- UNMAP is not supported on devices in use by Open Replicator, RecoverPoint, or FTS.
- UNMAP can be partially blocked by thin devices that have persistent allocations. The PowerMax accepts the UNMAP command but respects persistent allocations. If the specified UNMAP range has persistent and non-persistent allocations, the PowerMax will de-allocate the non-persistent allocations and leave the persistent allocations.

7 Monitoring VAAI operations with ESXTOP

The VMware ESXTOP tool provides a real-time view (updated every five seconds, by default) of ESXi Server performance details. ESXTOP lists CPU utilization for each physical processor, memory utilization, and disk and network bandwidth/information for each network and disk device available to the ESXi server. These statistics let you monitor the resource utilization for each of your virtual machines and the ESXi kernel. ESXTOP also provides VAAI statistics for each device present to the given ESXi host and allows users to monitor the number of commands being issued (if any) and if there are failures.

To use ESXTOP, login to the console of the ESXi host directly or via SSH.

ESXTOP can either be run in live mode to monitor in real time or batch mode to collect statistics over a period. For the purposes of this paper live mode will be used. For information on batch mode refer to VMware documentation.

ESXTOP is launched by executing the command `esxtop`. ESXTOP defaults to showing CPU information of current running processes. In order to see VAAI information the “disk device” screen must be selected. To change to this screen simply hit the “u” key (no quotation marks) on your keyboard and the screen will automatically change to the “disk device” screen (as seen in [Figure 55](#)) which will list the storage devices and their live statistics.

| DEVICE | PATH/WORLD/PARTITION | DQLEN | WQLEN | ACTV | QU |
|---------------------------------------|----------------------|-------|-------|------|----|
| eui.28600af20000000464b94eb917052f0f | - | 2032 | - | 0 | |
| eui.28600af30000000564b94eb917052f0f | - | 2032 | - | 0 | |
| eui.fd8080b917052f0f285fbc300000000 | - | 32 | - | 0 | |
| eui.fd8080b917052f0f28600af4000000006 | - | 512 | - | 0 | |
| eui.fd8080b917052f0f28600af5000000008 | - | 32 | - | 0 | |
| eui.fd8080b917052f0f72c9113e00000000a | - | 512 | - | 0 | |
| mpx.vmhbal:C0:T7:L0 | - | 31 | - | 0 | |
| naa.60000970000120200302533030303031 | - | 64 | - | 0 | |
| naa.60000970000120200341533030313339 | - | 64 | - | 0 | |
| naa.60000970000120200341533030313341 | - | 64 | - | 0 | |
| naa.60000970000120200341533030313342 | - | 64 | - | 0 | |
| naa.60000970000120200341533030313343 | - | 64 | - | 0 | |
| naa.60000970000120200341533030313344 | - | 64 | - | 0 | |
| naa.60000970000120200341533030313345 | - | 64 | - | 0 | |
| naa.60000970000120200341533030313533 | - | 64 | - | 0 | |
| naa.60000970000120200341533030323431 | - | 64 | - | 0 | |
| naa.60000970000120200341533030323432 | - | 64 | - | 0 | |
| naa.64cd98f06eb01b0024a5055d3e753724 | - | 128 | - | 0 | |

Figure 55. Disk device ESXTOP screen

For the purposes of VAAI monitoring, most of the statistics shown by default are unnecessary so they can be removed to make the display more manageable. To edit what is shown press the “f” key on the keyboard and this will show the “Current Field Order” screen. In this screen the user can re-order or add/remove statistics. For VAAI monitoring only the columns “Device Name,” “VAAI Stats” and the “VAAI Latency Stats” are required. Statistics can be added or removed by locating their assigned letter to the left of the field name and toggling the asterisk next to it by typing the associated letter. The presence of an asterisk means the column/field is enabled and will appear when the user navigates back to the “Disk Device” screen. In this example neither of the VAAI fields are enabled so they were enabled by pressing their respective assigned letters, “o” and “p.” Four other default fields which were unnecessary for VAAI purposes, were disabled by toggling their asterisk by typing their respective lower-case letters. These selections can be seen in [Figure 56](#).

