

USING VMWARE VSPHERE STORAGE APIs FOR ARRAY INTEGRATION WITH EMC VMAX

Increasing operational efficiency with VMware and
EMC VMAX

Abstract

This white paper discusses how VMware's vSphere Storage APIs for Array Integration, also known as VAAI, can be used to offload perform various virtual machine operations on the EMC® VMAX®.

September 2017

VMAX Engineering

Copyright © 2017 EMC Corporation. All Rights Reserved.

EMC believes the information in this publication is accurate of its publication date. The information is subject to change without notice.

The information in this publication is provided “as is.” EMC Corporation 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 EMC software described in this publication requires an applicable software license.

VMware, ESXi, vMotion, and vSphere are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions. All other trademarks used herein are the property of their respective owners.

EMC is now part of the Dell group of companies.

Part Number h8115.15

Table of Contents

Executive summary	5
Audience.....	7
VAAI primitives	7
Hardware-accelerated Full Copy or XCOPY	7
Hardware-accelerated Block Zero	8
Hardware-assisted locking	10
ATS heartbeat.....	12
UNMAP.....	14
Datastore	14
Guest OS.....	15
VAAI and EMC Engenuity/HYPERMAX OS version support	16
Compression	19
Viewing VAAI support	19
Enabling the Storage APIs for Array Integration	23
Full Copy Tuning	30
vSphere 4.1 – 6.x	30
vSphere 6.x	31
Thin vmdks.....	34
SRDF and XCOPY	35
eNAS and VAAI	36
Extended stats	37
Nested clones	38
Hardware-accelerated Full Copy	39
Use case configuration	39
Use Case 1: Deploying a virtual machine from a template.....	40
Use Case 2: Cloning hot and cold virtual machines.....	41
Use Case 3: Creating simultaneous multiple clones.....	44
Use Case 4: Storage vMotion.....	45
Server resources – impact to CPU and memory.....	46
Caveats for using hardware-accelerated Full Copy	47
SRDF/Metro specific.....	48
Compression specific	49
Hardware-accelerated Block Zero	49
Use Case 1: Deploying fully allocated virtual machines	49
Use Case 2: Benefits of Block Zero when using zeroedthick virtual disks.....	52

Server resources – impact to CPU and memory	55
Caveats for using hardware-accelerated Block Zero	55
UNMAP	56
Manual UNMAP after Storage vMotion	56
Differences in ESXi 5.5+	60
ESXi 5.5 P3+ and ESXi 6.x.....	62
Using UNMAP with devices over 2 TB in ESXi 5.0 and 5.1.....	63
Use case configuration	64
VMAX (V2)	64
VMAX3/VMAX All Flash (V3)	65
Use Case 1: UNMAP baseline on a standard thin device	65
Use Case 2: UNMAP with SRDF devices.....	66
Use Case 3: UNMAP with TimeFinder/Clone device.....	67
Use Case 4: UNMAP with device under high FA load	68
Caveats for using UNMAP	70
Monitoring VAAI operations with ESXTOP	72
Monitoring VAAI with NFS	76
Conclusion	77
References	77
EMC.....	77
VMware	78

Executive summary

VMware vSphere Storage APIs is a family of APIs used by third-party hardware, software, and storage providers, such as EMC, 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). EMC currently offers four storage integrations as part of VAAI. Three of these integrations are available beginning with Enginuity software version 5875 on the VMAX and HYPERMAX OS[®] software version 5977 on the VMAX3[®]/VMAX All Flash running with VMware vSphere™ 4.1. The fourth integration, Thin Provisioning Block Space Reclamation (UNMAP), is available with Enginuity 5876 2012 Q4 SR (5876.159.102) on the VMAX and HYPERMAX OS software version 5977 on the VMAX3/VMAX All Flash running with VMware vSphere 5.0 U1 and higher. VAAI, by default, provides the ability to offload specific storage operations to the EMC VMAX to increase both overall system performance and efficiency. The four main VMware's Storage APIs supported by VMAX¹ are:

- **Full Copy**². This feature delivers hardware-accelerated copying of data by performing all duplication and migration operations on the array. Customers can achieve considerably faster data movement via VMware Storage vMotion[®], and virtual machine creation and deployment from templates and virtual machine cloning. Full Copy is also referred to by the SCSI command that is issued for it, XCOPY.
- **Block Zero**. This feature delivers hardware-accelerated zero initialization, greatly 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 will be discussed 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. Until vSphere 6.5, UNMAP does not occur automatically - it is performed at the VMFS level as a manual process. In versions 5.0 U1 and 5.1, this is accomplished using vmkfstools. Starting in ESXi 5.5, the vmkfstools command for UNMAP has been deprecated. VMware has introduced a new command to the esxcli command framework to unmap storage. The new command is “unmap” and is under the esxcli storage vmfs namespace. In vSphere 6.0 Guest OS UNMAP was introduced which reclaims

¹ The term “VMAX” will be used in this paper to refer to VMAX, VMAX3 and VMAX All Flash platforms; however where required specific distinctions between the arrays will be noted.

² Full Copy is supported on the VMAX3 platform beginning with the 5977 2014 Q4 SR (5977.498.472) release.

storage within a thin vmdk, and beginning in vSphere 6.5 automated UNMAP at the VMFS level is available.

In addition to the four main APIs, VAAI includes an API for Thin Provisioning. Thin Provisioning is enabled by default in ESXi 5 and higher and Enginuity 5875 and higher (includes all HYPERMAX OS versions). Thin Provisioning permits a couple capabilities. First, the API allows ESXi to tell the array when space can be reclaimed 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 on the array allowing the VM to “resume”.

To determine whether a particular device supports the Thin Provisioning API, query it using the `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 either “no” or “unknown”. In the screen below, Figure 1, a VMAX supported device is shown on the left, while an unsupported, local device is on the right.

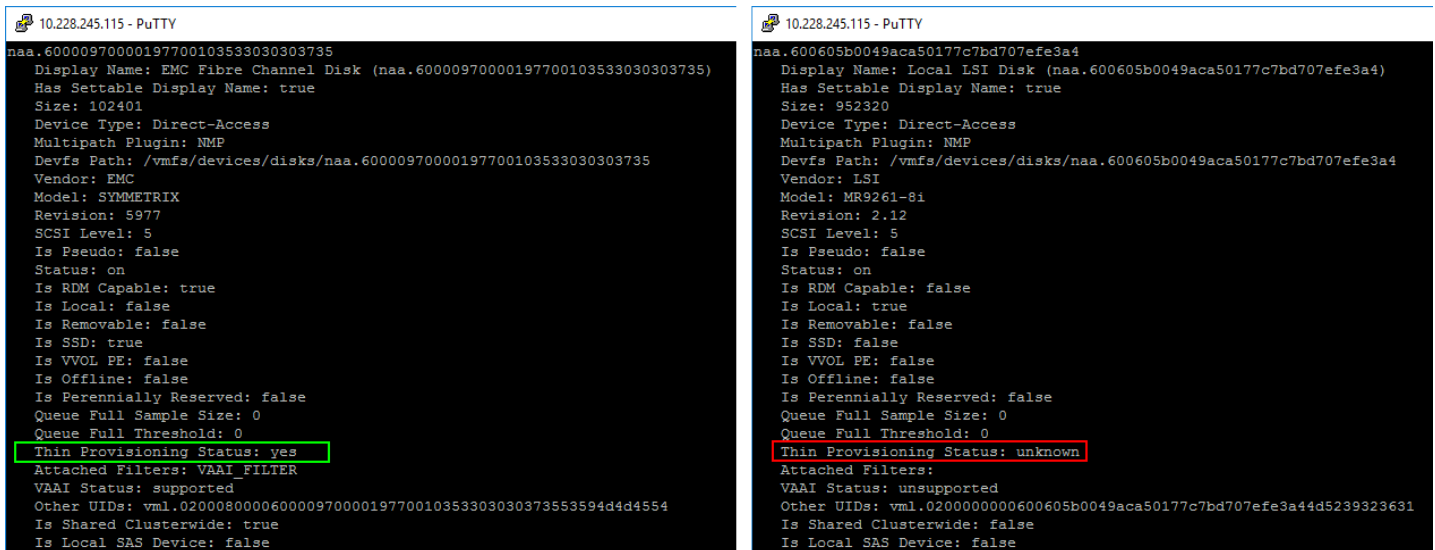


Figure 1. Thin Provisioning Status

When running the supported versions of Enginuity/HYPERMAX OS and vSphere, VAAI functionality is enabled by default on any of the VMAX family of arrays.³

This paper will discuss the operation of and best use cases for utilizing VAAI features with the VMAX. In doing so, it will demonstrate how to maximize efficiency and

³ Prior to vSphere 6.5, UNMAP will require running `vmkfstools` or `esxcli storage vmfs unmap` manually, depending on the ESXi version.

increase the effectiveness of a VMware vSphere 5.x or 6.x environment deployed on an EMC VMAX.

This paper covers Enginuity 5875 and later and HYPERMAX OS 2017 Q2 release and earlier.

Audience

This technical white paper is intended for VMware administrators and storage administrators responsible for deploying VMware vSphere 4.1 – 6.x, and the included VAAI features, on VMAX running Enginuity 5875 and later or on VMAX3/VMAX All Flash running HYPERMAX OS 5977 and later.

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 by EMC VMAX running Enginuity 5875 and later and on the VMAX3/VMAX All Flash running HYPERMAX OS 5977 and later. Beginning with the vSphere 4.1 release, this array offload capability supports three primitives: hardware-accelerated Full Copy², hardware-accelerated Block Zero, and hardware-assisted locking. Beginning with vSphere 5.0 U1 and Enginuity version 5876.159.102 and HYPERMAX OS version 5977, the UNMAP primitive is supported.

Hardware-accelerated Full Copy or XCOPY

The time it takes to deploy or migrate a virtual machine will be greatly 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 greatly reduces 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 VMAX. When using the VMAX, 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 VMAX has received all the information it needs to perform the task. The array, using the same mechanism as the EMC TimeFinder local replication software, then completes the transfer of data asynchronously. 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.

In order to ensure that the Full Copy process does not over-utilize VMAX 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. On the VMAX3/VMAX

All Flash in particular, reaching these limits are very unlikely; however if it does occur, vSphere seamlessly switches to software copy in mid-operation. VMware vSphere will retry the XCOPY command after a short 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.

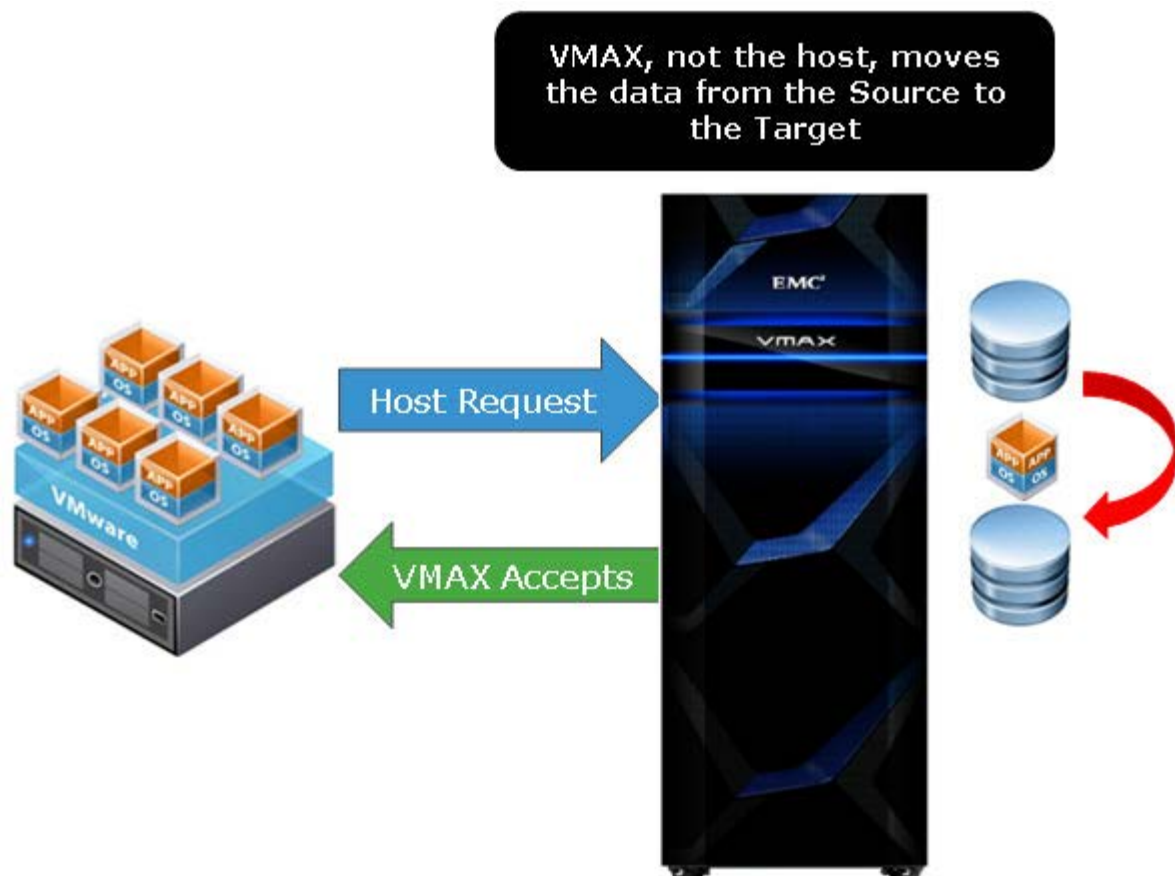


Figure 2. Hardware-accelerated Full Copy

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 VMAX 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 a very large disk 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.

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 on VMAX Virtual Provisioning when deploying “eagerzeroedthick” virtual disks or writing to unallocated sectors of a “zeroedthick” or “thin” virtual disk. When enabled, the VMAX will not actually write the zeros to disk but instead simply allocate the relevant tracks on the thin pool (if not already allocated) and then set the **Never Written By Host** flag on them. 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 copy.

On the VMAX3/VMAX All Flash, however, the architecture is different and no space is actually reserved. This is because the VMAX3/VMAX All Flash 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 array runs out of space. It was deemed unnecessary, therefore, to reserve space on the VMAX3/VMAX All Flash when using Block Zero. Note, however, that because of the architecture of the VMAX3/VMAX All Flash there is no performance impact even though the space is not pre-allocated.

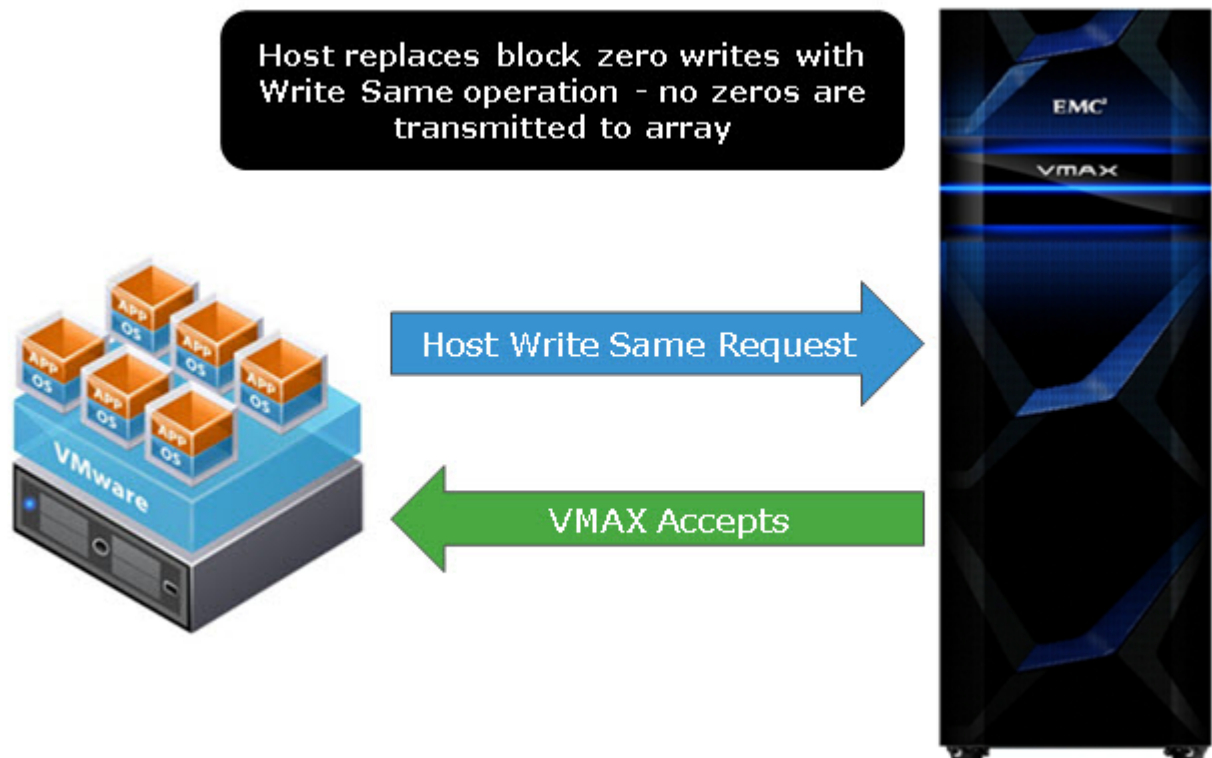


Figure 3. Hardware-accelerated Block Zero

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 4, 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.

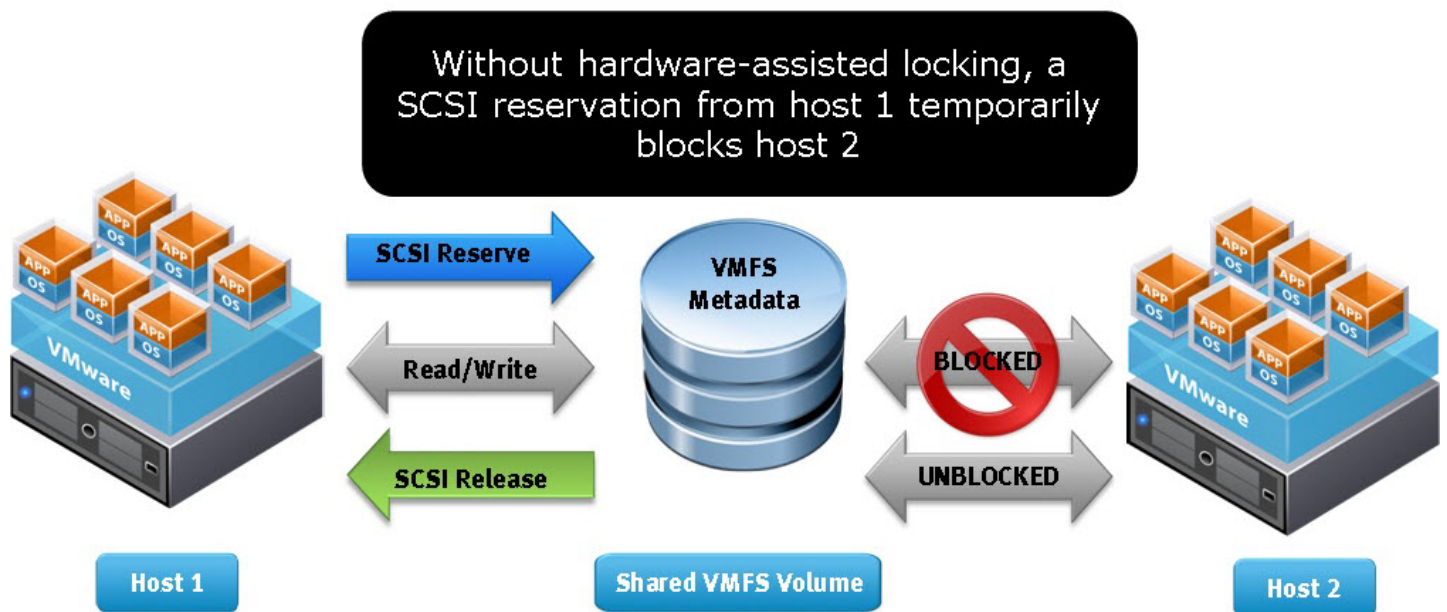


Figure 4. Traditional VMFS locking before vSphere 4.1 and hardware-assisted locking

This led to the development of the third primitive for VAAI in the vSphere 4.1 release, 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 5. 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 hardware-assisted locking, access is based on a single block coherence. 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

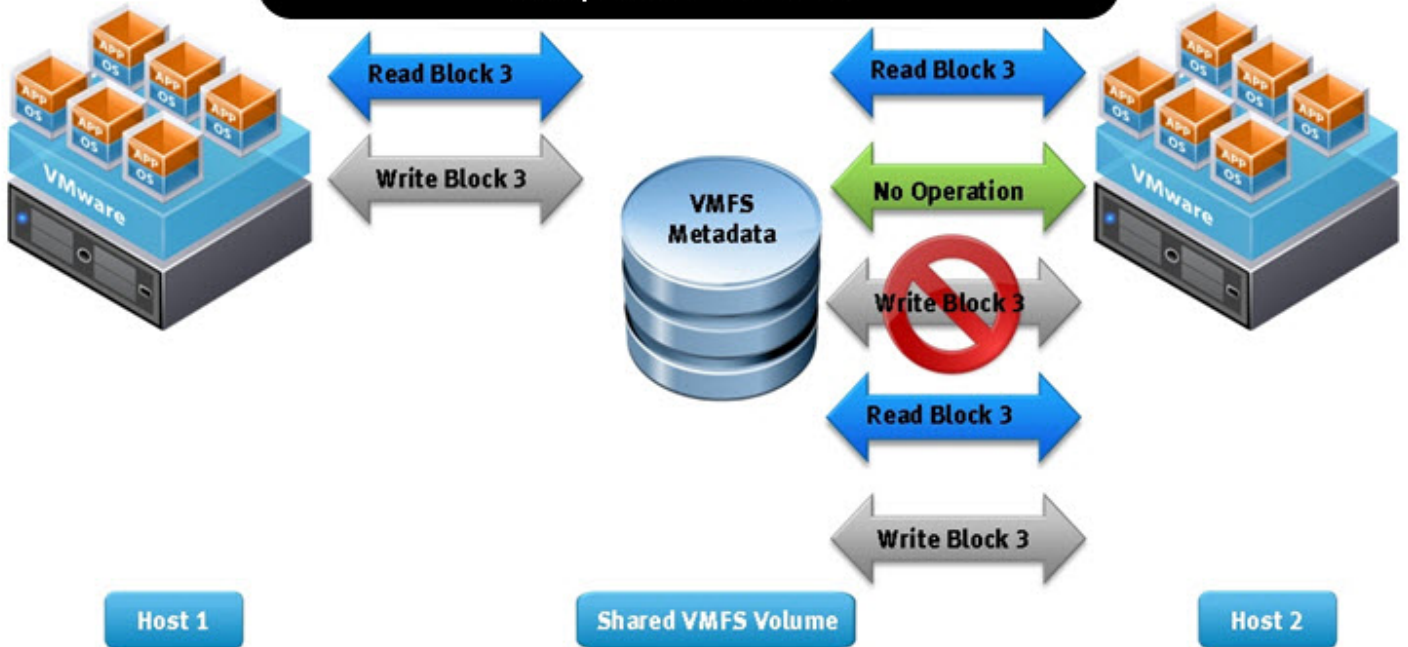


Figure 5. VMFS locking with vSphere 4.1+ and hardware-assisted locking

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.

ATS heartbeat

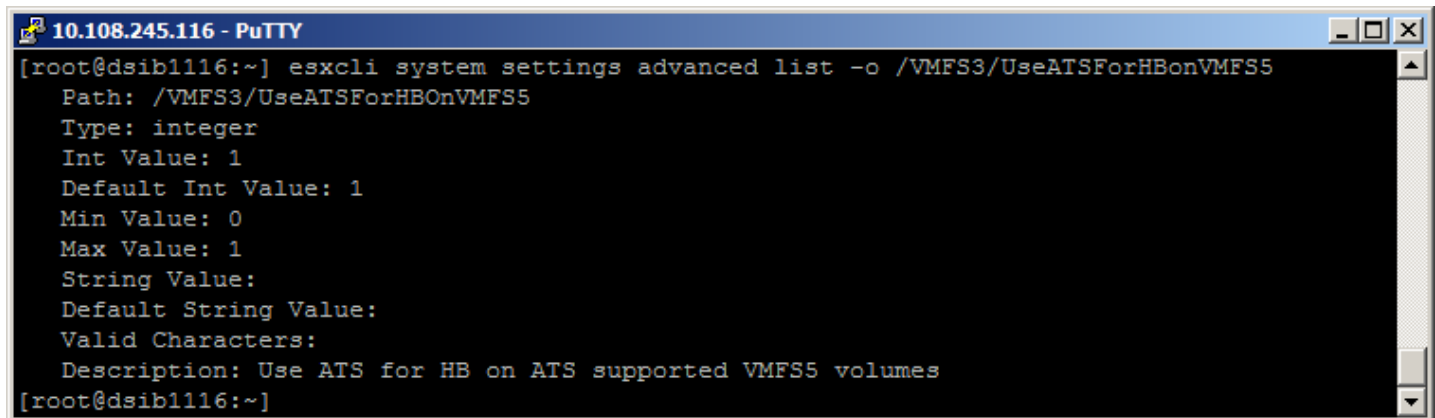
One of the types of locks that VMware performs is called the heartbeat. The heartbeat is used to synchronize operations between multiple hosts as well as check on the vitality of the host. Before ESXi 5.5 U2, VMware used SCSI read and writes to implement the heartbeat, even on storage arrays that supported VAAI (ATS). Beginning with ESXi 5.5 U2, however, VMware defaults 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, in this case the VMAX. Under certain load conditions, ATS may fail with a false ATS miscompare.⁴ 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

⁴ VMware KB article 2113956. Although the KB refers to an IBM storage system, the error can occur on any storage platform.

be found in the `/var/log/vmkernel.log` file. In the unlikely event that this error is recurring in the environment, VMware and EMC recommend disabling ATS for the heartbeat mechanism only, which means ESXi will revert to the non-ATS functionality, or SCSI reads and writes. 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 6:

```
esxcli system settings advanced list -o /VMFS3/UseATSForHBOnVMFS5
```



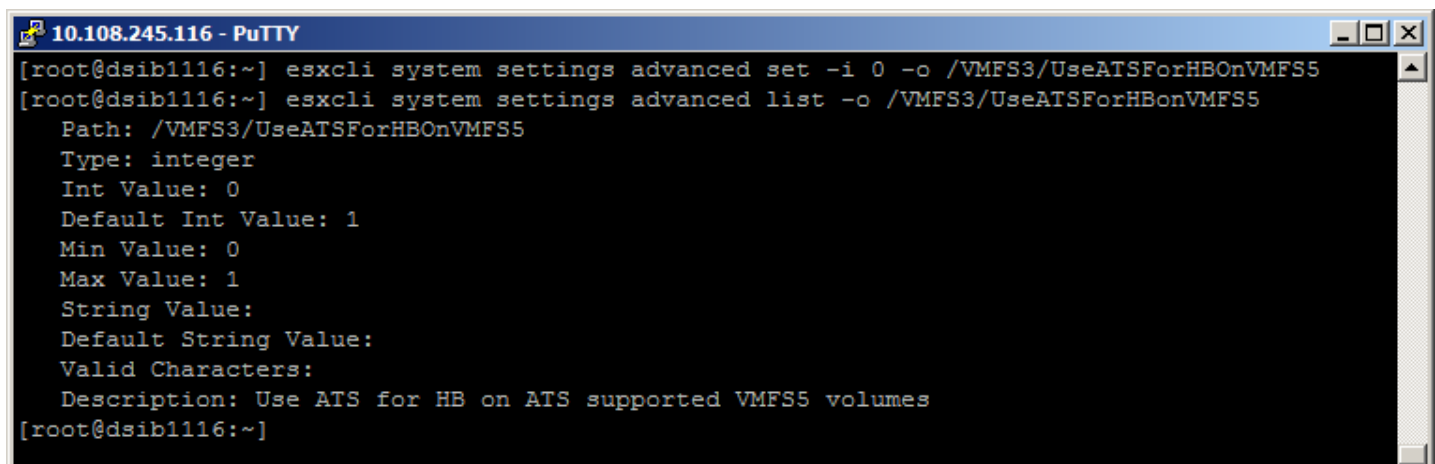
```
10.108.245.116 - PuTTY
[root@dsib1116:~] esxcli system settings advanced list -o /VMFS3/UseATSForHBOnVMFS5
Path: /VMFS3/UseATSForHBOnVMFS5
Type: integer
Int Value: 1
Default Int Value: 1
Min Value: 0
Max Value: 1
String Value:
Default String Value:
Valid Characters:
Description: Use ATS for HB on ATS supported VMFS5 volumes
[root@dsib1116:~]
```

Figure 6. ATS heartbeat setting

As noted in Figure 6, 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 is shown in Figure 7.



```
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 7. Disabling ATS heartbeat setting

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. VMware indicates there should be no performance impact by disabling the ATS heartbeat. It is critical to understand that the ATS heartbeat is a subset of functionality to ATS and that the ATS primitive (*DataMover.HardwareAcceleratedInit*) itself should never be disabled.

vSphere 6.5

Beginning in vSphere 6.5, VMware has changed how it reacts to an ATS mismatch. Instead of aborting the IOs and assuming the ATS mismatch is legitimate, VMware will retry the read to determine if the mismatch is accurate. If it is accurate, the previous process applies; however if the on-disk heartbeat data has not changed, meaning the mismatch was false, VMware will retry the heartbeat after some milliseconds.

VMware has also corrected another mismatch event where it was not able to determine if a heartbeat made it to disk. VMware now uses an earlier heartbeat record to determine the validity of the ATS mismatch and then proceeds appropriately. The new vSphere 6.5 functionality works in both VMFS 5 and VMFS 6 and negates the need to ever disable the ATS heartbeat. If upgrading from ESXi 5.5 U2 or higher where the heartbeat was disabled, be sure to re-enable when reaching ESXi 6.5.

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 automatically⁵. This capability has been available in one or both forms since vSphere 5.0 U1. 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, and was introduced in vSphere 6.0 with Windows 2012 R2 and expanded to Linux in vSphere 6.5. It supports both VMFS and VVol datastores.

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 VMAX, 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 VMAX 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 **Never Written By Host (NWBH)** on the VMAX but the extent cannot be deallocated. This is still beneficial, however, as those tracks do not have to be

⁵ UNMAP in this scenario does not support VVol datastores as a VVol datastore 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.

retrieved from disk should a read request be performed. Instead, the VMAX array sees the NWBH flag and immediately returns all zeros. The VMAX3/VMAX All Flash performs a similar process.

VMware will, of course, reuse existing space in the datastore even if the data has not been reclaimed on the array; however it may not be able to use all of the space for a variety of reasons. In addition, by reclaiming the space in the thin pool 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.

Guest OS

Guest OS or in-guest UNMAP was introduced in vSphere 6.0. In the initial release it supported Windows 2012 R2 and higher but not Linux. In vSphere 6.5 SPC-4 support was added which enables Guest OS UNMAP for Linux VMs. There are a number of prerequisites that must be met to take advantage of Guest OS UNMAP. There are some differences between the two vSphere versions that should be noted. They are covered in the next two sections.

vSphere 6.0 and Windows 2012 R2+

To enable Guest OS UNMAP in vSphere 6.0, the prerequisites are as follows:

- ESXi 6.0+
- Thin virtual disk (vmdk)
- VM hardware version 11+
- 64k allocation unit (recommended) for NTFS⁶
- The advanced parameter **EnableBlockDelete** must be set to 1 if using VMFS (VVols do not require the parameter). It is off by default.

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

An important note is that change block tracking (CBT) is not supported with Guest OS UNMAP in vSphere 6.0 but is supported in vSphere 6.5.

vSphere 6.5 and Linux

To enable Guest OS UNMAP in vSphere 6.5, the prerequisites are as follows:

- ESXi 6.5+
- Thin virtual disk (vmdk)
- VM hardware version 13+
- A Linux OS and file system that supports UNMAP

⁶ A VMware bug exists which can limit the amount of space reclaimed if the allocation unit is not 64k. If using Guest OS UNMAP, applying the patch which resolves the bug is recommended. See VMware KB 2148987 for detail.

- The advanced parameter **EnableBlockDelete** must be set to 1 if using VMFS (VVols do not require the parameter). It is off by default.
 - o `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 (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
```

VAAI and EMC Engenuity/HYPERMAX OS version support

EMC's implementation of VAAI is done within the Engenuity and HYPERMAX OS software. As with any software, bugs are an inevitable byproduct. While EMC 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 our customers might experience with VAAI, the following table, Table 1, is provided. It includes all current Engenuity and HYPERMAX OS releases and any E Packs that must be applied at that level, beginning with the recommended base level of 5875 to the current 5977 release. Using this table will ensure that a customer's Engenuity or HYPERMAX OS release is at the proper level for VAAI interaction.

In August 2016, VMware ended general support for vSphere 5.0 and 5.1. Therefore, starting with the 5977 2016 Q3 SR in the table below, these releases are no longer listed as minimum versions. It is important to remember, however, that VMware allows for customers to purchase extended support for up to 2 years and EMC will support those versions with the HYPERMAX OS code levels if customers choose to do that.

Table 1. Engenuity/HYPERMAX OS and VAAI support

Engenuity/HYPERMAX OS Code Level	E Pack required	VAAI primitives supported	Minimum version of vSphere required	Important Notes
5977.1125.1125 (5977 2017 Q2 Release)	No	ATS, Block Zero, Full Copy, UNMAP	vSphere 5.5, vSphere 6.5 for automated UNMAP	Important VMware patch for vSphere 6.5 to fix UNMAP issues - VMware KB 2148987.
5977.952.892 (5977 2016 Q4 SR)	No	ATS, Block Zero, Full Copy, UNMAP	vSphere 5.5, vSphere 6.5 for automated UNMAP	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 5.5, vSphere 6.5 for automated UNMAP	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 5.0 (U1 for UNMAP), vSphere 6.5 for automated UNMAP	All primitives supported with SRDF/Metro except Full Copy.
5977.813.785	No	ATS, Block Zero, Full Copy, UNMAP	vSphere 5.0 (U1 for UNMAP), vSphere 6.5 for automated UNMAP	All primitives supported with SRDF/Metro except Full Copy.
5977.811.784 (5977 2016 Q1 SR)	Recommended	ATS, Block Zero, Full Copy, UNMAP	vSphere 5.0 (U1 for UNMAP), vSphere 6.5 for automated UNMAP	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 5.0 (U1 for UNMAP), vSphere 6.5 for automated UNMAP	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 5.0 (U1 for UNMAP), vSphere 6.5 for automated UNMAP	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 5.0 (U1 for UNMAP), vSphere 6.5 for automated UNMAP	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 5.0 (U1 for UNMAP), vSphere 6.5 for automated UNMAP	VMAX3 GA; No Full Copy support.
5876.288.400	No	ATS, Block Zero, Full Copy, UNMAP	vSphere 4.1, vSphere 5.0 U1 for UNMAP	
5876.286.194	No	ATS, Block Zero, Full Copy, UNMAP	vSphere 4.1, vSphere 5.0 U1 for UNMAP	

5876.268.174	Yes	ATS, Block Zero, Full Copy, UNMAP	vSphere 4.1, vSphere 5.0 U1 for UNMAP	Refer to ETA 184320 for E Pack information.
5876.251.161 (5876 2013 Q3 SR)	Recommended	ATS, Block Zero, Full Copy, UNMAP	vSphere 4.1, vSphere 5.0 U1 for UNMAP	Refer to ETA 184320 for E Pack information.
5876.229.145 (5876 2013 Q2 SR)	Recommended	ATS, Block Zero, Full Copy, UNMAP	vSphere 4.1, vSphere 5.0 U1 for UNMAP	Refer to ETA 184320 for E Pack information.
5876.163.105	Yes	ATS, Block Zero, Full Copy, UNMAP	vSphere 4.1, vSphere 5.0 U1 for UNMAP	Refer to ETAs emc319797 and 184320 for E Pack information.
5876.159.102 (5876 2012 Q4 SR)	Yes	ATS, Block Zero, Full Copy, UNMAP	vSphere 4.1, vSphere 5.0 U1 for UNMAP	First Enginuity level with support for the UNMAP primitive. Refer to ETAs emc319797 and 184320 for E Pack information.
5876.85.59	No	ATS, Block Zero, Full Copy	vSphere 4.1	
5876.82.57 (5876 GA)	No	ATS, Block Zero, Full Copy	vSphere 4.1	
5875.300.232	No	ATS, Block Zero, Full Copy	vSphere 4.1	
5875.286.218	No	ATS, Block Zero, Full Copy	vSphere 4.1	
5875.269.201	No	ATS, Block Zero, Full Copy	vSphere 4.1	
5875.267.201	No	ATS, Block Zero, Full Copy	vSphere 4.1	First Enginuity level that supports vSphere 5.1 without E Pack.
5875.249.188	Yes - vSphere 5.1 and higher	ATS, Block Zero, Full Copy	vSphere 4.1	This E Pack described in ETA emc289707 fixes a problem with ATS where a VMFS-5 datastore can fail to create (fix included in 5875.267.201). Note this Enginuity level + E Pack is the minimum version required to use vSphere 5.1.
5875.231.172	No	ATS, Block Zero, Full Copy	vSphere 4.1	Recommended base level for using VAAI.
5875.198.148	Upgrade to 5875.231.172	ATS, Block Zero, Full Copy	vSphere 4.1	If upgrade is not possible refer to ETA emc263675 for E Pack information. EMC does not recommend using VAAI prior to this Enginuity release and E Pack.

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

Compression

Starting with HYPERMAX OS release 5977 2016 Q3 SR, EMC offers compression on VMAX All Flash arrays. Compression is not available on VMAX3 arrays. The VAAI primitives are fully supported with compression and no user interaction is required.

Viewing VAAI support⁷

Through the CLI or within the vSphere Client, one can see whether or not 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 8.

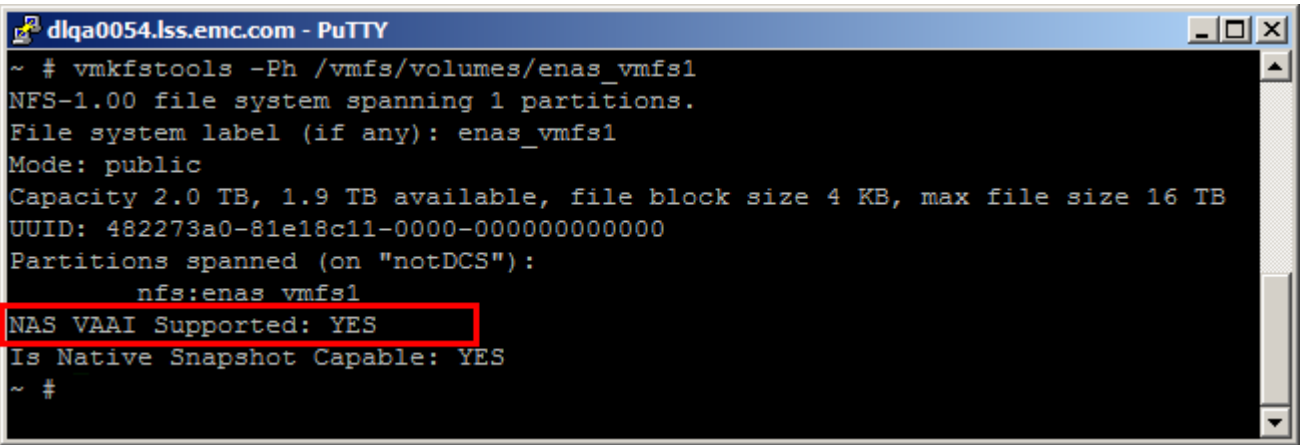
```
10.108.245.203 - PuTTY
~ # esxcli storage core device vaaI status get -d naa.60000970000195900283533030334334
naa.60000970000195900283533030334334
VAAI Plugin Name: VMW_VAAIP_SYMM
ATS Status: supported
Clone Status: supported
Zero Status: supported
Delete Status: unsupported
~ #

10.108.245.115 - PuTTY
~ # esxcli storage core device vaaI status get -d naa.60000970000195900283533030344241
naa.60000970000195900283533030344241
VAAI Plugin Name: VMW_VAAIP_SYMM
ATS Status: supported
Clone Status: supported
Zero Status: supported
Delete Status: supported
~ #
```

Figure 8. Viewing VAAI primitive support through the CLI with and without UNMAP support (Delete Status)

If using eNAS and thus NFS with VMAX3/VMAX All Flash (see section eNAS and VAAI), to view VAAI support run the `vmkfstools` command shown in Figure 9.