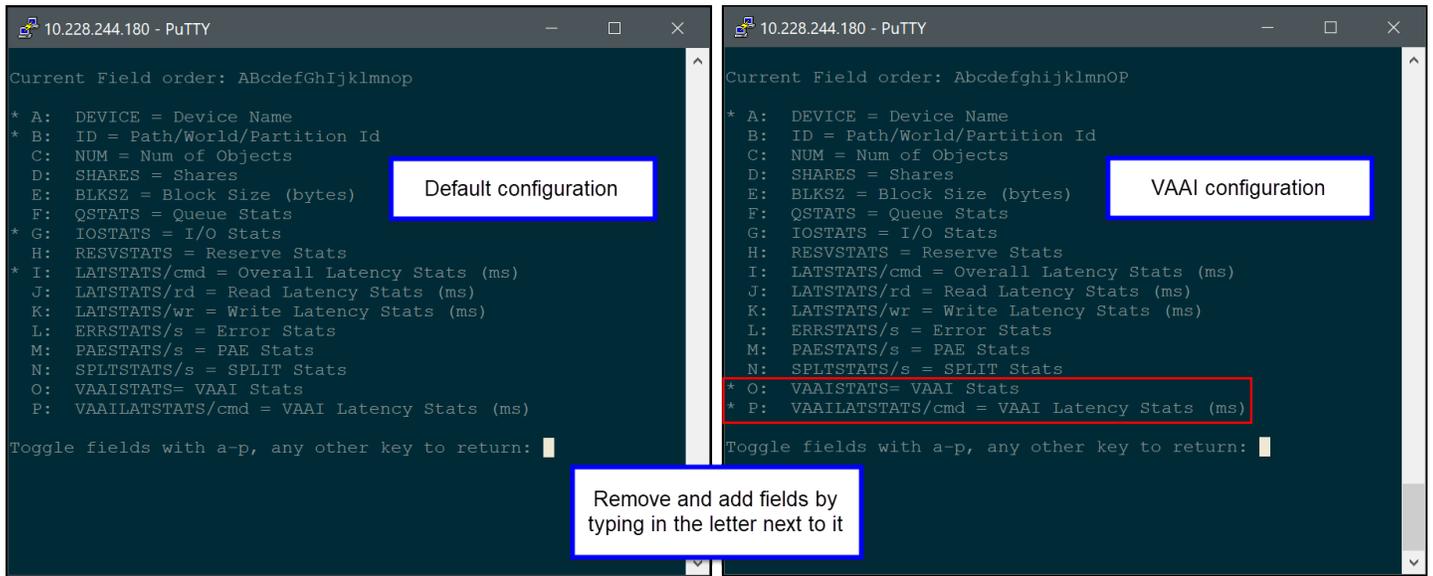


Figure 56. Adding VAAI statistics

Once the statistics desired have been added and others removed, return to the original screen by pressing return or enter on the keyboard. Figure 57 displays the VAAI statistic columns while Table 2 defines the most critical ones.

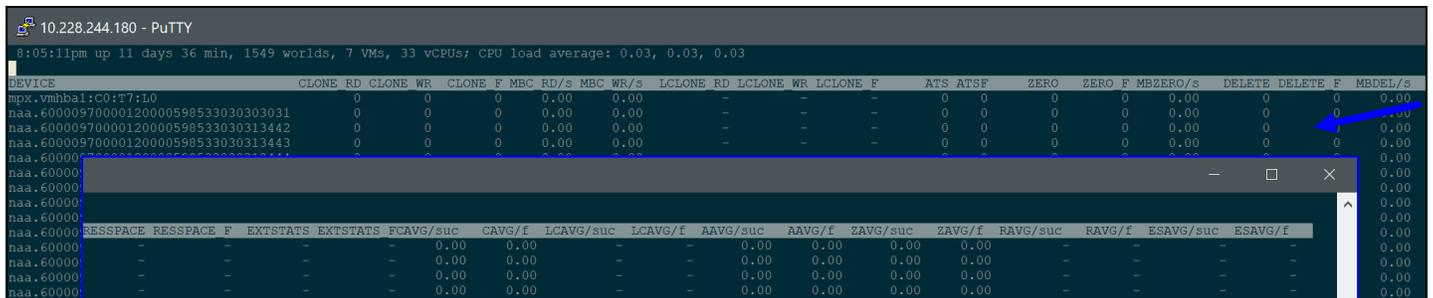


Figure 57. ESXTOP VAAI Statistic counters

Table 2. ESXTOP VAAI Statistics

| Statistic | Description |
|-----------|---|
| CLONE_RD | The number of XCOPY commands successfully completed where this device was a source. |
| CLONE_WR | The number of XCOPY commands successfully completed where this device was a target. |
| CLONE_F | The number of failed XCOPY commands. |
| MBC_RD/s | XCOPY source reads throughput in MB per second. |
| MBC_WR/s | XCOPY target reads throughput in MB per second. |
| ATS | Successful hardware assisted locking/Atomic Test & Set commands |
| ATSF | Failed hardware assisted locking/Atomic Test & Set commands |
| ZERO | Successful WRITE SAME commands |

| | |
|----------|--|
| ZERO_F | Failed WRITE SAME commands |
| MBZERO/s | WRITE SAME throughput in MB per second |
| DELETE | Successful UNMAP commands |
| DELETE_F | Failed UNMAP commands |
| MBDEL/s | UNMAP throughput in MB per second |