⁷ There is no user interface to view VAAI primitive activity on the VMAX. 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”.



```
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 9. Viewing VAAI primitive support for NFS

For the vSphere Client GUI, there are a couple views for VAAI support. From the Configuration tab, under Hardware/Storage, there is a column present in the datastore and device views labeled “Hardware Acceleration”. This column indicates the support status of the primitives for the device or datastore. There are three possible values that can populate this column: “Supported”, “Not supported”, or “Unknown”⁸. This column is shown in Figure 10.

⁸ A particular device or datastore may be labeled as “Unknown” until it is successfully queried.

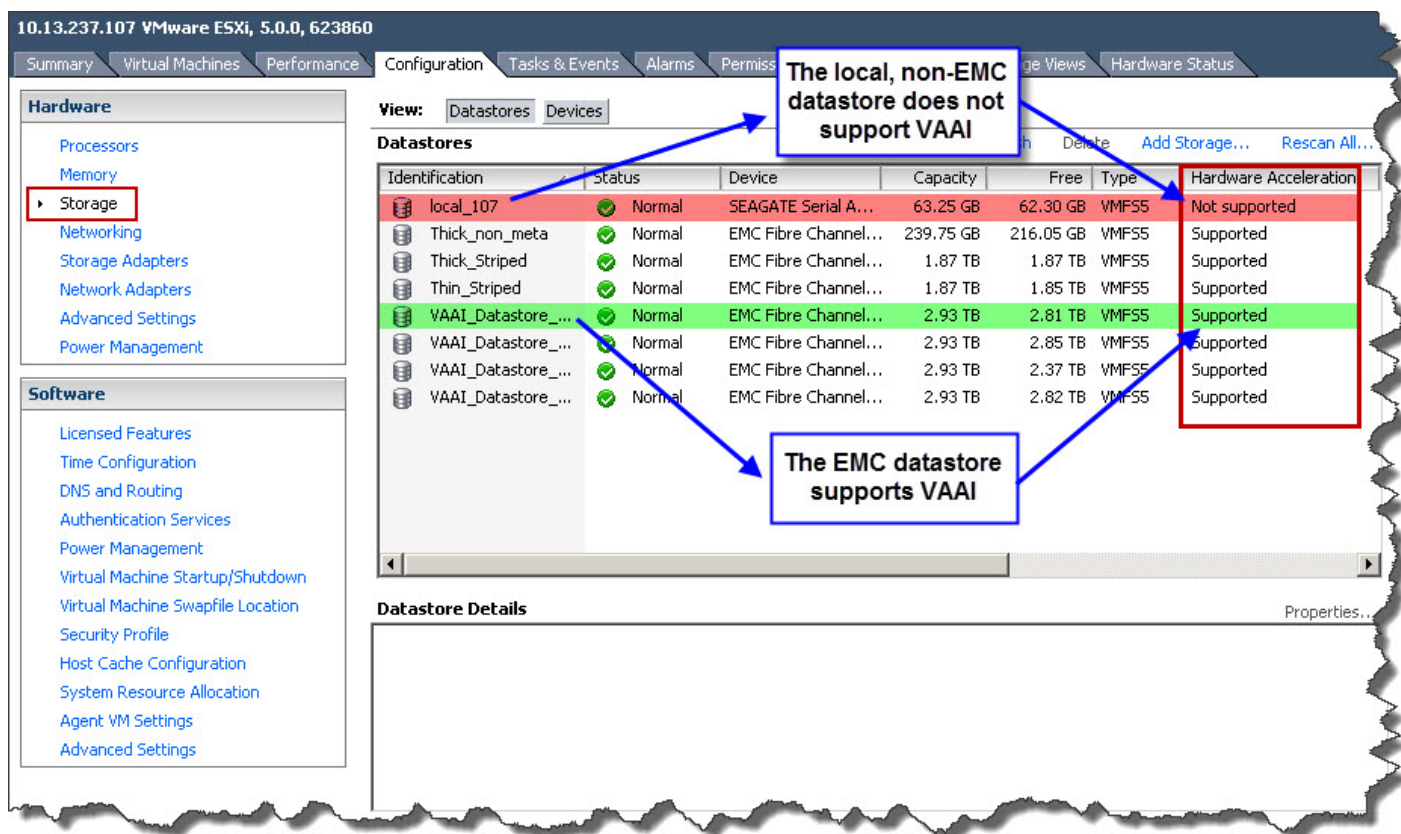


Figure 10. Viewing VAAI support in the vSphere Client

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”.

A similar view is also available using EMC’s Virtual Storage Integrator (VSI) plug-in. This free tool, available to download on support.EMC.com, enables additional capabilities to the vSphere Client so that users may view storage-specific information as seen in Figure 11. VSI will detail all the VMAX information about the device backing the datastore, including the location of the device in storage pool(s) if applicable. There is also the ability to generate performance statistics on the datastore.

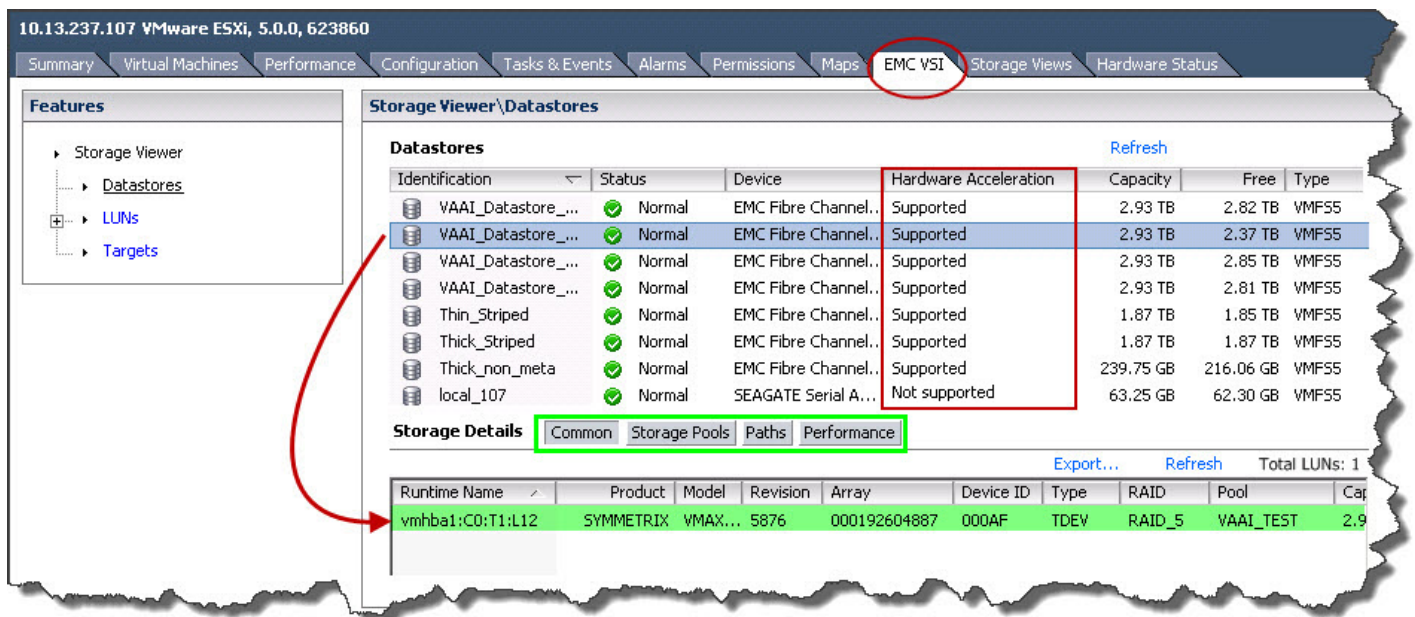


Figure 11. EMC Virtual Storage Integrator

In the vSphere Web Client the Hardware Acceleration column is not available from the host view as it in the vSphere Client. To determine if a particular datastore supports VAAI, one needs to navigate to the individual datastore settings where a Datastore Capabilities area contains this information, shown in Figure 12.

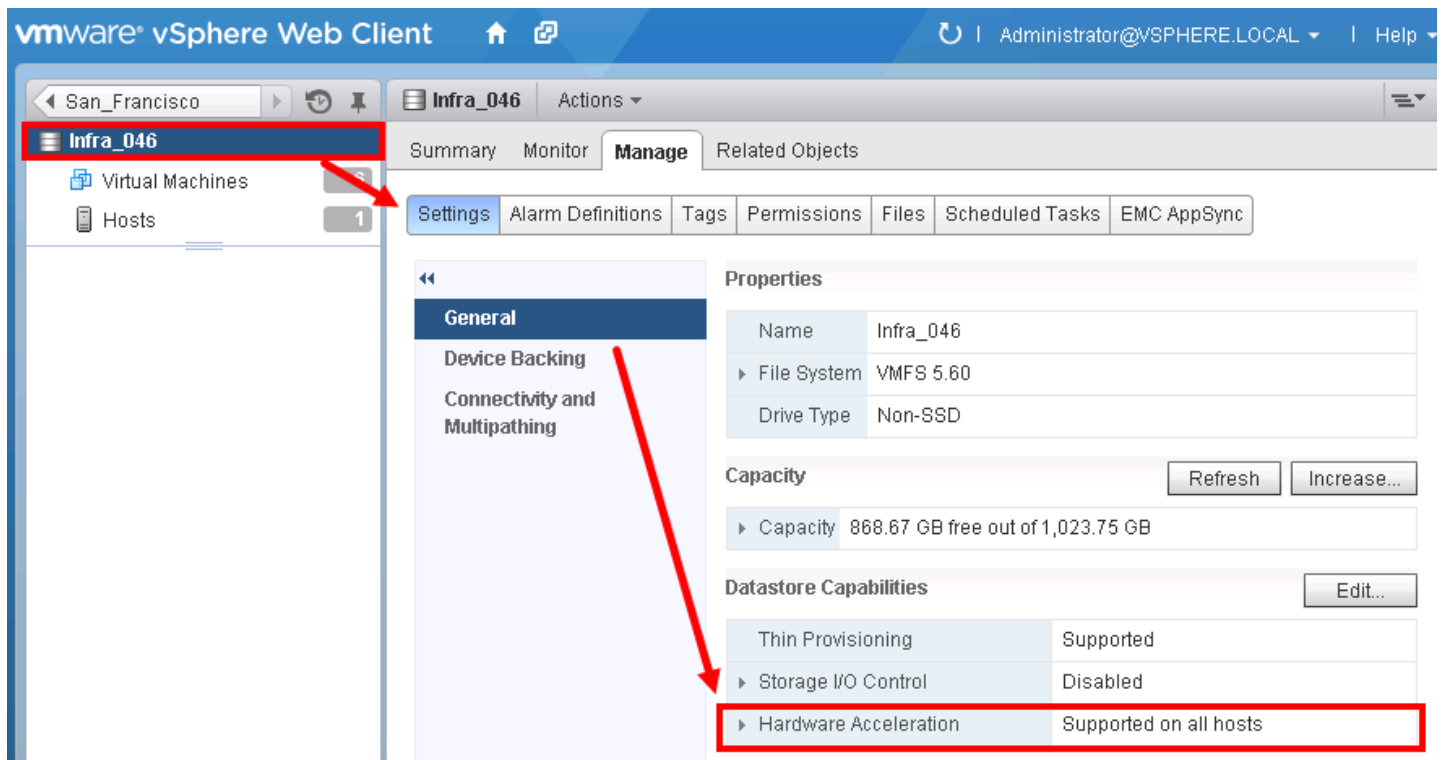


Figure 12. Hardware Acceleration column in vSphere Web Client

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

Enabling the Storage APIs for Array Integration⁹

Full Copy, Block Zero, and Hardware-assisted locking

The VAAI primitives are enabled by default on both the VMAX running 5875 Enginuity or later and the VMAX3/VMAX All Flash running HYPERMAX 5977 or later, used with 4.1 – 6.x ESXi servers (properly licensed¹⁰) and should not require any user intervention.³ 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 13. Under normal circumstances, these settings should not be altered.

⁹ This section on enabling and disabling VAAI does not apply to VAAI with NFS. Once the EMC NAS plug-in is installed, VAAI for NFS is enabled and cannot be disabled using any of the methods described in this paper.

¹⁰ Refer to VMware documentation for required licensing to enable VAAI features.

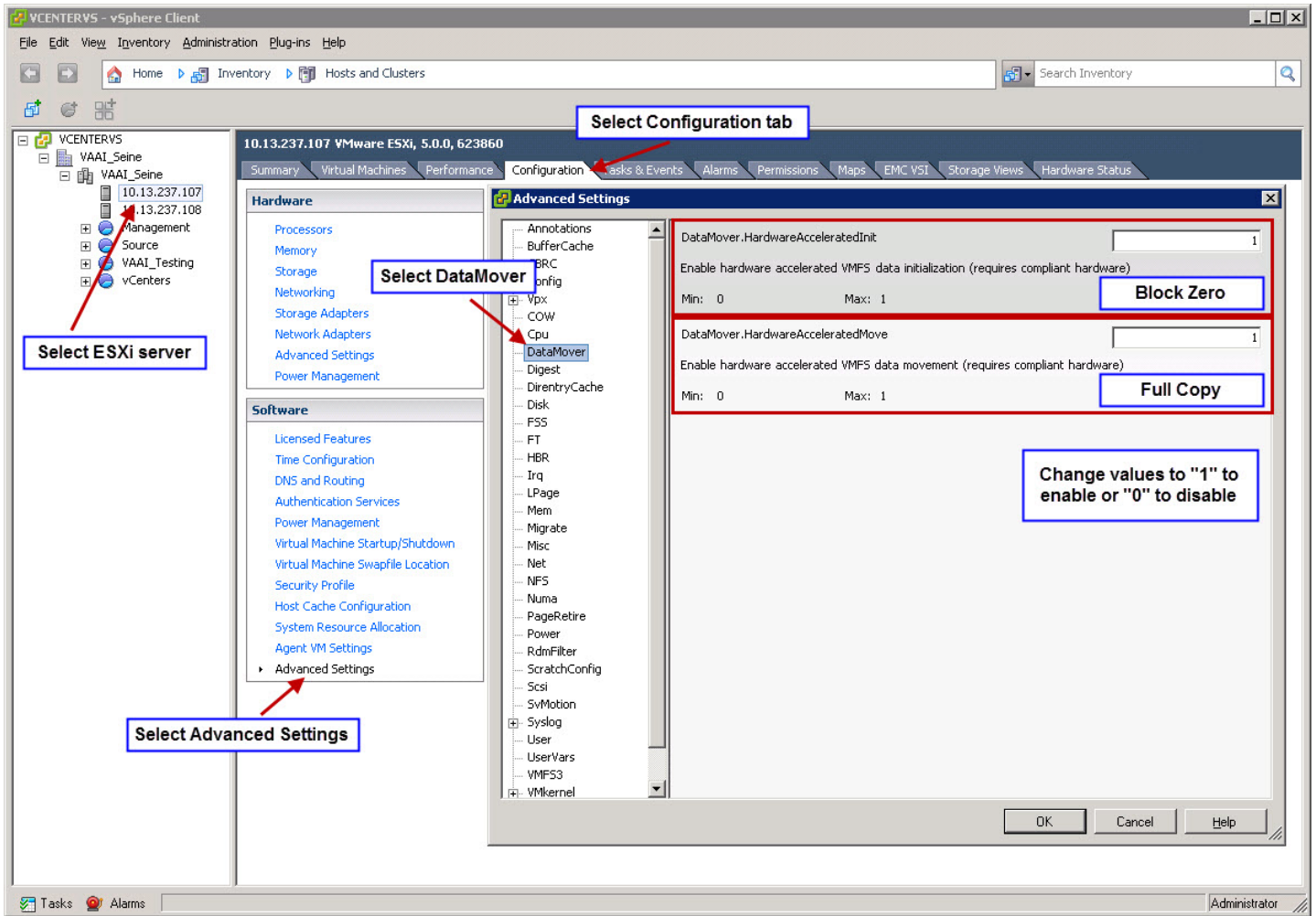


Figure 13. Enabling hardware-accelerated Full Copy or Block Zero on ESXi 5.x

Hardware-assisted locking can be disabled or enabled by changing the setting *VMFS3.HardwareAcceleratedLocking* in the ESXi server advanced settings under VMFS3 shown in the vSphere Client in Figure 14.

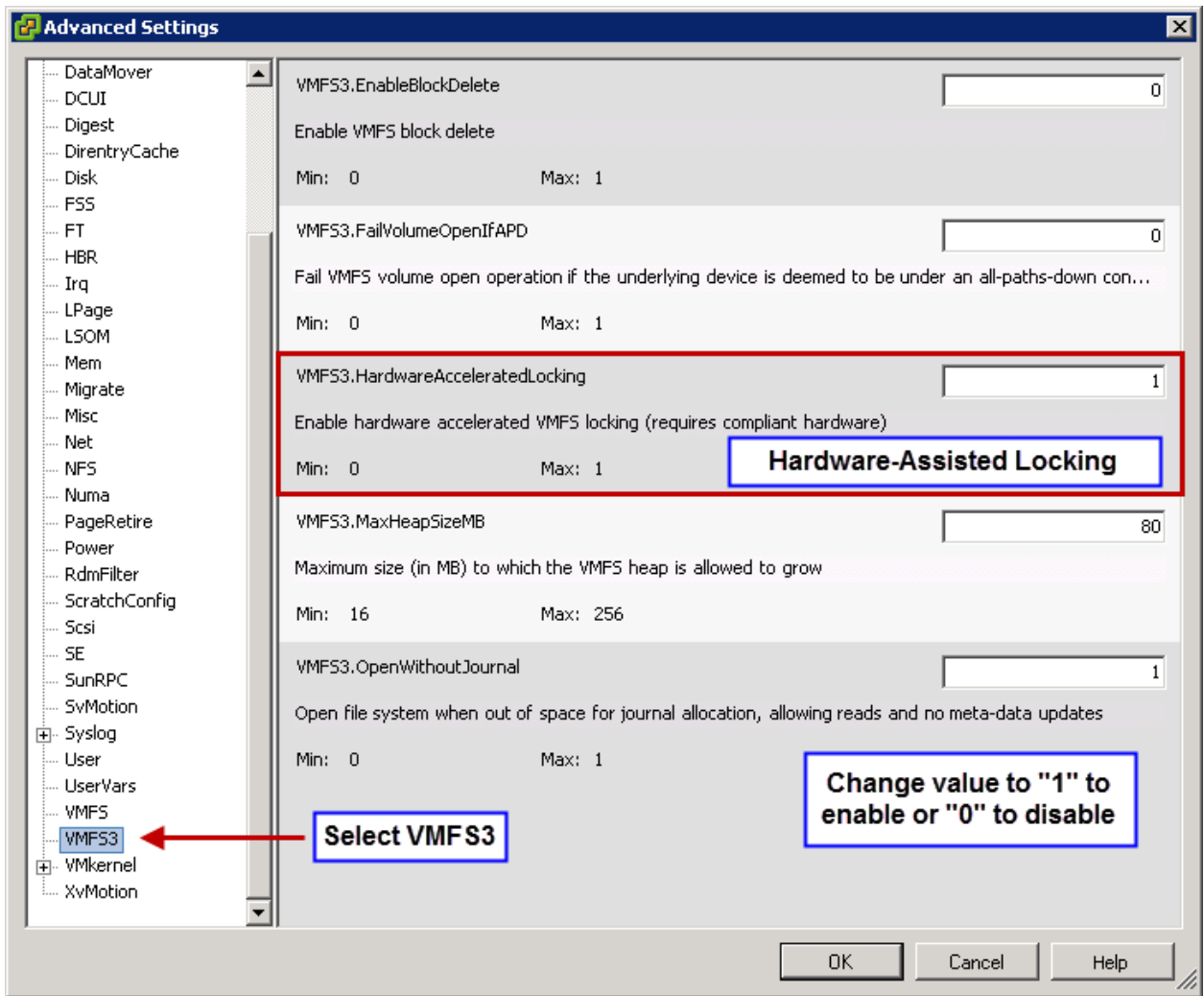


Figure 14. Enabling hardware-assisted locking on ESXi 5.x

In the vSphere Web Client there is a similar navigation to the advanced settings as see in Figure 15.

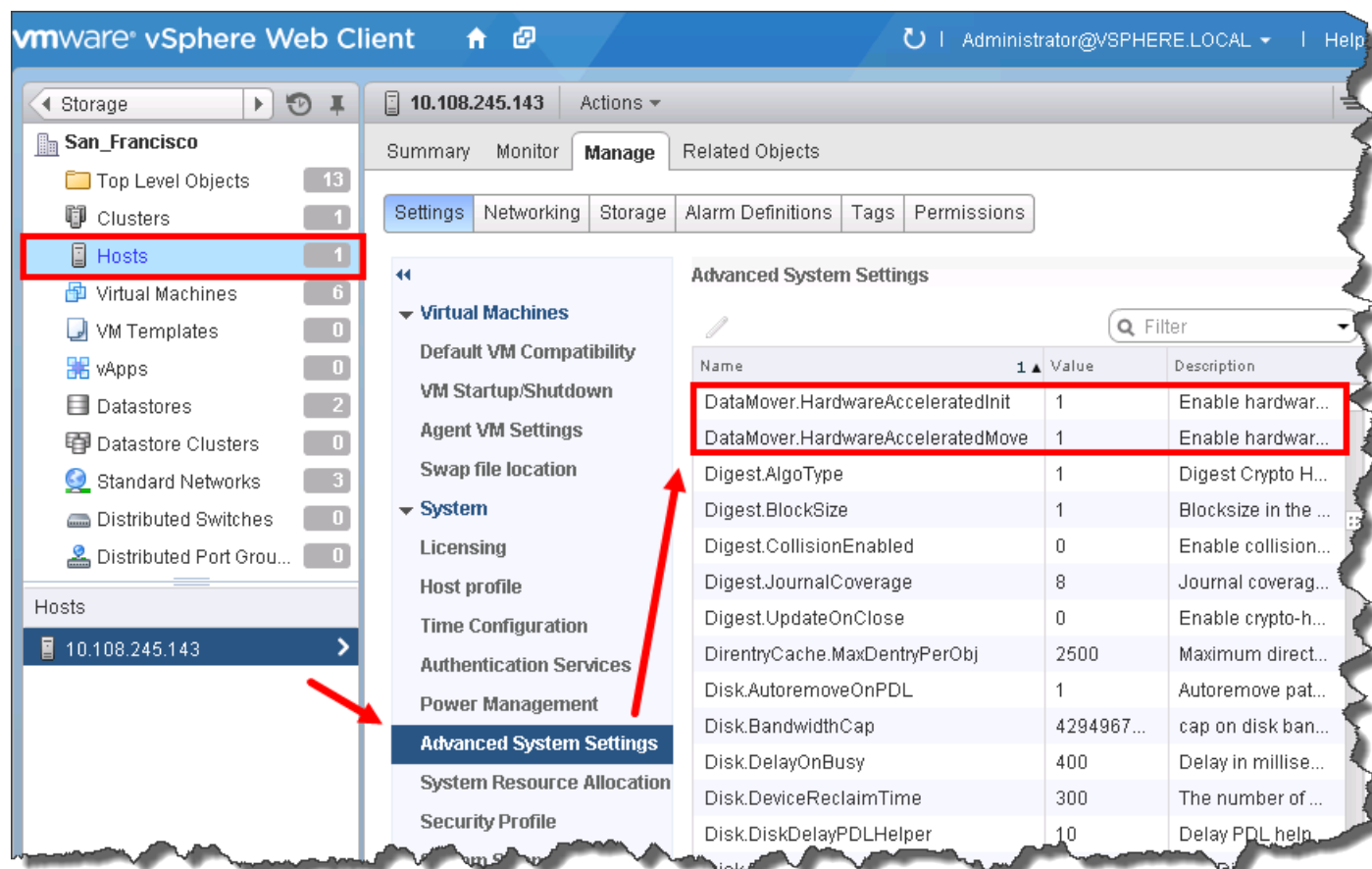


Figure 15. Enabling VAAI in the vSphere Web Client

The new, partial-functionality HTML 5 client is shown in Figure 16 for completeness.

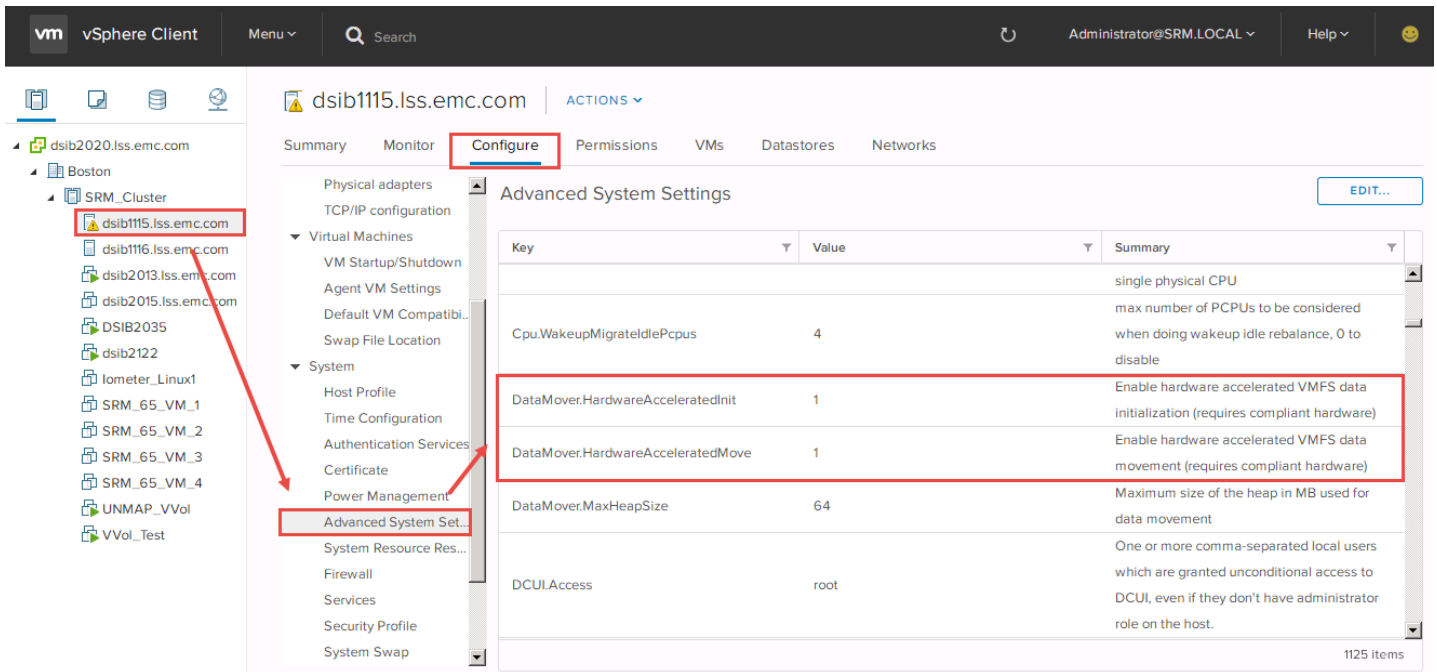


Figure 16. Enabling VAAI in the vSphere HTML 5 Client

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.

Automated UNMAP

Beginning with vSphere 6.5, 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) which keeps track of which LBAs that can be unmapped and then issues those UNMAPs at various intervals. It is an asynchronous process. VMware estimates that if there is dead space in a datastore, automated UNMAP should reclaim it in under 12 hours.

UNMAP is enabled by default when a VMFS 6 datastore is created through a 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 2 values: None (off), and Low. It is set to Low by default. These parameters are seen in Figure 17.

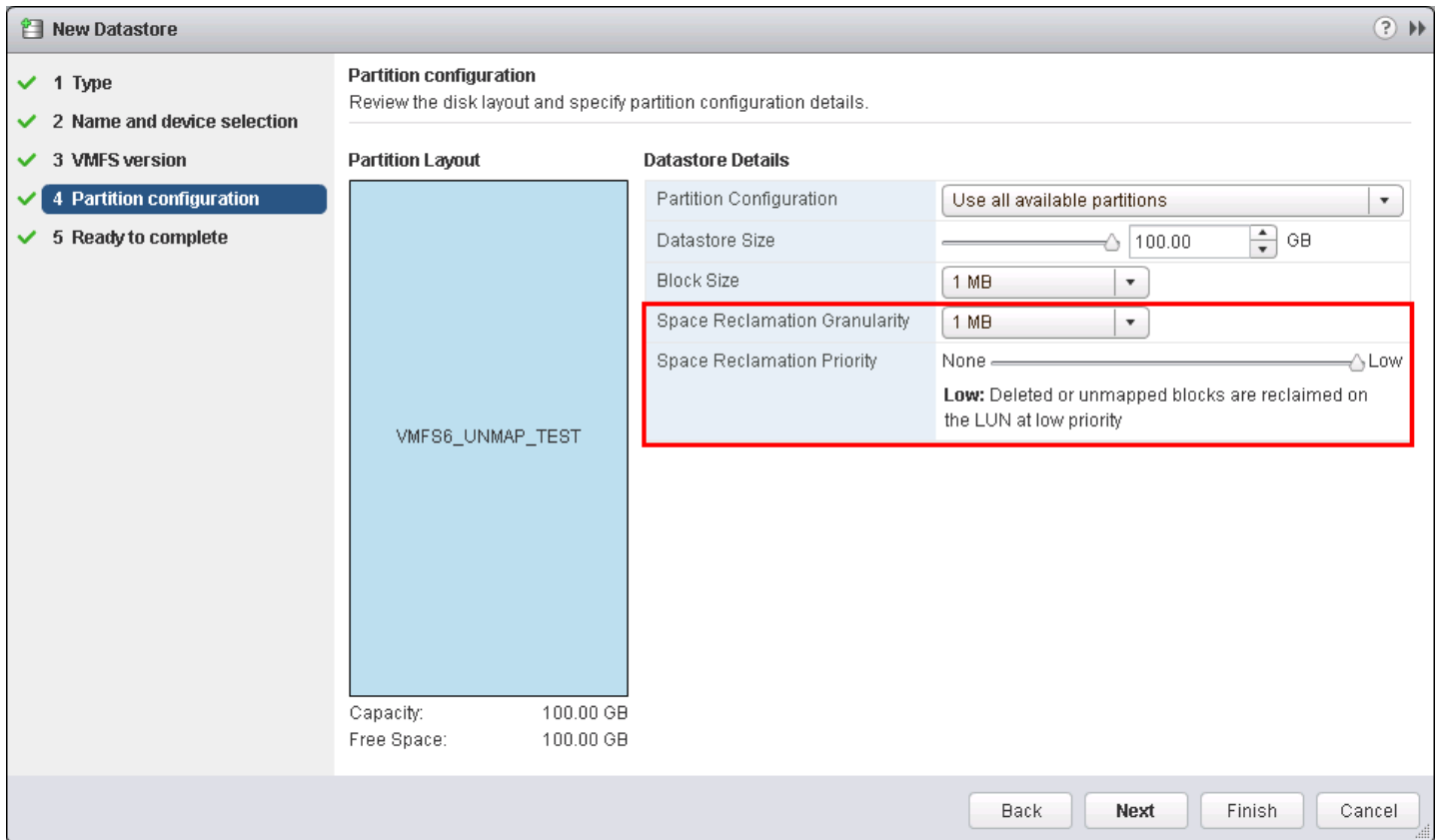


Figure 17. Enabling automated UNMAP on a VMFS 6 datastore

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 Web Client or CLI. Within the datastore view in the Client, navigate to the Configure tab and the General sub-tab. Here find a section on Space Reclamation where the Edit button can be used to make the desired change as in Figure 18.

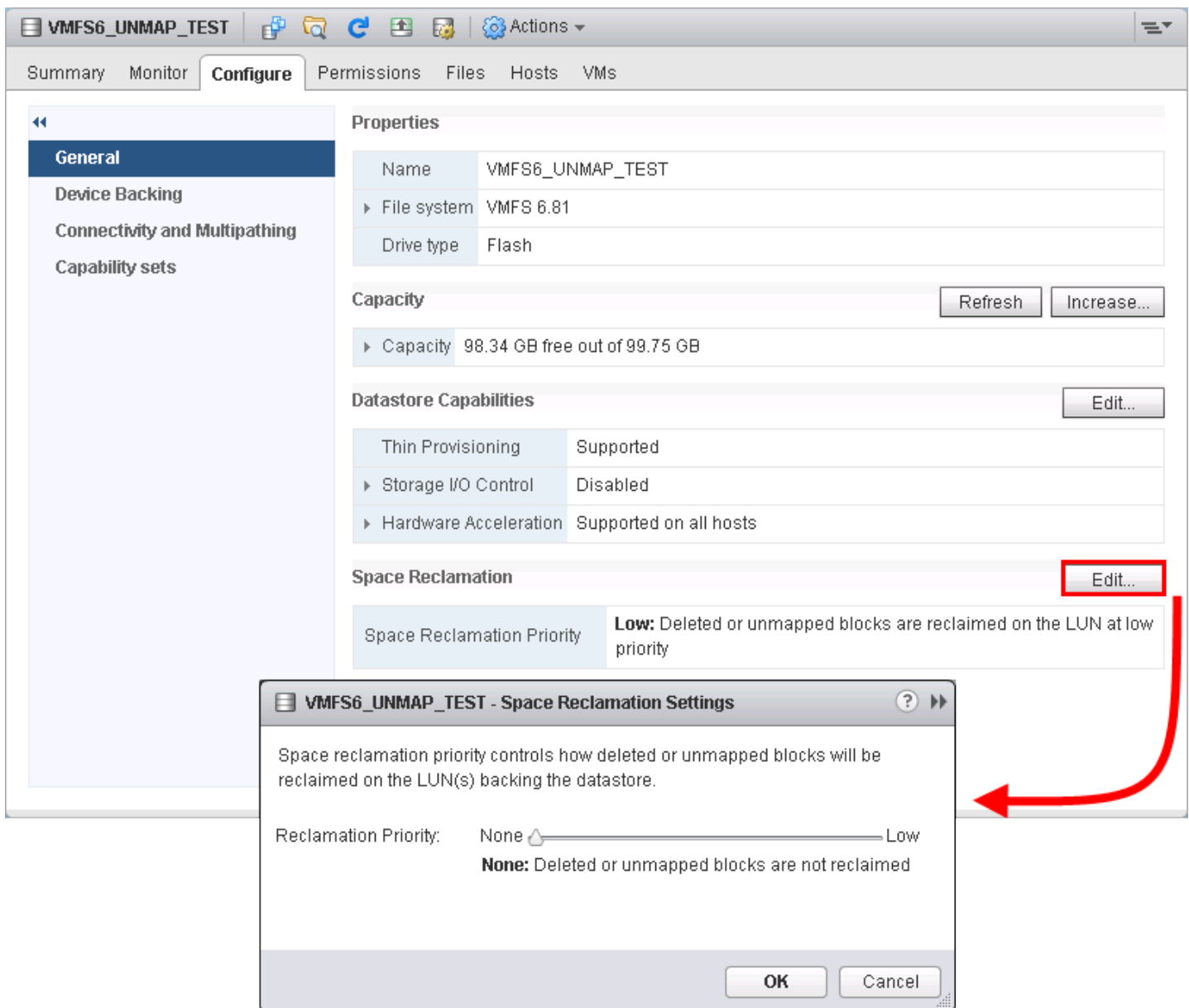


Figure 18. Modifying automated UNMAP on a VMFS 6 datastore

Changing the parameter is also possible through CLI, demonstrated in Figure 19.

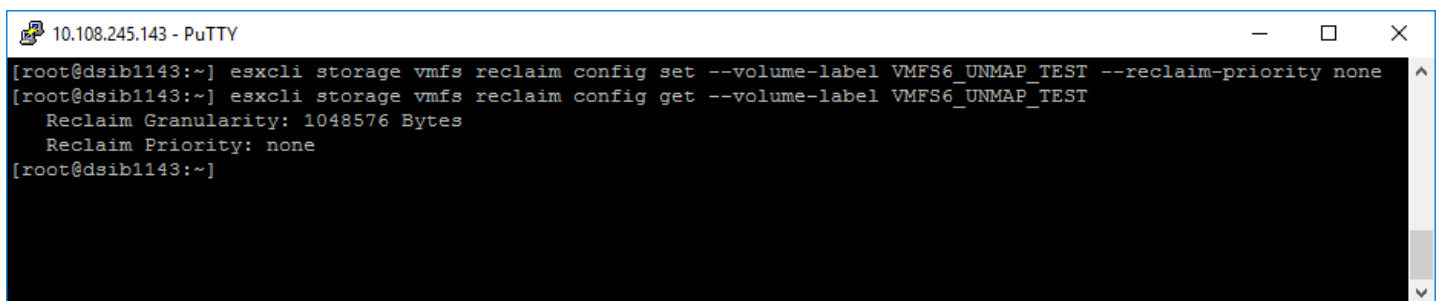


Figure 19. Modifying automated UNMAP through CLI

Though vSphere 6.5 offers automated UNMAP, manual UNMAP using *esxcli* is still supported and can be run on a datastore whether or not it is configured with automated UNMAP.

It is important to note that 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 being used – software to offload or offload to software – and finish the task with that methodology (assuming no further change is made).

Full Copy Tuning

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 VMAX arrays, the default value should not be altered. However, in all-VMAX or a combination of VMAX and VNX environments for vSphere 4.1 – 6.x, customers are advised to take advantage of VMware’s ability to send larger chunks to the array for copy. In vSphere 6 there is a more advanced capability that can be used in all-VMAX environments to increase the extent size even higher. The following two sections outline how to make these changes in the different vSphere environments.

vSphere 4.1 – 6.x

In vSphere 4.1 – 6.x the default copy size can be incremented to a maximum value of 16 MB and should be set at that value for all VMAX/VNX 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 20:

```

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 20. 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 21.

```

VMware vSphere PowerCLI 5.5 Release 1
PowerCLI C:\> connect-VIServer -Server 10.108.246.40 -Protocol https -User administrator@vsphere.local -Password
Name          Port  User
-----
10.108.246.40  443  administrator@vsphere.local

PowerCLI C:\> Get-VMHost | Get-AdvancedSetting -Name DataMover.MaxHWTransferSize
! Set-AdvancedSetting -Value 16384

Perform operation?
Modifying advanced setting 'DataMover.MaxHWTransferSize'.
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help
(default is "Y"):A

Name          Value          Type          Description
-----
DataMover.MaxHWTr... 16384          VMHost
DataMover.MaxHWTr... 16384          VMHost
DataMover.MaxHWTr... 16384          VMHost

PowerCLI C:\> _

```

Figure 21. Changing the default copy size for Full Copy using vSphere PowerCLI

vSphere 6.x

In vSphere 6.x, the previous outlined mechanism will still work; however 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 and can be utilized with both VMAX and VMAX3/VMAX All Flash.

Claim rules have always been used to implement VAAI for VMAX (Figure 7). VAAI is a plug-in claim rule that says if a device is from a particular vendor, in this case an EMC VMAX device, it should go through the VAAI filter. In other words if VAAI can be used it will be used (various situations exists – 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 has done is to add 3 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 22 shows the output from vSphere 5.x and Figure 23 shows the output from vSphere 6 with the new columns and their default values.

```

10.108.245.115 - PuTTY
~ # esxcli storage core claimrule list --claimrule-class=VAAI
Rule Class Rule Class Type Plugin Matches
-----
VAAI 65429 runtime vendor VMW_VAAIP_MASK vendor=MSFT model=Virtual HD
VAAI 65429 file vendor VMW_VAAIP_MASK vendor=MSFT model=Virtual HD
VAAI 65430 runtime vendor VMW_VAAIP_SYMM vendor=EMC model=SYMMETRIX
VAAI 65430 file vendor VMW_VAAIP_SYMM vendor=EMC model=SYMMETRIX
VAAI 65431 runtime vendor VMW_VAAIP_CX vendor=DGC model=*
VAAI 65431 file vendor VMW_VAAIP_CX vendor=DGC model=*
VAAI 65432 runtime vendor VMW_VAAIP_EQL vendor=EQLOGIC model=*
VAAI 65432 file vendor VMW_VAAIP_EQL vendor=EQLOGIC model=*
VAAI 65433 file vendor VMW_VAAIP_NETAPP vendor=NETAPP model=*
VAAI 65434 runtime vendor VMW_VAAIP_HDS vendor=HITACHI model=*
VAAI 65434 file vendor VMW_VAAIP_HDS vendor=HITACHI model=*
VAAI 65435 runtime vendor VMW_VAAIP_LHN vendor=LEFTHAND model=*
VAAI 65435 file vendor VMW_VAAIP_LHN vendor=LEFTHAND model=*
~ #
  
```

Figure 22. VAAI claim rule class in vSphere 5.x

```

10.108.245.116 - PuTTY
[root@dsib1116:~] esxcli storage core claimrule list --claimrule-class=VAAI
Rule Class Rule Class Type Plugin Matches XCOPY Use Array Reported Values XCOPY Use Multiple Segments XCOPY Max Transfer Size
-----
VAAI 65429 runtime vendor VMW_VAAIP_MASK vendor=MSFT model=Virtual HD false false 0
VAAI 65429 file vendor VMW_VAAIP_MASK vendor=MSFT model=Virtual HD false false 0
VAAI 65430 runtime vendor VMW_VAAIP_SYMM vendor=EMC model=SYMMETRIX false false 0
VAAI 65430 file vendor VMW_VAAIP_SYMM vendor=EMC model=SYMMETRIX false false 0
VAAI 65431 runtime vendor VMW_VAAIP_CX vendor=DGC model=* false false 0
VAAI 65431 file vendor VMW_VAAIP_CX vendor=DGC model=* false false 0
VAAI 65432 runtime vendor VMW_VAAIP_EQL vendor=EQLOGIC model=* false false 0
VAAI 65432 file vendor VMW_VAAIP_EQL vendor=EQLOGIC model=* false false 0
VAAI 65433 file vendor VMW_VAAIP_NETAPP vendor=NETAPP model=* false false 0
VAAI 65434 runtime vendor VMW_VAAIP_HDS vendor=HITACHI model=* false false 0
VAAI 65434 file vendor VMW_VAAIP_HDS vendor=HITACHI model=* false false 0
VAAI 65435 runtime vendor VMW_VAAIP_LHN vendor=LEFTHAND model=* false false 0
VAAI 65435 file vendor VMW_VAAIP_LHN vendor=LEFTHAND model=* false false 0
[root@dsib1116:~]
  
```

Figure 23. VAAI claim rule class in vSphere 6.x

By altering the rules and changing these 3 new settings, one can adjust the max transfer size up to a maximum of 240 MB. Though the columns show in both the Filter and VAAI claim rule classes, only the VAAI class needs to be changed. The new required settings are:

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

It is not sufficient to simply change the transfer size because VMware requires that the 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 VMAX 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 so it is important to set that to 16 MB. In addition, in order for the VMAX to accept up to 240 MB, multiple segments must be enabled since a single segment is limited to 30 MB (there are 8 segments available). Therefore when we modify the claim rules we will set the size to 240 MB and set both other columns to true.

The process to change the claim rules involves removing the existing VAAI rule class, adding it back with the new columns set, and then loading the rule (to be completed on all servers involved in testing). Following this a reboot of the server is needed.¹¹ The commands to run on the ESXi host as root are the following. They will not return a response:

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

To verify the changes are set before reboot, list the claim rules as in Figure 24 below:

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

¹¹ 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.

Rule Class	Rule	Vendor	XCOPY Use Array Reported Values	XCOPY Use Multiple Segments	XCOPY Max Transfer Size
VAAI	65429	Virtual HD	false	false	0
VAAI	65429	Virtual HD	false	false	0
VAAI	65430	SYMMETRIX	true	true	240
VAAI	65430	SYMMETRIX	true	true	240
VAAI	65431	*	false	false	0
VAAI	65431	*	false	false	0
VAAI	65432	C model=*	false	false	0
VAAI	65432	C model=*	false	false	0
VAAI	65433	model=*	false	false	0
VAAI	65433	model=*	false	false	0
VAAI	65433	model=*	false	false	0
VAAI	65434	I model=*	false	false	0
VAAI	65434	I model=*	false	false	0
VAAI	65435	VD model=*	false	false	0

Figure 24. Updated VAAI claim rule for XCOPY

Note however that the changes will not take effect until a reboot of the server. Once the reboot is complete, re-run the command to ensure the changes appear. EMC always recommends using the maximum value of 240 MB. Using the maximum ensures that if possible, a 240 MB extent will be used.

It is important to remember that 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* when the claim rules are not changed or a non-VMAX 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.

In the event it is desired 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 add -r 65430 -t vendor -V EMC -M SYMMETRIX -P VMW_VAAIP_SYMM -c VAAI

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

A reboot is again required.

Thin vmdks

One area where the increased extent size has made no difference in the past is with thin vmdks. As the section Caveats for using hardware-accelerated Full Copy explains, VMware only uses 1 MB extents when copying thin vmdks no matter the value of the extent size. This holds true for both vSphere 4.1 and 5.x. In vSphere 6, however, VMware makes an attempt 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 VMAX. 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 somewhat unexpected and that is 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 essentially 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.

The following graph, Figure 25, demonstrates the effectiveness of the new vSphere 6 claim rule for XCOPY. Note in particular the clone of the clone for the thin vmdk (light green) as explained in the previous paragraph.