Note that some VAAIStats counters are running totals and some are averages. The throughputs are averages for the last interval (default is five seconds and can be changed by typing “s” and in inputting another time and pressing enter) so if a VAAI command has not been run in the last interval the number will be zero. All of the other commands are running totals since the last reboot.

A common use case for the ESXTOP VAAI statistics is to find out if XCOPY is being rejected after noticing a task that uses it, e.g., clone, is running slow. By watching the CLONE_F column for the given device, a user can watch for failed XCOPY commands to appear¹³. If the CLONE_F number increments and the CLONE_RD/CLONE_WR statistics are not incrementing for that device, VMware is using software copy. Review the device configuration to see if one of the known caveats is to blame. Be aware, however, that if VMware knows XCOPY does not apply, it will not even attempt the commands. An example of this is if the UUID is enabled on the VM. Figure 58 shows an ESXTOP example of a XCOPY failure that occurred after some successful commands. It reports four failed XCOPY operations before aborting XCOPY and reverting to traditional software copy.

```

svpg-dell-c4-s03.ebc.emc.local - PuTTY
7:45:48pm up 117 days 18:45, 296 worlds, 0 VMs, 0 vCPUs; CPU load average: 0.01, 0
DEVICES
CLONE RD CLONE WR CLONE F MBC RD/s MBC WR/s
mpx.vmhba32:CD:TD:LD 0 0 0 0.00 0.00
mpx.vmhba33:CD:TD:LD 0 0 0 0.00 0.00
naa.5000cca0004caad4 0 0 0 0.00 0.00
naa.60000970000194901253533030324133 34926 34926 4 1013.95 1013.95
    
```

Figure 58. ESXTOP XCOPY Statistics

ESXTOP will only show the number of commands sent for XCOPY. It cannot show the user how large the extent size of each command request is. When using claim rules, therefore, it is important to get a baseline of a VM clone with the transfer size of 16 MB before adding the claim rule so that it will be easy to determine if the new 240 MB extent size is in use.

In addition to the “VAAIStats” are the “VAAILATSTATS/cmd” which shows the average latency for the different VAAI commands in milliseconds. Table 3 shows the most important available statistics.

Table 3. ESXTOP VAAI Latency Statistics

¹³ The exact number of expected failures cannot be provided as it can vary depending on the number of logical paths to the device(s), the Path Selection Policy (PSP) and the Multi-Pathing Plug-in (MPP). In general, however, this number will be low (single digits or low double digits).

| Statistic | Description |
|-----------|---|
| CAVG/suc | The average latency of successful XCOPY commands in milliseconds. |
| CAVG/f | The average latency of failed XCOPY commands in milliseconds. |
| AAVG/suc | The average latency of successful hardware-assisted locking/ATS commands in milliseconds. |
| AAVG/f | The average latency of failed hardware-assisted locking/ATS commands in milliseconds. |
| ZAVG/suc | The average latency of successful WRITE SAME commands in milliseconds. |
| ZAVG/f | The average latency of failed WRITE SAME commands in milliseconds. |

If VAAI operations are taking a long time or failures are reported, the VAAI latency statistics (Figure 59) are a valuable mechanism to help troubleshoot the problem. If for instance XCOPY sessions are running slower than normal but there are no rejections reported in the CLONE_F column, abnormally high latency could be a factor. If the latencies are much higher than the baseline this could indicate resource contention on the SAN fabric or the frontend of the PowerMax array. At that point, the System Administrator would need to troubleshoot the problem through Unisphere for PowerMax or other SAN fabric tools.