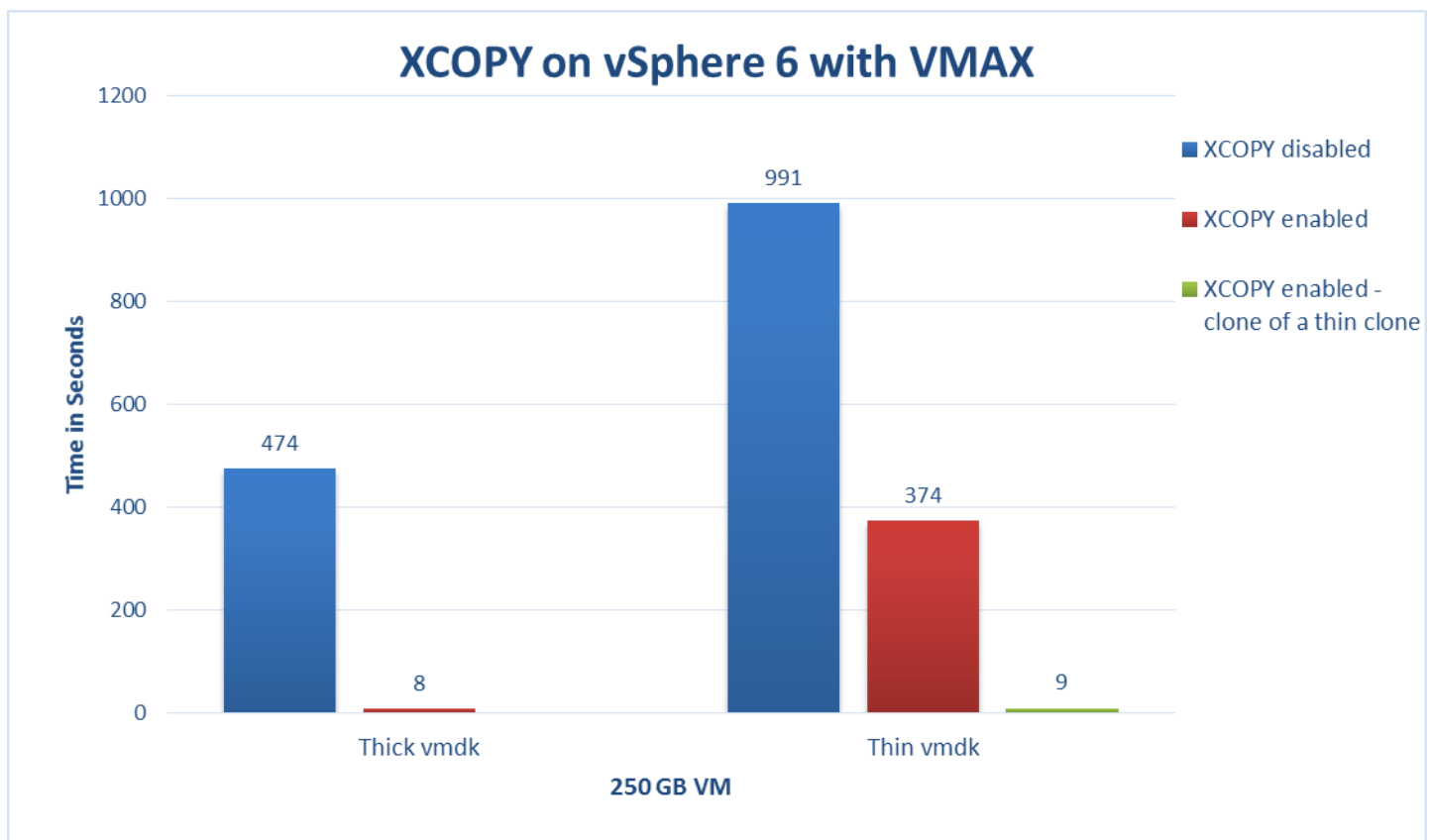


Figure 25. XCOPY performance using vSphere 6 claim rules

SRDF and XCOPY

As previously mentioned in the section Hardware-accelerated Full Copy or XCOPY, XCOPY is implemented asynchronously on the VMAX. Unfortunately this implementation can cause a potential issue for customers when XCOPY is used in conjunction with SRDF target devices. On the VMAX running the Enginuity operating system, if the target device of an XCOPY operation is

an SRDF/S device in a consistency group, the group can be rendered into a “sync in progress” mode until the copy completes on the backend of the VMAX. If this is undesirable state, customers are advised to disable XCOPY through the GUI or CLI while cloning to these devices. This will ensure VMware uses host software copy which is synchronous. Alternatively at Engenuity level 5876.268.174, XCOPY can be disabled on the VMAX for all SRDF devices. This setting will not impact other XCOPY activities.

By default, on the VMAX3 XCOPY is disabled for all SRDF devices so no user intervention is required. Beginning with HYPERMAX OS version 5977 2015 Q3 SR, EMC implements synchronous XCOPY for SRDF devices.¹² The synchronous process ensures that a consistency group will not enter a “sync in progress” state. The two main reasons EMC implemented the synchronous 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 EMC’s asynchronous implementation. As such, while EMC synchronous copy will generally outpace VMware’s host copy, 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.

eNAS and VAAI

On the VMAX3 starting with the 5977 Q4 release EMC offers Embedded NAS (eNAS) guest OS (GOS). VMAX3/VMAX All Flash with eNAS extends the value of VMAX3/VMAX All Flash to file storage by enabling customers to leverage vital Tier 1 features including SLO-based provisioning, Host I/O Limits, and FAST technologies for both block and file storage. eNAS enables virtual instances of VNX Software Data Movers and Control Stations to leverage a single platform creating a unified VMAX3/VMAX All Flash array. The VMAX3/VMAX All Flash unified solution eliminates gateway complexity by leveraging standard VMAX3/VMAX All Flash hardware and a factory pre-configured eNAS solution.

Because eNAS is based on the VNX software, file systems configured on the VMAX3/VMAX All Flash support VAAI. The feature in particular that NAS can take advantage of is NFS Clone Offload, though other features include extended stats, space reservations, and snap of a snap. In essence the NFS clone offload works much the same way as XCOPY as it offloads ESXi clone operations to the VNX Data Mover. The implementation of VAAI on VNX for NAS, however, is a manual process and is not enabled by default like block. It requires the installation of a plug-in on the ESXi host.

To install the plug-in, download the NAS plug-in from EMC support - EMCNasPlugin-2.0-4.zip. 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 26.

¹² This change is automatically applied during upgrade to this SR. If circumstances require, it can be changed by EMC.

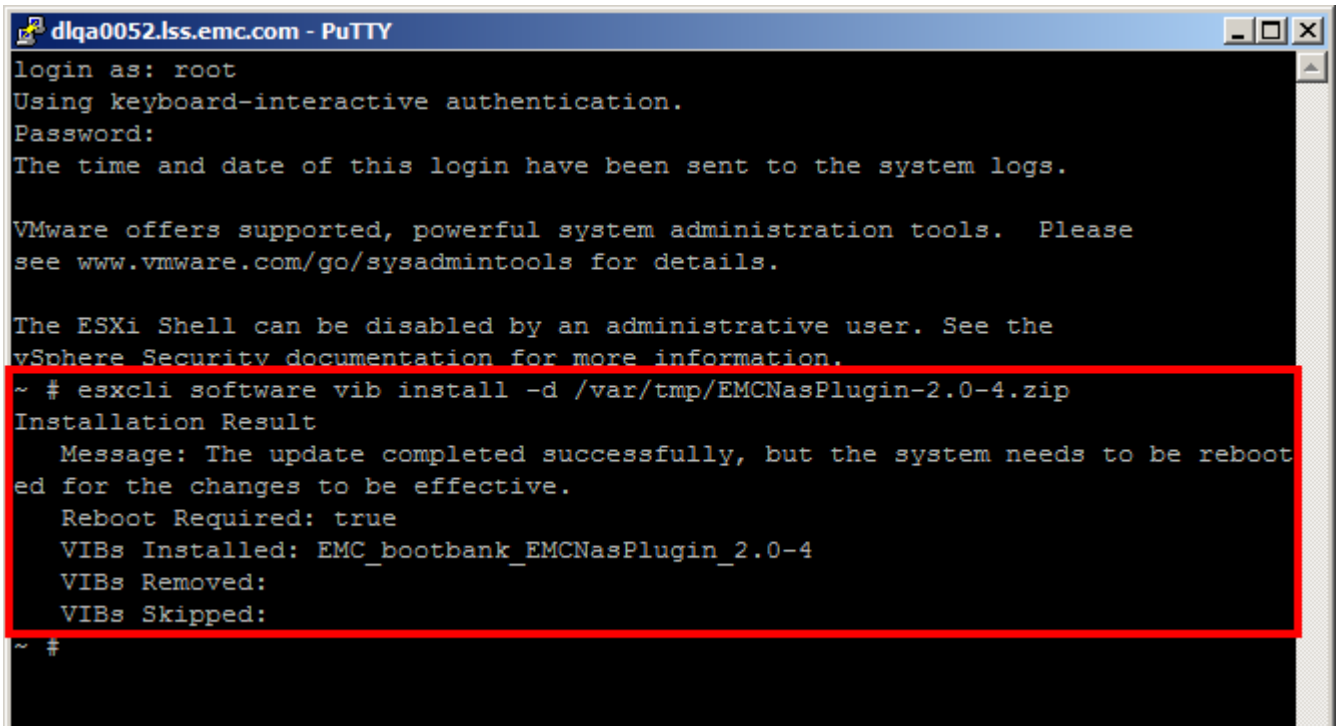


Figure 26. Installation of the NAS plug-in for VAAI

Once installed, the vSphere Client will show that VAAI is supported on the NFS datastores as in Figure 27.

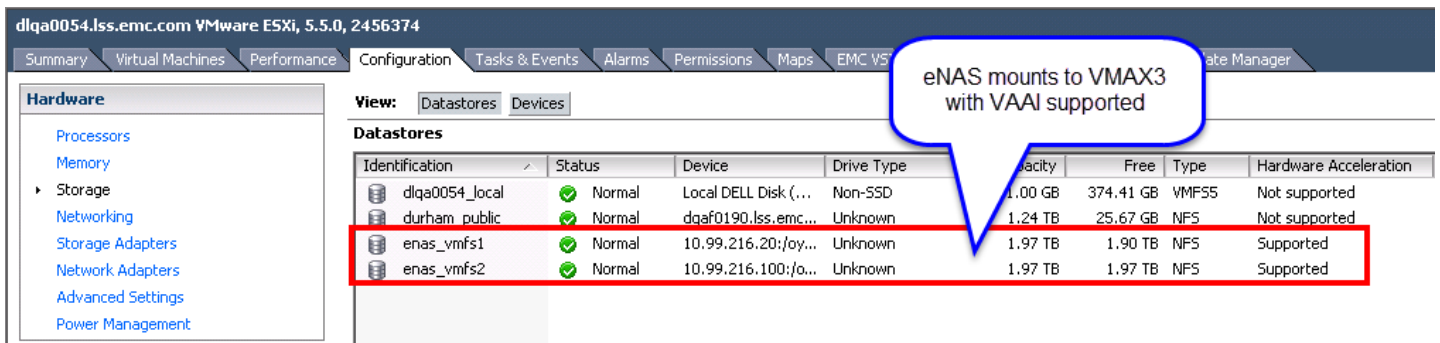


Figure 27. NFS datastore with VAAI support

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 28.

```

dlqa0054.lss.emc.com - PuTTY
~ # vmkfstools --extendedstat /vmfs/volumes/482273a0-81e18c11/dlqa0054_W2K12R2
dlqa0054_W2K12R2-000002.vmdk
Capacity bytes: 42949672960
Used bytes: 23700660224
Unshared bytes: 23700660224
~ #

```

Figure 28. VAAI NFS extended stats feature

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. VMware products such as View Composer and vCloud Director use View Composer Array Integration (VCAI) to initiate virtual machine clones using the VNX Data Mover API for nested snapshots. The clone operations are off-loaded to the VNX Data Mover. To take advantage of this feature, enable the nested clone support property when you create the file system as highlighted in Figure 29. It cannot be changed after the file system is created.

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

OK Apply Cancel Help

Figure 29. Setting nested clone support on a file system

Hardware-accelerated Full Copy

The primary use cases that show the benefit of hardware-accelerated Full Copy are related to deploying virtual machines – whether from a template or executing a hot or cold clone. With Full Copy the creation of any of these virtual machines will be offloaded to the array. In addition to deploying new virtual machines, Full Copy is also utilized by Storage vMotion operations, significantly reducing the time to move a virtual machine to a new datastore.

Following are a number of use cases for Full Copy concerned with the deployment of virtual machines – some from a template and others from cloning – and Storage vMotion. Each use case test is run on thin devices with hardware-accelerated Full Copy disabled and enabled. For the Full Copy enabled results both the 4 MB and 16 MB maximum copy sizes are displayed, as well as the results for VMAX and VMAX3/VMAX All Flash where possible.¹³

The benefits of using a 16 MB copy size over the default of 4 MB are readily apparent in the use cases, as is the superiority of the VMAX3/VMAX All Flash platform when compared to the previous generation.

In order to distinguish the VMAX platform from the VMAX3/VMAX All Flash platform in the graphs, the abbreviations V2 and V3 are used. Also the terms Full Copy and XCOPY will be used interchangeably.

Use case configuration

In general, the Full Copy use cases were conducted using the same virtual machine, or a clone of that virtual machine converted to a template. This virtual machine was created using the vSphere Client, taking all standard defaults and with the following characteristics:

- 40 GB virtual disk
- Windows Server 2008 R2 operating system (no VMware Tools)¹⁴

The virtual machine was then filled with data equal to 25 GB of the 40 GB. This configuration might be thought of as a worst case scenario for customers since most virtual machines that are used for cloning, even with the OS and all applications deployed on it, do not approach 25 GB of actual data. Remember that the default configuration of a virtual machine uses a zeroedthick virtual disk. This type of disk, though reserving the 40 GB of space on the VMFS, will not actually write to disk until necessary. This is distinct from a fault-tolerant virtual machine, which must have an eagerzeroedthick (EZT) virtual disk, a disk that is zeroed out entirely upon creation. Since only a small portion of a 40 GB virtual machine will typically contain data, the

¹³ As it was not possible to recreate the testing environment used for active clones (Figure 26), VMAX3 results are not included; though it is a reasonable assumption to conclude given the other testing that the VMAX3 numbers would be about ½ the 16 MB VMAX results for a similar extent size.

¹⁴ If using VMware Tools with Windows 2008 R2, there is a limitation wherein a hot clone will not use Full Copy because of VSS application quiescing. This can be disabled through a configuration parameter. Please see VMware KB articles 1031298 and 1028881 for more information.

time to clone a VM in a customer's environment with Full Copy will invariably take less time than the results presented in this study. In order to fully test the functionality and performance, however, 25 GB of data was used.

When creating non-fault-tolerant virtual machines, the cloning times are directly related to how much data is in the virtual machine, not the size of the virtual disk. This is true whether performing a software copy (Full Copy disabled), or a hardware-accelerated Full Copy. From the standpoint of cloning, a zeroedthick 20 GB VM that has 5 GB of data will take approximately the same amount of time to clone as a 100 GB VM with 5 GB of data.

All results presented in graphs of Full Copy functionality are dependent upon the lab environment under which they were generated. While the majority of the time hardware acceleration will outperform a software copy, there are many factors that can impact how fast both software clones and Full Copy clones are created. For software clones the amount of CPU and memory on the ESXi server as well as the network will play a role. For Full Copy the number of engines and directors and other hardware components on the VMAX will impact the cloning time. The VMAX also intelligently balances workloads to ensure the highest priority to mission critical tasks. In both cases, the existing load on the environment will most certainly make a difference. Therefore results will vary.

Use Case 1: Deploying a virtual machine from a template

The first use case for Full Copy is the deployment of a virtual machine from a template. This is probably the most common type of virtual machine duplication for customers. A customer begins by creating a virtual machine, then installs the operating system and following this, loads the applications. The virtual machine is then customized so that when users are presented with the cloned virtual machine, they are able to enter in their personalized information. When an administrator completes the virtual machine base configuration, it is powered down and then converted to a template.

As using thin striped metadevices is a VMAX best practice for VMware environments, these clones were created on a datastore backed by a thin striped metadvice.¹⁵

The following graph in Figure 30 shows the time difference between clones created using Full Copy and those using software copy.

¹⁵ VMAX3 does not have a concept of metadevices.

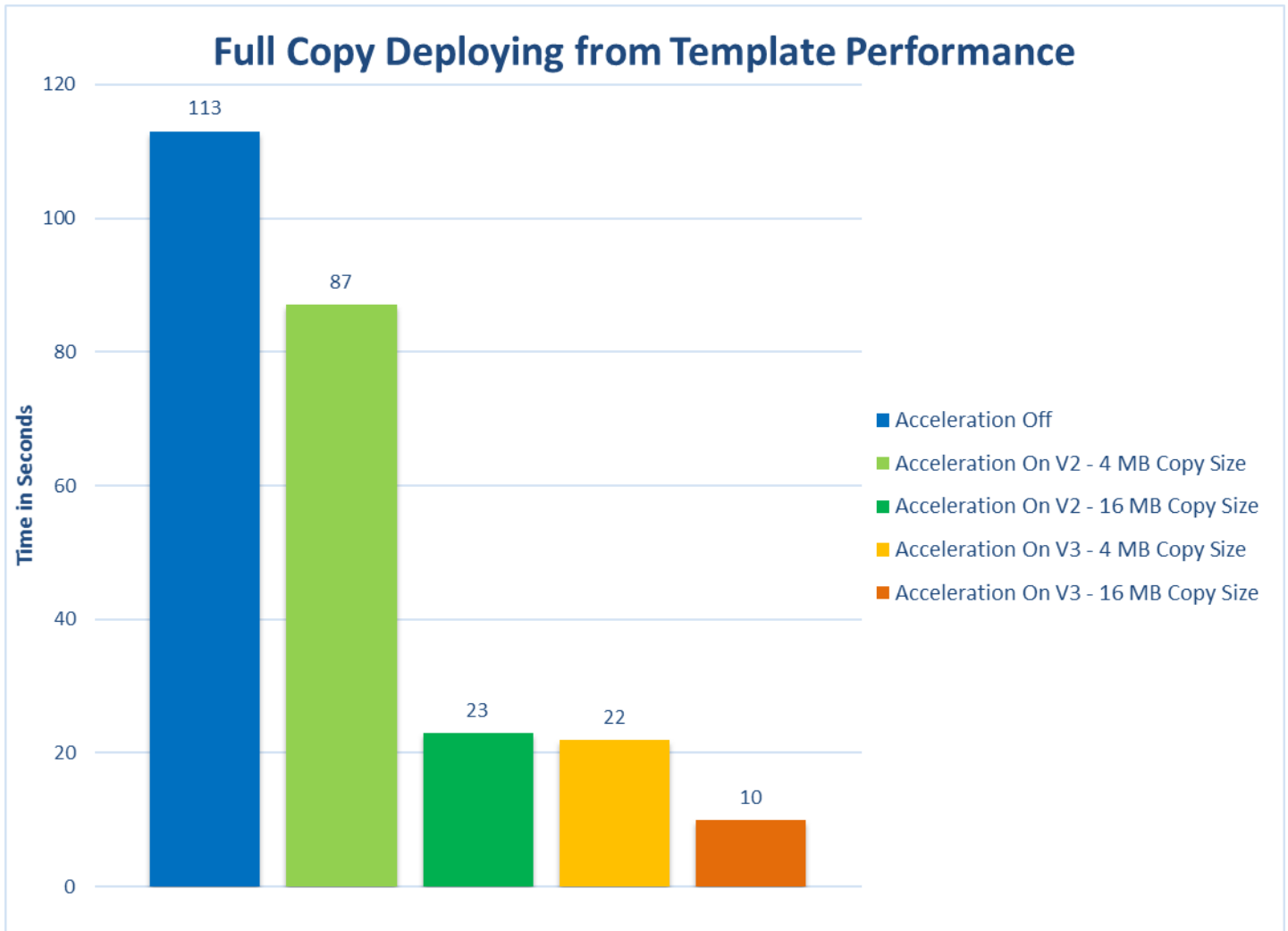


Figure 30. Deploying virtual machines from template

Use Case 2: Cloning hot and cold virtual machines

The second use case involves creating clones from existing virtual machines, both running (hot) and not running (cold). In addition, a third type of test was run in which a hot clone was sustaining heavy read I/O while it was being cloned. In each instance, utilizing the Full Copy functionality resulted in a significant performance benefit as visible in Figure 31, Figure 32, and Figure 33.

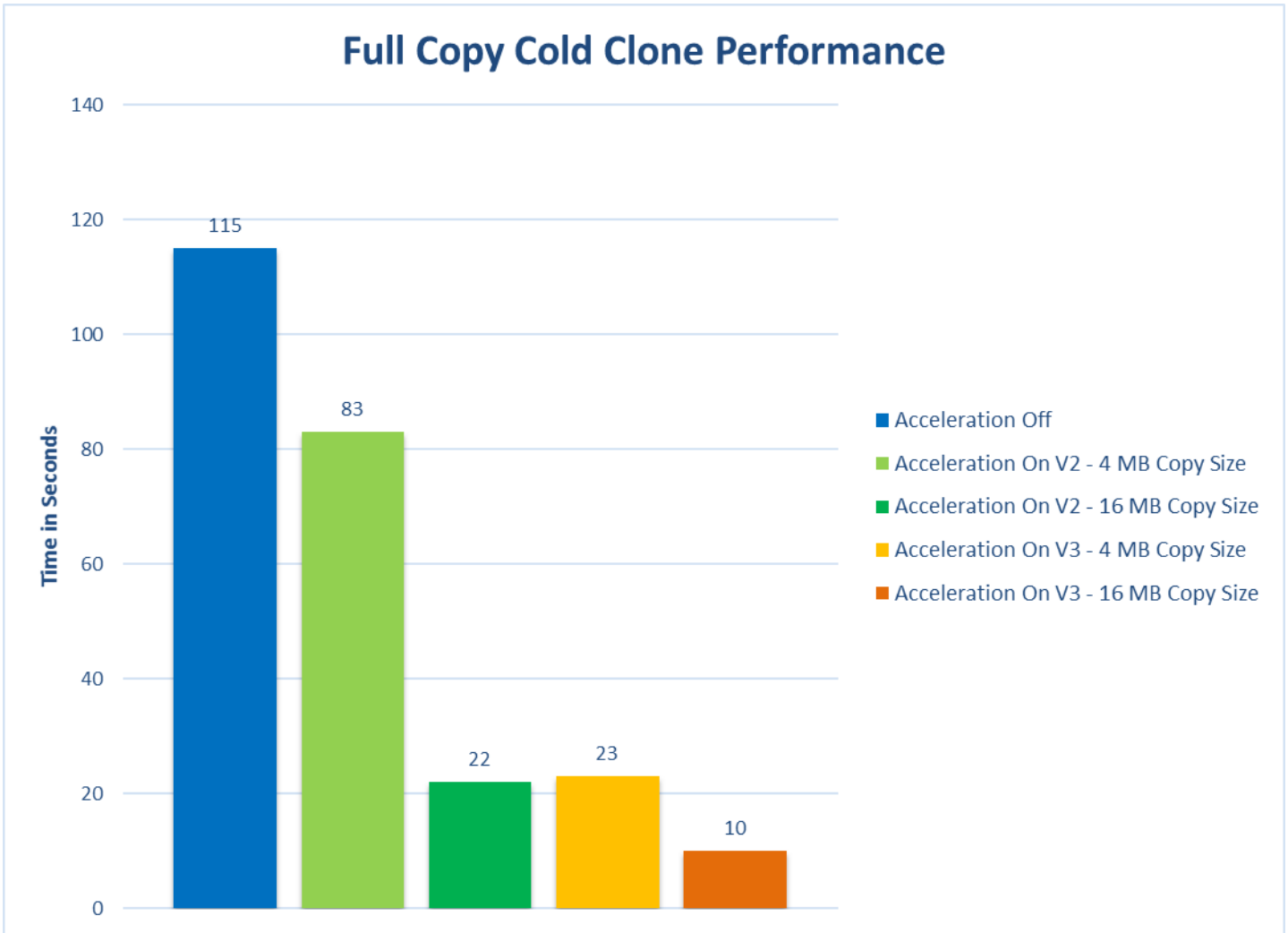


Figure 31. Performance of cold clones using Full Copy

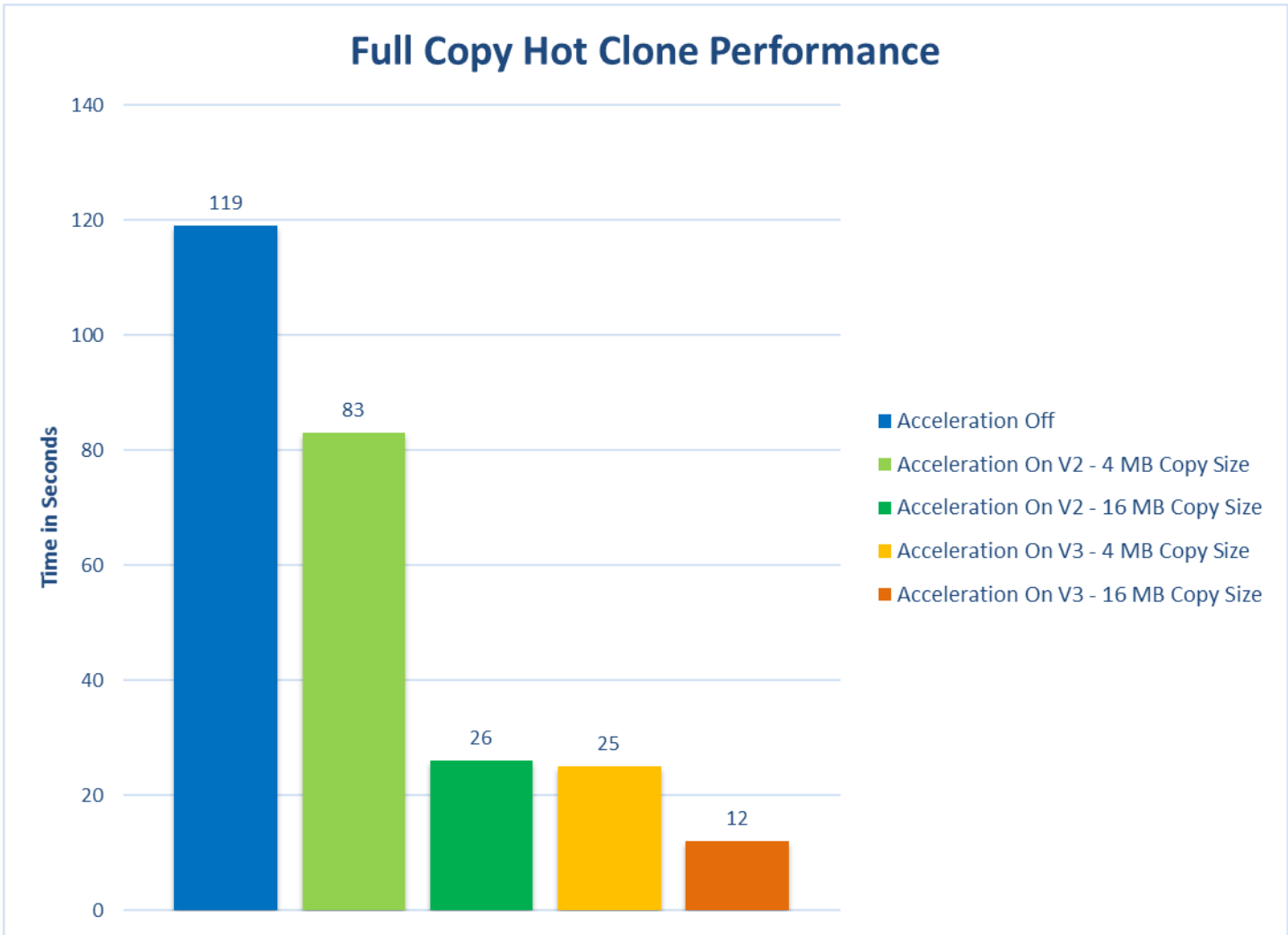


Figure 32. Performance of hot clones using Full Copy

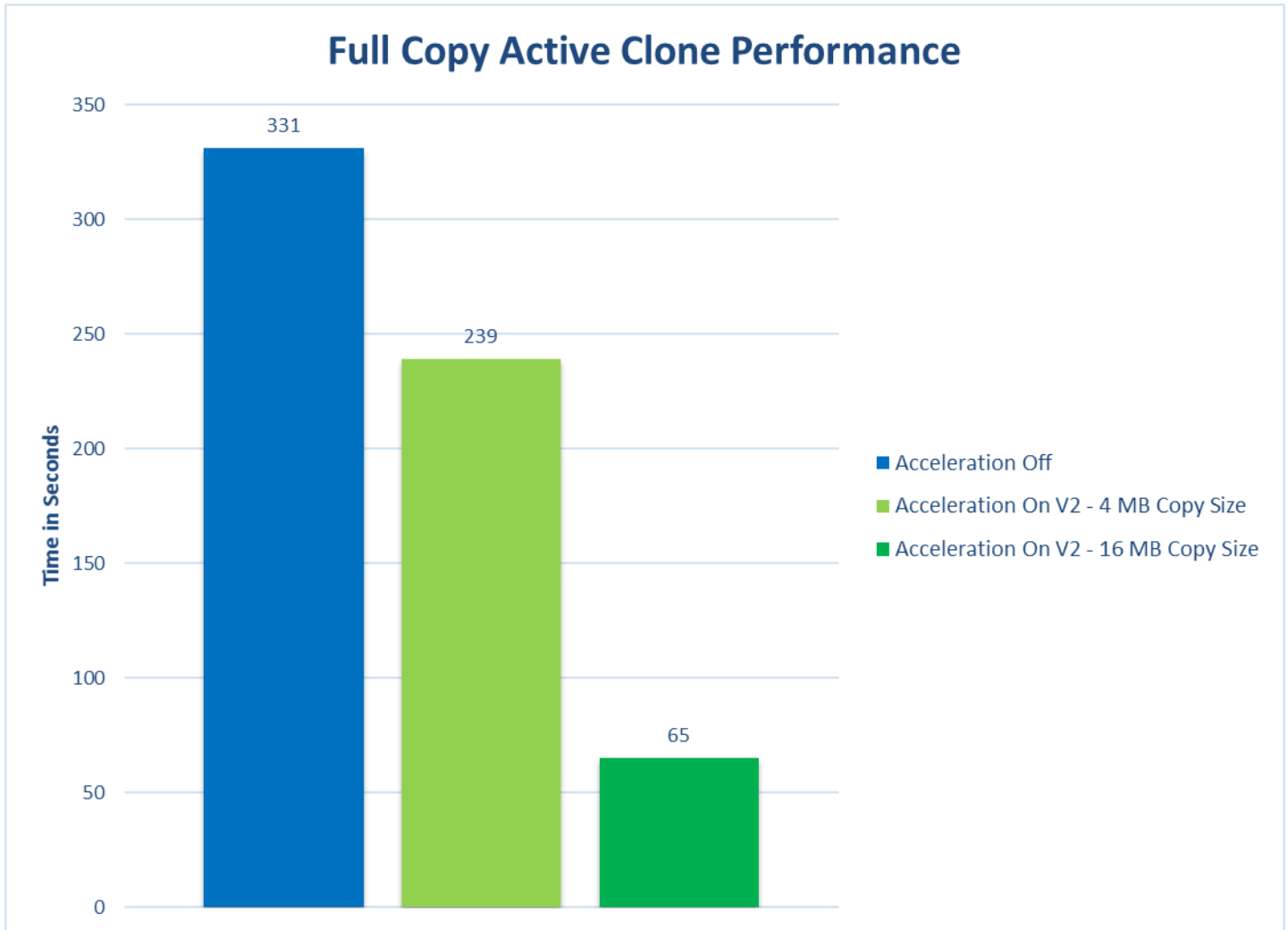


Figure 33. Cloning running VMs under load using Full Copy

Use Case 3: Creating simultaneous multiple clones

The third use case that was tested was to deploy four clones simultaneously from the same source virtual machine. This particular test can only be conducted against a cold virtual machine.

As seen in Figure 34, the results again demonstrate the benefit of offloading as compared to software copying.

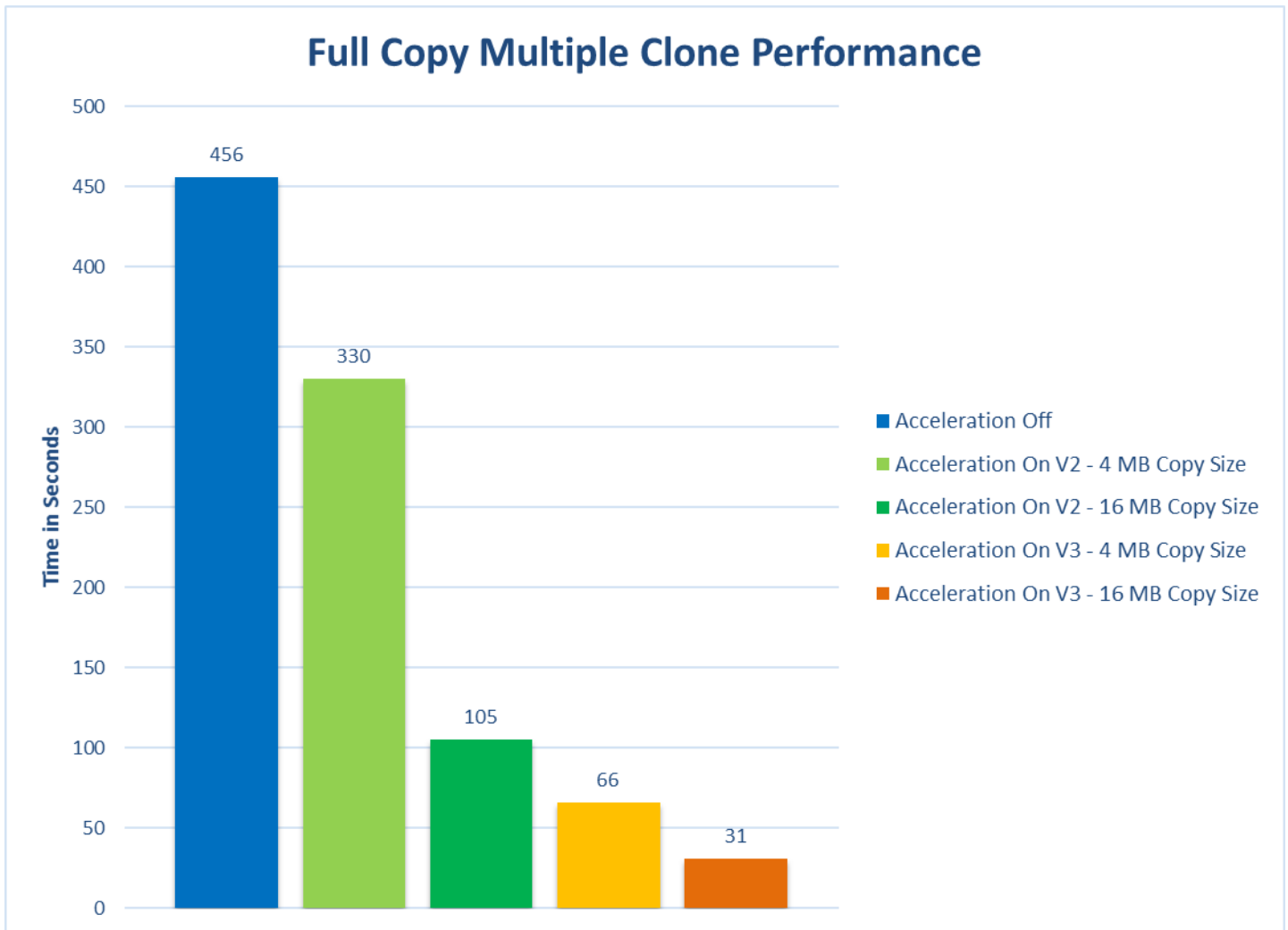


Figure 34. Cloning multiple virtual machines simultaneously with Full Copy

Use Case 4: Storage vMotion

The final use case is one that demonstrates a great benefit for customers who require datastore relocation of their virtual machine, but at the same time desire to reduce the impact to their live applications running on that virtual machine. This use case then is for Storage vMotion. With Full Copy enabled, the process of moving a virtual machine from one datastore to another datastore is offloaded. As mentioned previously, software copying requires CPU, memory, and network bandwidth. The resources it takes to software copy, therefore, might negatively impact the applications running on that virtual machine that is being moved. By utilizing Full Copy this is avoided and additionally as seen in Figure 35, Full Copy is far quicker in moving that virtual machine.

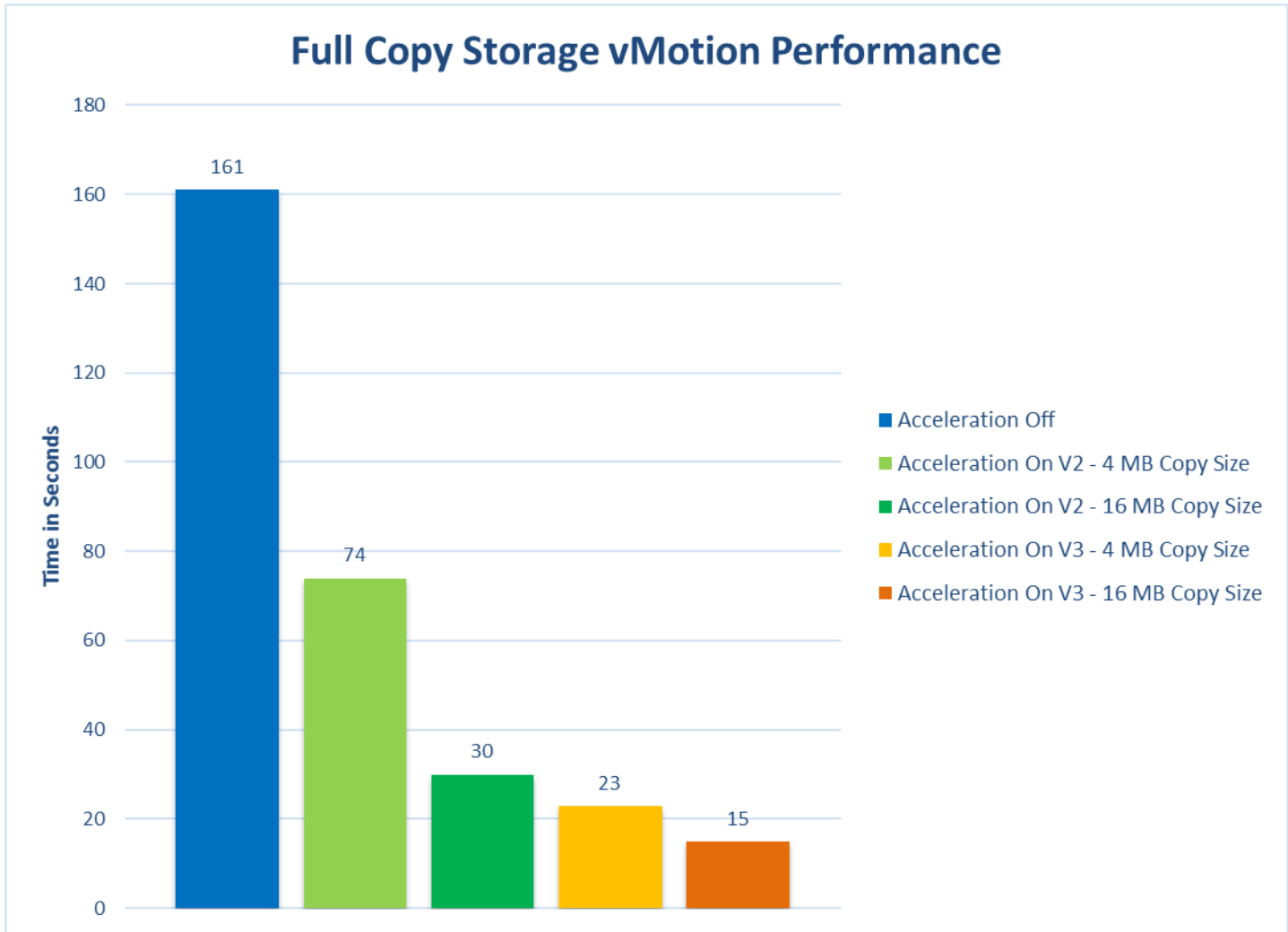


Figure 35. Storage vMotion using Full Copy

Server resources – impact to CPU and memory

While the relative time benefit of using Full Copy is apparent in the presented use cases, what of the CPU and memory utilization on the ESXi host? In this particular test environment and in these particular use cases, the CPU and memory utilization of the ESXi host did not differ significantly between performing a software clone or a Full Copy clone. That is not to say, however, this will hold true of all environments. The ESXi host used in this testing was not constrained by CPU or memory resources in any way as they were both plentiful; and the ESXi host was not under any other workload than the testing. In a customer environment where resources may be limited and workloads on the ESXi host many and varied, the impact of running a software clone instead of a Full Copy clone may be more significant.

Caveats for using hardware-accelerated Full Copy¹⁶

The following are some general caveats that EMC 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.
- As can be seen from the graphs, utilizing the 16 MB copy size improves performance by factors over the default of 4 MB. Therefore if the VMware environment is only utilizing VMAX or VMAX and VNX storage, adjust the parameter accordingly as detailed in this paper. If using vSphere 6, follow the process in this paper to increase the extent size to 240 MB.
 - One caveat to the extent size change is that the virtual thin disk format (vmdk) is not impacted by the copy size in vSphere 4.1-5.x. Thin disks will always copy in the extent size of the datastore block size, regardless of the parameter setting. For VMFS3 and VMFS5, the default is 1 MB. Therefore it will always take longer to clone/move a thin disk than a thick disk of equal size (actual data size) with Full Copy. This is not the case with vSphere 6.
- Limit the number of simultaneous clones using Full Copy to three or four. This is not a strict limitation, but EMC believes this will ensure the best performance when offloading the copy process.
- A VMAX metadvice that has SAN Copy, TimeFinder/Clone, TimeFinder/Snap, TimeFinder SnapVX or ChangeTracker sessions and certain RecoverPoint sessions will not support hardware-accelerated 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.
- 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 Copy¹⁷, certain RDF operations, such as an RDF failover, will be blocked until the VMAX has completed copying the data from a clone or Storage vMotion.
 - SRDF/Star is not supported with Full Copy if the device is a metadvice.

¹⁶ Beginning with Engenuity 5876.159.102, Full Copy (XCOPY) can be disabled at the system level if the caveats are particularly problematic for a customer. A special E Pack is required for any Engenuity 5875 release and any 5876 release prior to 159.102. If this change is required, please contact EMC Customer Support as there is no capability for customers to disable XCOPY even if at the correct Engenuity version.

¹⁷ On the VMAX3 Full Copy is automatically disabled for RDF devices until release 5977 2015 Q3 SR.

- Full Copy will not be used if there is a metadevice reconfiguration taking place on the device backing the source or target datastore. VMware will revert to traditional writes in these cases. Conversely if a Full Copy was recently executed and a metadevice reconfiguration is attempted on the source or target metadevice backing the datastore, it will fail until the background copy is complete.
- When the available capacity of a Storage Resource Pool (SRP) in a VMAX3 or VMAX All Flash 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 EMC 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.

SRDF/Metro specific¹⁸

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 not the same as 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 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

¹⁸ Full Copy is enabled on SRDF/Metro devices beginning with release 5977 2016 Q3 SR.

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 still 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.

Compression specific¹⁹

The following are caveats when using Full Copy with devices in storage groups with compression enabled.

- 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.

Hardware-accelerated Block Zero

Hardware-accelerated Block Zero provides benefits and performance increases in a variety of use cases. This paper will cover the two most commonly 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 both VMAX and VMAX3/VMAX All Flash. The second use case discusses the performance gains achieved through the use of Block Zero when performing I/O in a virtual machine with virtual disks using the zeroedthick allocation format on the VMAX.

Use Case 1: Deploying fully allocated virtual machines

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 of these mechanisms 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

¹⁹ Compression on the VMAX All Flash is available beginning with release 5977 2016 Q3 SR.

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 VMAX. It therefore writes these zeros faster and more efficiently.

On the VMAX if thin devices are used in conjunction with Block Zero, the zeros are discarded by the VMAX and thin pool capacity is reserved for non-zero data. As a result no zeros are actually written by the VMAX; rather it just sets a flag called **Never Written By Host (NWBH)** which reserves the space. As previously explained, on the VMAX3/VMAX All Flash no space is actually reserved.

As the VMAX3/VMAX All Flash 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 essentially 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), that disk type must be selected when creating the virtual machine.