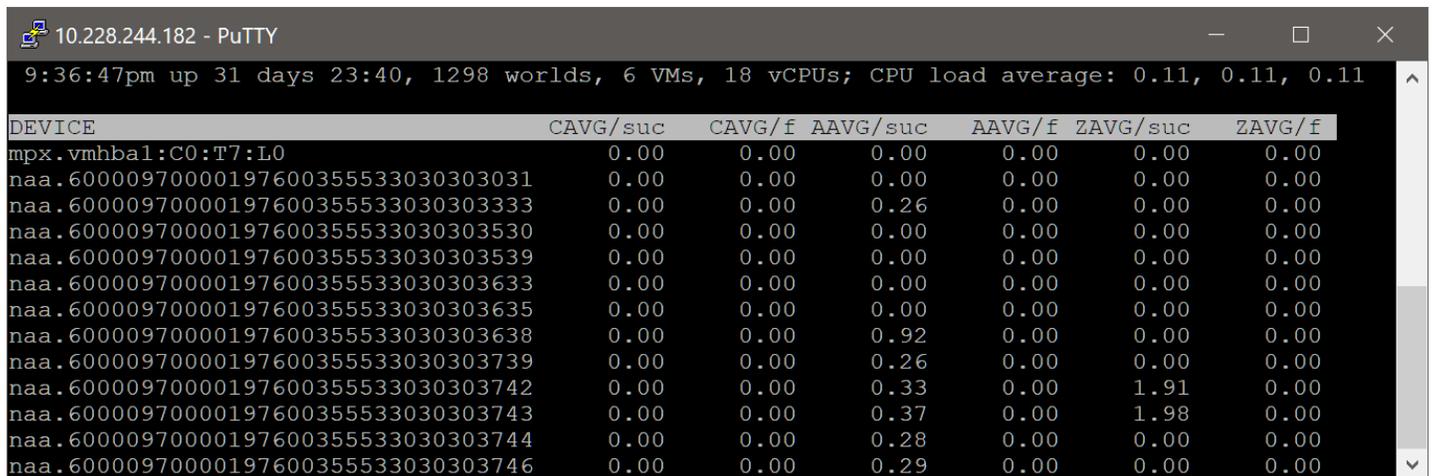
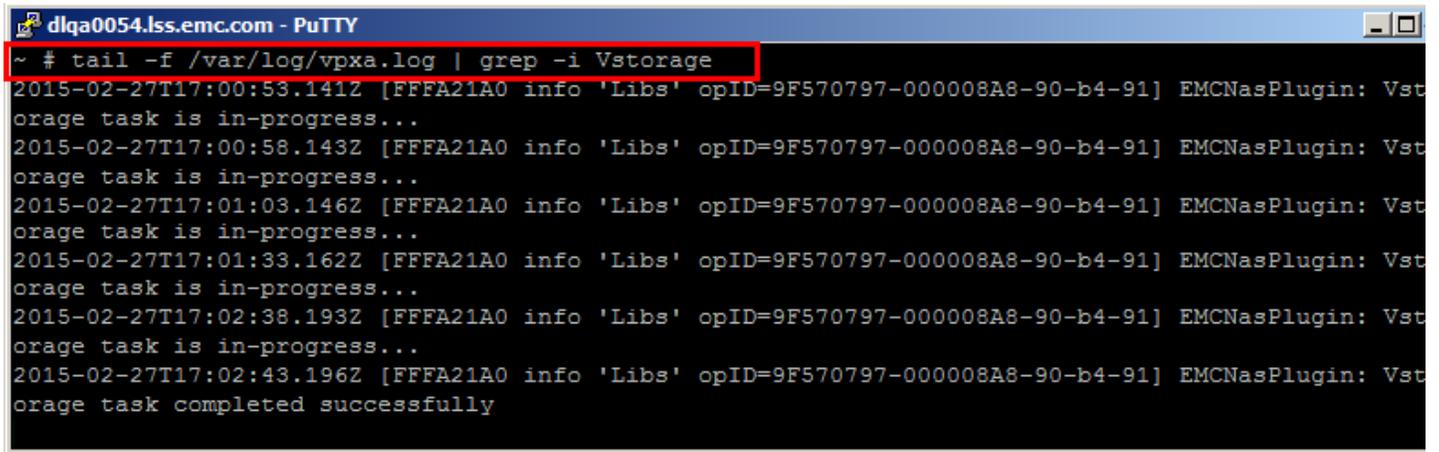