In this use case, new virtual disks of varying sizes (10 GB, 20 GB, 40 GB, 60 GB, 100 GB, and 200 GB) were deployed using the “eagerzeroedthick” allocation mechanism on thin devices (striped metadevices on VMAX²⁰) with hardware-accelerated Block Zero disabled and enabled. As shown by the graph in Figure 36, the noted improvements are significant. Although in viewing the graph it may appear that larger disk sizes bring greater improvement, the benefit is actually very consistent within each VMAX platform (abbreviated in the graph as follows: VMAX – V2, VMAX3/VMAX All Flash – V3). The performance benefit of the VMAX3/VMAX All Flash over the VMAX is also apparent both with and without Block Zero enabled.

²⁰ VMAX3 does not have a concept of metadevices.

Block Zero Virtual Disk Deployment Times

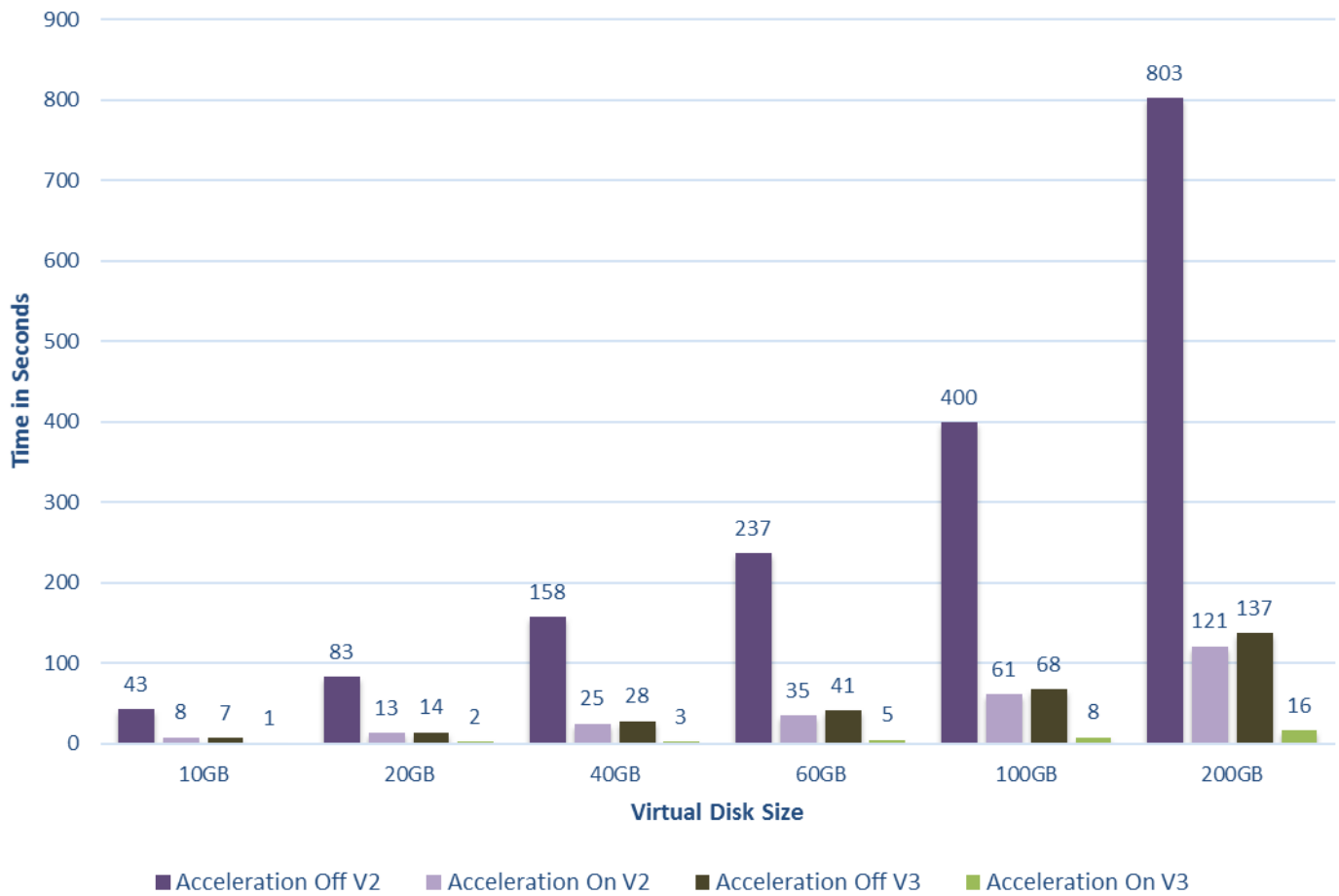


Figure 36. Fully allocated virtual disk deployment times using Block Zero

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 37 shows the ESXi performance graph of the write rate in KB per second (throughput) of the target VMAX3/VMAX All Flash 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 2 minutes and has a write rate exceeding 150,000 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 VMAX3/VMAX All Flash 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 587 KB/s for around 2 seconds — a 250x decrease in required throughput to create the virtual disk!

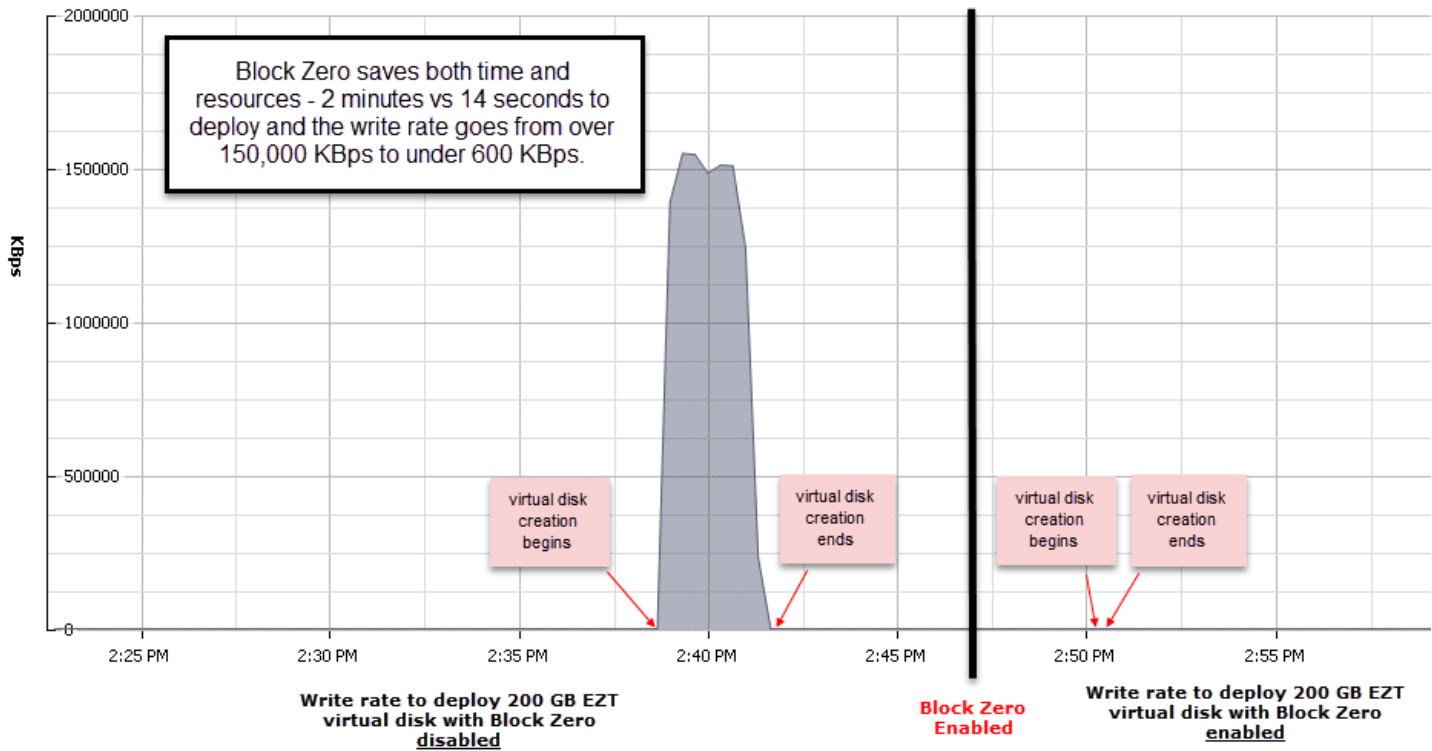


Figure 37. Throughput difference deploying an EZT virtual disk with Block Zero disabled and enabled

Use Case 2: Benefits of Block Zero when using zeroedthick virtual disks

By default in ESXi 4.x -6, new virtual disks are created using the “zeroedthick” allocation mechanism. This is also the recommended disk format for use with VMAX Virtual Provisioning 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 a thing of the past as the zeros are no longer written to initialize blocks before the guest OS writes to them.

In this use case, the benefits of having hardware-accelerated Block Zero enabled will be explored when writing to unallocated regions of a “zeroedthick” virtual disk using a “real-world” testing configuration. In this test, the tool IOMETER was used to demonstrate the performance impact of using Block Zero. The testing environment consisted of a VM that was configured in the same manner as the one used in the Full Copy use cases, created on a thin striped metadata datastore. For each test a 10 GB virtual disk was added to the VM and put online but not formatted. A “real-world” performance test was executed on the 10 GB virtual disk in IOMETER which had the following characteristics:

- 8 KB transfer request size
- 40% sequential and 60% random workload
- 35% write and 65% read distribution
- 5 minute test cycle

The IOMETER test was executed with Block Zero disabled and enabled, using a different 10 GB virtual disk for each test, though each was located on the same thin striped metadata datastore. By displacing the process of the zero initialization from the ESXi kernel to the VMAX, it introduces a marked improvement in writing to unallocated virtual disk regions. This performance improvement is depicted graphically in Figure 38, which shows a 30% improvement in both response time and the amount of connections possible when Block Zero is enabled.

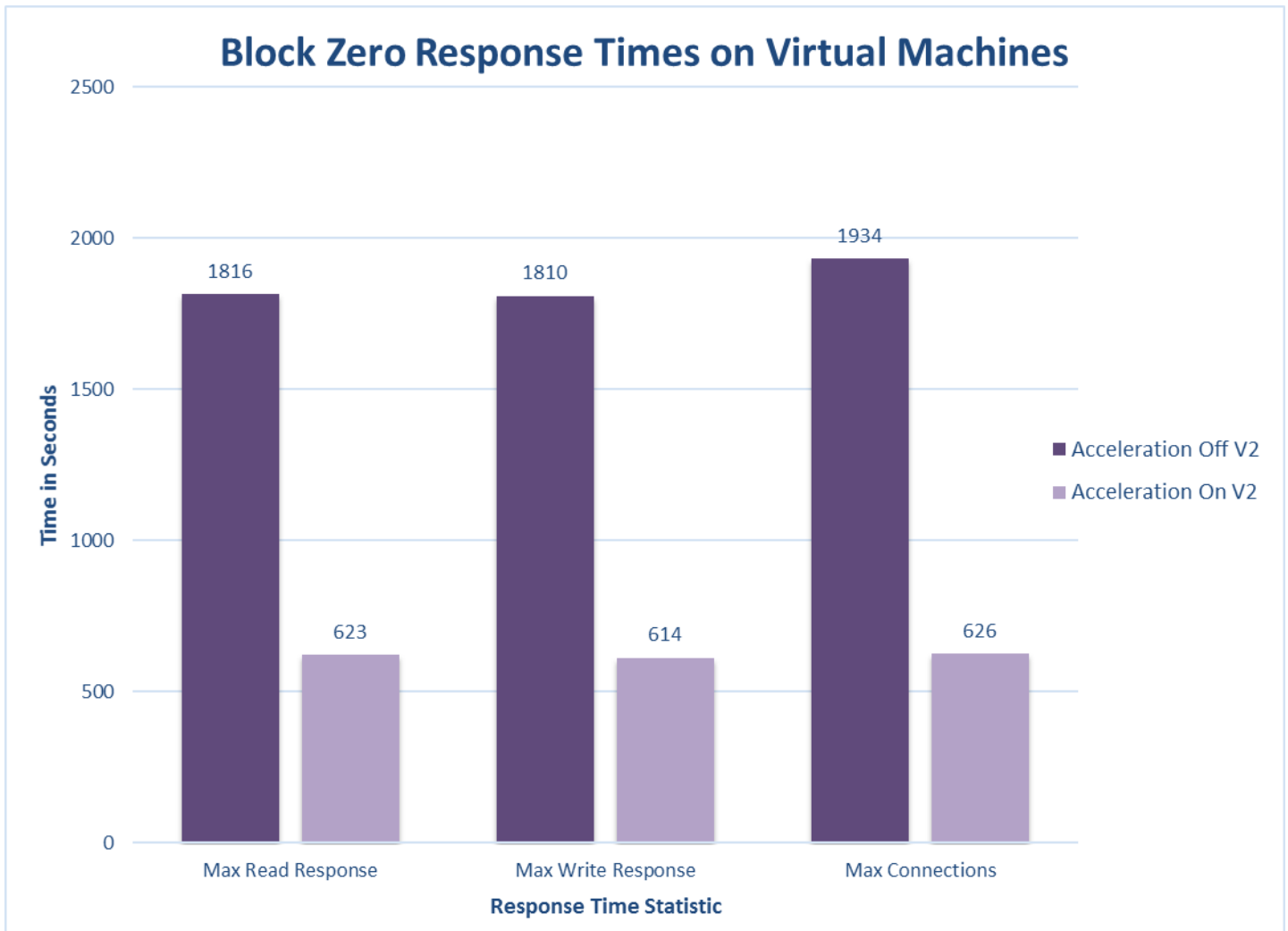


Figure 38. Response Time Statistics in IOMETER Block Zero test

In addition, Figure 39 demonstrates a 15% increase in the number of transactions per second accomplished during this test with Block Zero enabled.

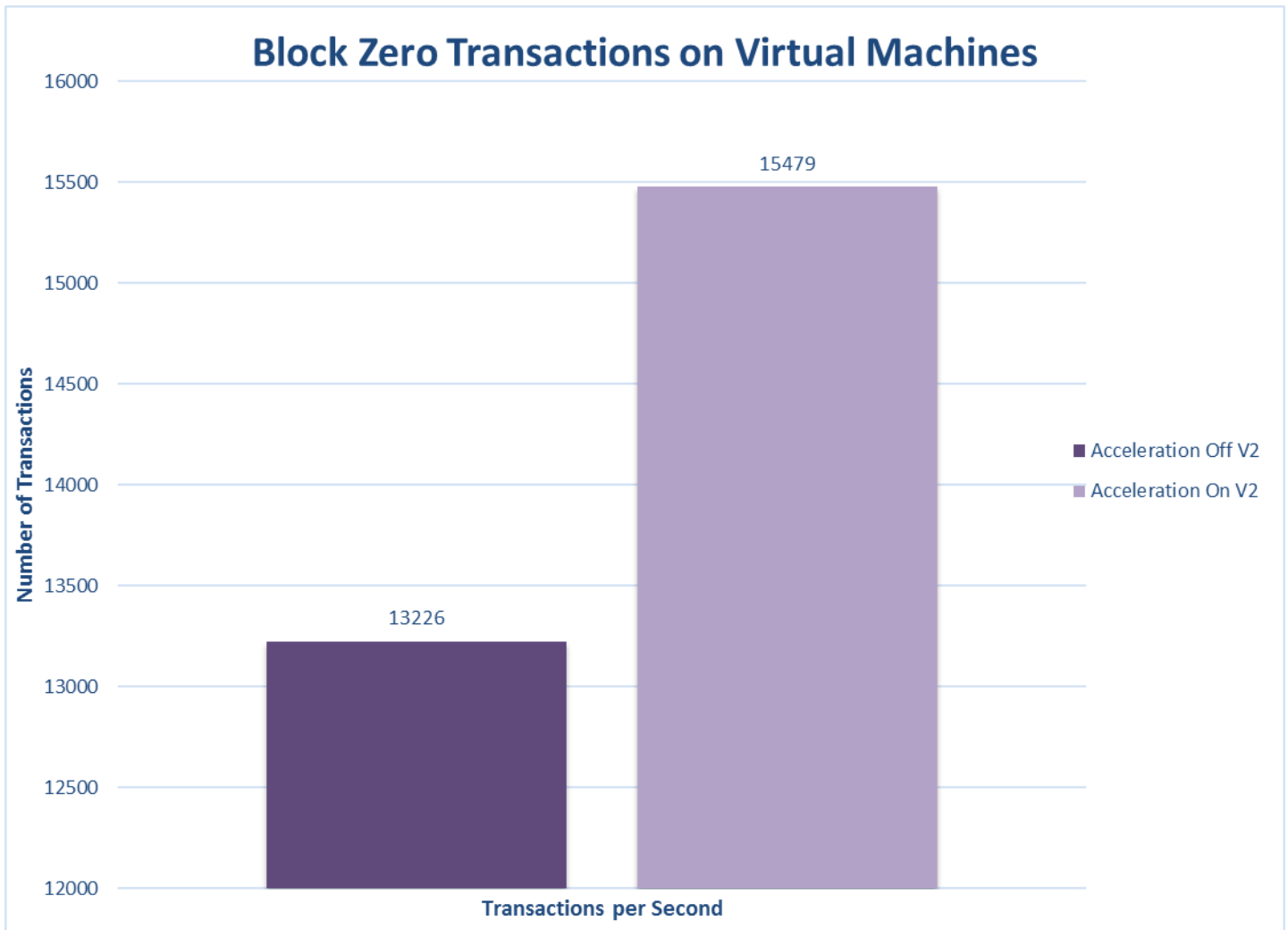


Figure 39. Transaction throughput with Block Zero disabled and enabled

Server resources – impact to CPU and memory

While the relative time benefit and drop of overall throughput of using block zero is readily apparent in the presented use cases, what of the CPU and memory utilization on the ESXi host? In this particular test environment and in these particular use cases, the CPU and memory utilization of the ESXi host did not differ significantly between when Block Zero was enabled or disabled. That is not to say, however, this will hold true of all environments. The ESXi host used in this testing was not constrained by CPU or memory resources in any way as they were both plentiful; and the ESXi host was not under any other workload than the testing. In a customer environment where resources may be limited and workloads on the ESXi host many and varied, the impact of utilizing Block Zero may be more significant.

Caveats for using hardware-accelerated Block Zero

The following are some general caveats that EMC 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 will not be used if there is a metadvice reconfiguration taking place on the device backing the datastore. VMware will revert to traditional writes in these cases.
- Block Zero commands will take longer to execute on devices in SRDF relationships.

UNMAP²¹

In this paper the VMware use cases for UNMAP 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 in the backing thin pool on the array. What is of particular interest from the array side, though, is how a device's configuration might impact the time it takes to deallocate the thin pool space. Dell EMC replication technologies such as TimeFinder and SRDF add an extra layer of complexity, so comparing these devices to standard thin devices will provide guidance to the user on how long a reclaim on a datastore might take.

The use cases herein are based on the manual reclamation of space and do not include the automatic UNMAP feature in vSphere 6.5 or Guest OS UNMAP in VMFS 5/6 or VVol datastores.

Before the use cases are presented, however, a number of practical examples of the UNMAP capability will be presented.

Manual UNMAP after Storage vMotion

In the following example we start with datastore UNMAP_500GB_DS, and as its name suggests it is a 500 GB datastore. The datastore has been created on a VMAX striped metadvice, similar to that utilized in the Full Copy use cases. The datastore has two 150 GB virtual machines configured on it, for a total of 300 GB of occupied space. Since these virtual machines are completely full²², they are also occupying 300 GB of space in the thin pool on the VMAX. EMC's Virtual Storage Integrator (VSI) provides a complete view of the storage usage both from the vSphere side, and the VMAX side in Figure 40.

²¹ 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.

²² Each virtual machine was created with a single 150 GB eagerzeroedthick virtual disk. Before creation, the Block Zero primitive was disabled which forced VMware to write all 150 GB worth of zeros to the array, completely occupying all the space.

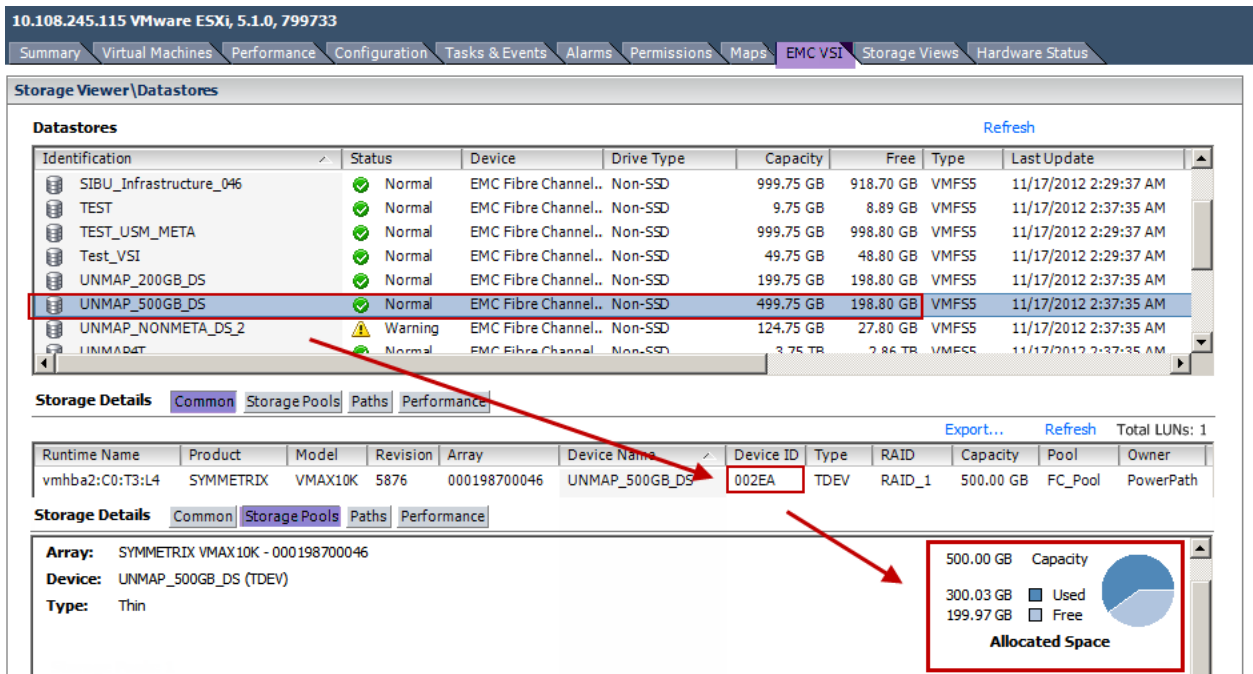


Figure 40. EMC Virtual Storage Integrator

Unisphere for VMAX can also be used to obtain the thin pool information. Using the device ID of 02EA and the thin pool name of FC_Pool from VSI, we can locate it in the Fibre Channel pool (FC_Pool) as seen in Figure 41.