Figure 59. ESXTOP VAAI latency statistics

7.1 Monitoring VAAI with NFS

Unfortunately, ESXTOP will not show the user VAAI NFS activity. It is necessary to tail the vpxa log file, searching for “Vstorage” entries. Figure 60 has a sample output of an environment during a clone.



```
dlqa0054.lss.emc.com - PuTTY
~ # tail -f /var/log/vpxa.log | grep -i Vstorage
2015-02-27T17:00:53.141Z [FFFA21A0 info 'Libs' opID=9F570797-000008A8-90-b4-91] EMCNasPlugin: Vst
orage task is in-progress...
2015-02-27T17:00:58.143Z [FFFA21A0 info 'Libs' opID=9F570797-000008A8-90-b4-91] EMCNasPlugin: Vst
orage task is in-progress...
2015-02-27T17:01:03.146Z [FFFA21A0 info 'Libs' opID=9F570797-000008A8-90-b4-91] EMCNasPlugin: Vst
orage task is in-progress...
2015-02-27T17:01:33.162Z [FFFA21A0 info 'Libs' opID=9F570797-000008A8-90-b4-91] EMCNasPlugin: Vst
orage task is in-progress...
2015-02-27T17:02:38.193Z [FFFA21A0 info 'Libs' opID=9F570797-000008A8-90-b4-91] EMCNasPlugin: Vst
orage task is in-progress...
2015-02-27T17:02:43.196Z [FFFA21A0 info 'Libs' opID=9F570797-000008A8-90-b4-91] EMCNasPlugin: Vst
orage task completed successfully
```

Figure 60. Monitoring VAAI commands on NFS

8 Performance

In the vast majority of customer environments, the VAAI primitives provide the benefit they intend by offloading tasks to the array with little performance impact, completing the tasks far quicker than VMware could. As noted above in the discussion around XCOPY and SRDF, however, there are scenarios where they can have an unintended impact to the environment. This is most common in arrays which are already under heavy stress and the extra commands from VAAI prove too much to process in an acceptable time for the customer. In these rare cases, it is likely that more than one VAAI primitive is being issued, for example WRITE SAME and UNMAP or XCOPY and UNMAP. When SRDF is part of the configuration the issue can be compounded because SRDF has to break down commands before sending them to the remote site. When the issue does occur, some customers choose to disable one or more primitives, reverting to VMware's software solution. The UNMAP primitive is the most logical to disable both because it is done at the datastore level, and because manual UNMAP can still be run and executed during low activity which will minimize the impact. Conversely, the other primitives must be disabled on each individual ESXi host and cannot be run manually.

8.1 VDI

VDI environments which are highly dynamic, built up and torn down frequently, if not daily, can be susceptible to VAAI performance impacts. This is most common when deploying to a single large datastore with automated UNMAP enabled. Deploying VMs to the datastore issues significant XCOPY commands for the copy and then WRITE SAME commands to allocate new extents. When those VMs are torn down all at once, VMware sends the UNMAP commands to reclaim the storage which will include more WRITE SAME commands. For some customers, the impact will not be a problem given the array's resources, but for others it can be significant. For those latter customers steps can be taken to mitigate the issue. By sizing the datastore closer to the actual usage of the desktops and disabling automated UNMAP, this ensures that both WRITE SAME and UNMAP will be issued only sparingly since VMware will reuse existing extents and over time the array will not need to allocate new ones for the device. The datastore can be resized on-the-fly if storage needs increase. Although XCOPY could also be disabled, it is incredibly beneficial in VDI deployments and unlikely to cause issues with the WRITE SAME primitive neutered.

9 Conclusion

Utilizing VMware's Storage API integrations which permit the offloading of specific VMware operations to the Dell PowerMax benefiting overall system performance and efficiency. The VMware Storage APIs for Array Integration are: Full Copy, Block Zero, hardware-assisted locking, and Dead Space Reclamation. In supported VMware and PowerMax configurations, all these primitives are enabled by default and will be utilized on the PowerMax without any user configuration or intervention.

A Technical support and resources

[Dell.com/support](https://dell.com/support) is focused on meeting customer needs with proven services and support.

A.1 Dell

- The Dell PowerMax and VMware vSphere Configuration Guide
<https://infohub.delltechnologies.com/t/dell-powermax-and-vmware-vsphere-configuration-guide-1/>
- Using VMware vSphere Virtual Volumes and VASA with Dell PowerMax
<https://www.delltechnologies.com/asset/en-us/solutions/infrastructure-solutions/industry-market/h19812-vmware-vvol-vasa-powermax-whitepaper.pdf>
- Unisphere for PowerMax
https://support.emc.com/products/44740_Unisphere-for-PowerMax/Documentation/

A.2 VMware

- VMware vSphere
<https://docs.vmware.com/>