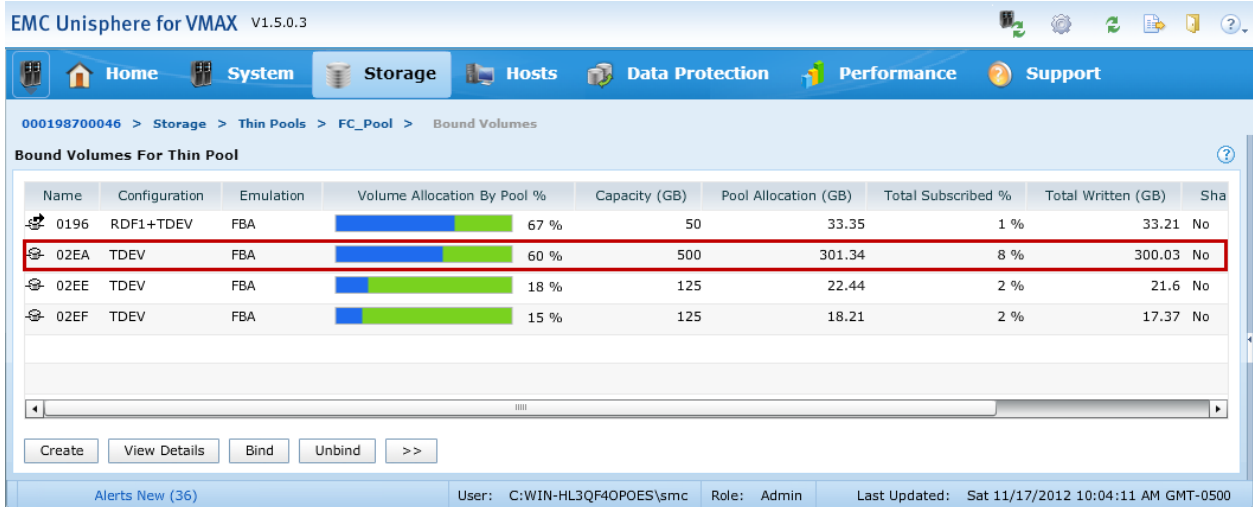


Figure 41. Obtaining thin pool information in EMC Unisphere for VMAX

Note that the *Total Written (GB)* column in Figure 41 shows 300 GB, just as in VSI.

Now using Storage vMotion one of these virtual machines will be moved over to a new datastore. Figure 42 shows both 150 GB VMs with the target of the move, VM UNMAP_150G_VM2, highlighted.

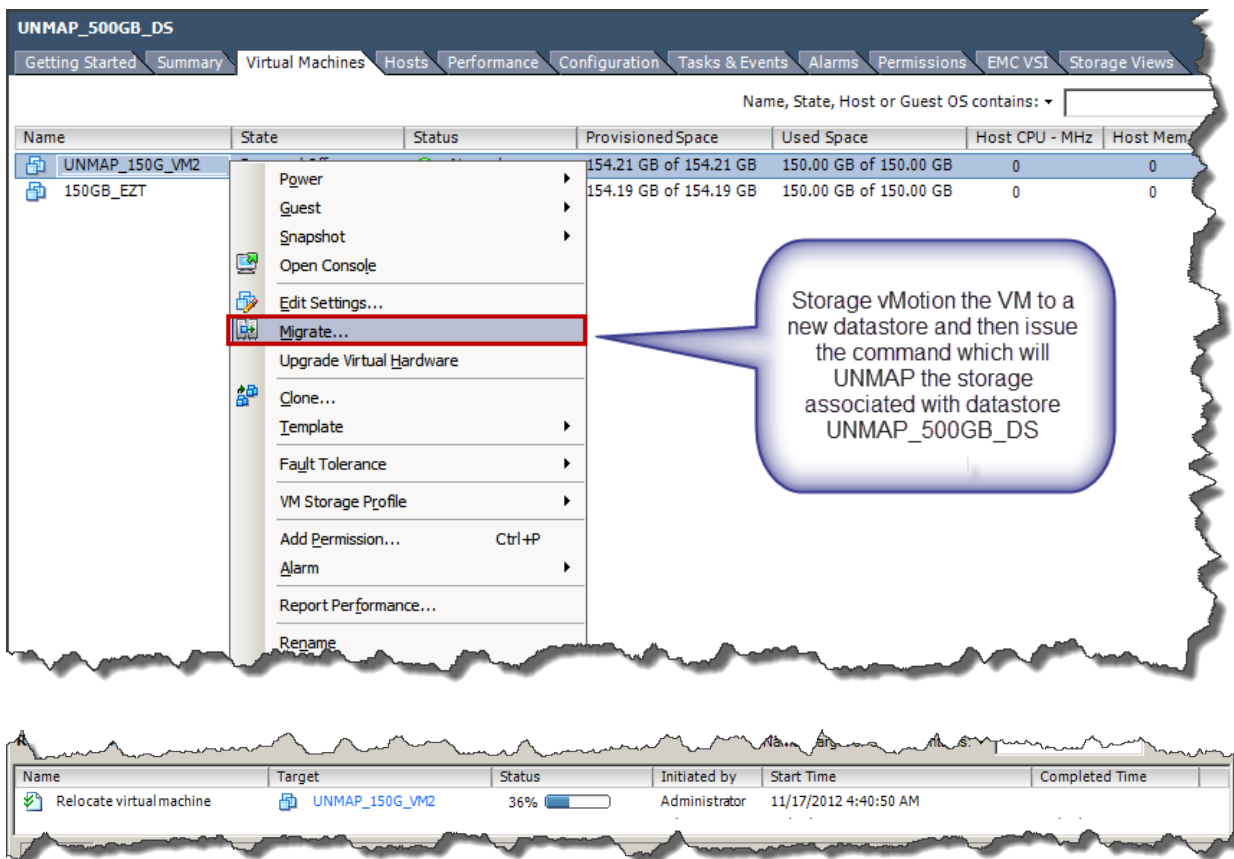


Figure 42. Storage vMotion a 150 GB virtual machine

When the migration completes, the datastore now shows that 150 GB has returned to the free space pool for a total of about 350 GB. The problem, though, is the free space is only on the VMware-side, the VMAX has not freed up that space in the thin pool. Using VSI both views can be seen in Figure 43: the thin device still maintains 300 GB of usage in the FC_Pool while the datastore only shows 150 GB in use.

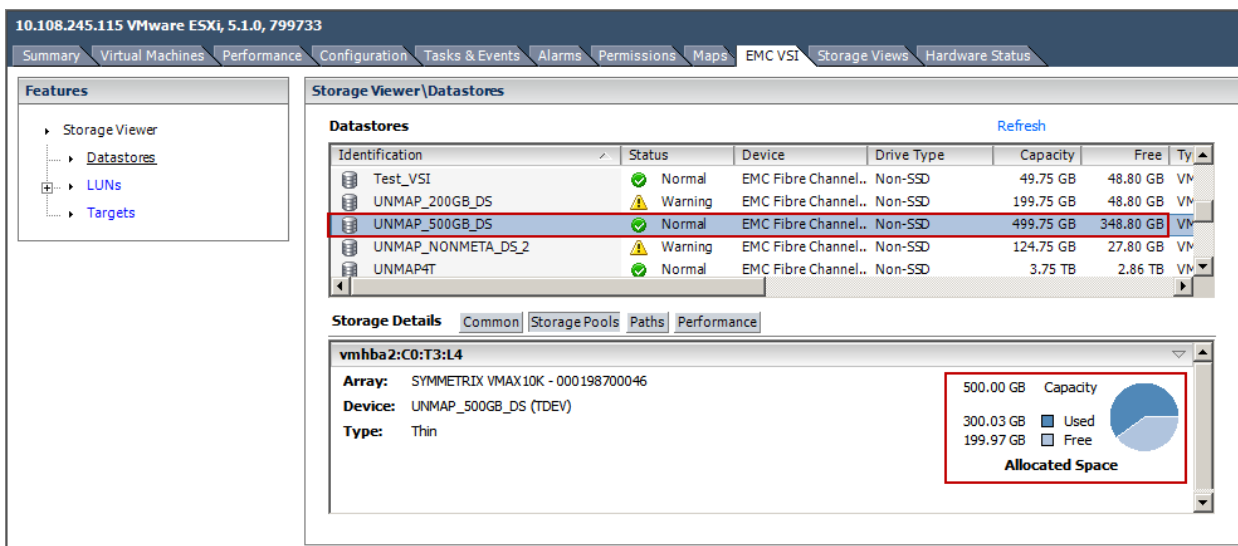


Figure 43. EMC VSI's view of datastore and VMX thin pool usage

At this point space can be reclaimed using the UNMAP primitive. As mentioned before, this particular Storage API is not currently automated until vSphere 6.5. In ESXi 5.0 U1 and 5.1 VMware has provided the ability to issue a command using vmkfstools that tells the VMX what range of blocks can be unmapped. Figure 44 demonstrates the vmkfstools command with the UNMAP switch “-y” and a percentage between 1 and 99 inclusive. In this example ‘99’ is used to reclaim as much space as possible as this datastore is idle.

VMware recommends using a smaller percentage than 99 so as not to impact any other activity on the datastore. Using a high number such as 99% will mean that during the reclaim there will be no available storage for creating VMs or other activities like Storage vMotion into the datastore. Both those actions would fail due to a lack of space.

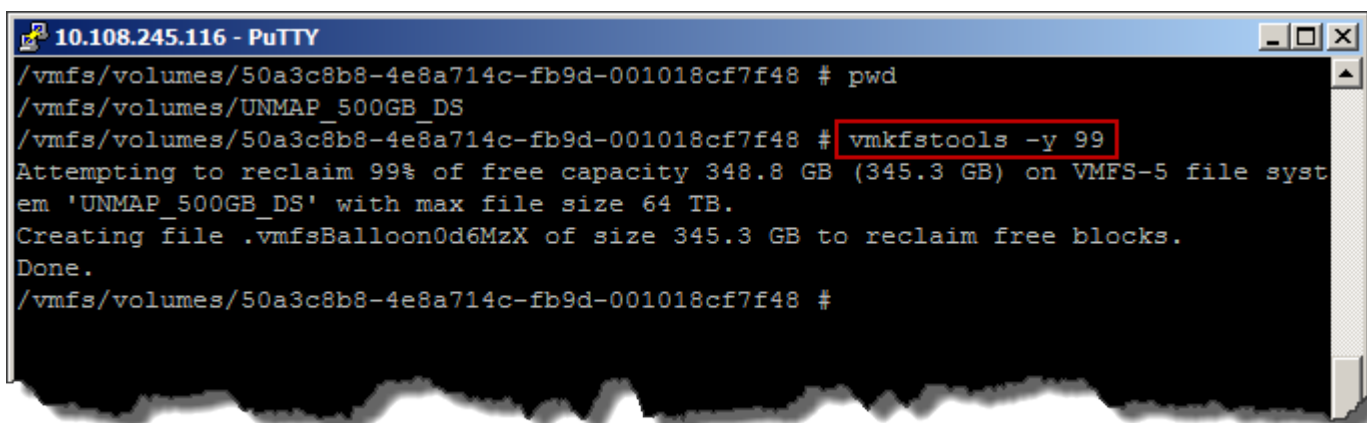


Figure 44. Reclaiming space (UNMAP) through vmkfstools in ESXi 5.0 U1, 5.1

The amount of free space being reclaimed will dictate how long the command takes to run. Reclaiming the 350 GB in this datastore took a little over three minutes. The UNMAP process is a synchronous task so when VMware shows it being “Done”, the data is unmapped and reclaimed on the VMAX. Back in VSI, running a refresh in the Storage Viewer updates the thin pool usage. The actual disk space used on the VMAX is now similar to the free space reported in the datastore. The free space has been returned to the FC_Pool for use by any of the devices bound to it, not just the 2EA device backing the UNMAP_500GB_DS datastore, as shown in Figure 45.

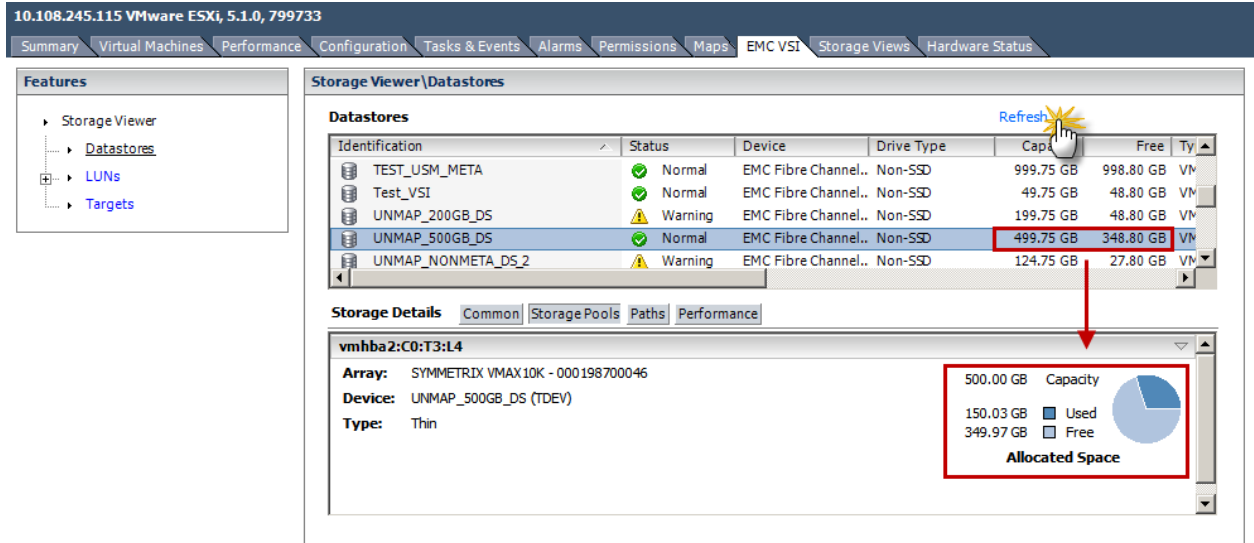


Figure 45. VSI after reclaiming space in the thin pool

Differences in ESXi 5.5+

The process for reclaiming space in ESXi 5.5+ using the previous example is exactly the same, save for the command to issue on the CLI as shown in Figure 46.

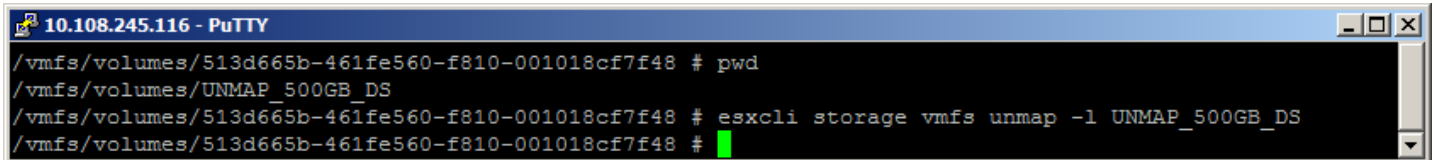


Figure 46. Reclaiming space (UNMAP) through esxcli storage vmfs in ESXi 5.5+

VSI also offers the ability to execute this UNMAP command, or even schedule it through the GUI in Figure 47.

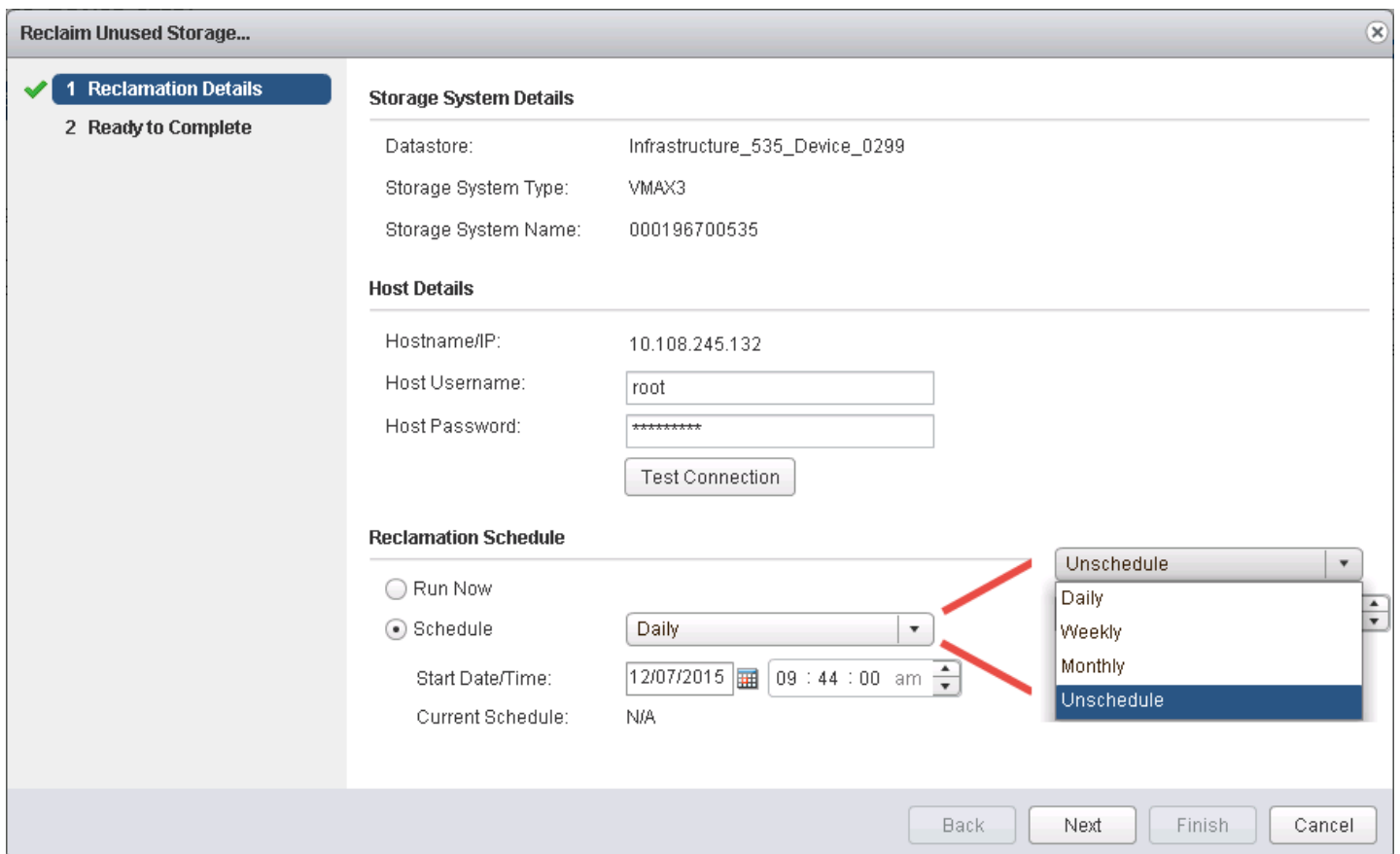


Figure 47. Reclaiming space (UNMAP) through VSI

How the space is actually reclaimed with this command, whether through CLI or GUI, is a bit different. With the former implementation of vmkfstools, a temp file(s) was created at the start of the operation and then an UNMAP was issued to all the blocks that were allocated to these temp files. Since the temp file(s) could take up all the free blocks in the volume, any other operation that required a new block allocation would fail. To prevent that failure, VMware permitted the user to provide a percentage of the free blocks to reclaim. The user could then issue the command multiple times to reclaim all of the space.

In ESXi 5.5+, dead space is now reclaimed in multiple iterations instead of all at once. A user can now provide the number of blocks (the default is 200) to be reclaimed during each iteration. This specified block count controls the size of each temp file that VMware creates. Since vSphere 5.5+ 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. The benefit now is that the risk of temporarily filling up available space on a datastore due to a large balloon file is essentially zero.

As before, this command uses the UNMAP primitive to inform the array that these blocks in this temporary file can be reclaimed. This enables a correlation between what the array reports as free space on a thin-provisioned datastore and what vSphere reports as free space. Previously, there was a mismatch between the host and the storage regarding the reporting of free space on thin-provisioned datastores.

Unlike the previous unmap methodology with vmkfstools, VMware advises that the new procedure in ESXi 5.5+ can be run at any time, and not relegated to maintenance windows; however, EMC still suggests reclaiming space in off-peak time periods.

EMC recommends using VMware's default block size number (200), though fully supports any value VMware does. EMC has not found a significant difference in the time it takes to unmap by increasing the block size number.

ESXi 5.5 P3+ and ESXi 6.x

Beginning with ESXi 5.5 Patch 3 (build 2143827), VMware changes UNMAP behavior once again. At this code version, when issuing an UNMAP, any time a non-default block value is used that is higher than 1% of the free space in the datastore, VMware will revert to the default size (200). In addition, if the datastore is 75% full or more, it will also revert to the default. To determine the most ideal value to pass to VMware, therefore, calculate the amount of free space in the datastore in MB and multiply it by .01 and round down. Failing to round down may result in a number that exceeds 1% and VMware will revert to the default. Using this methodology is most beneficial when there is a large amount of free space to reclaim as the VMAX3/VMAX All Flash is very fast to UNMAP even at the default. Free space can be determined through the vSphere Client or even a PowerCLI script. Figure 48 demonstrates the benefit of using 1% over the default.

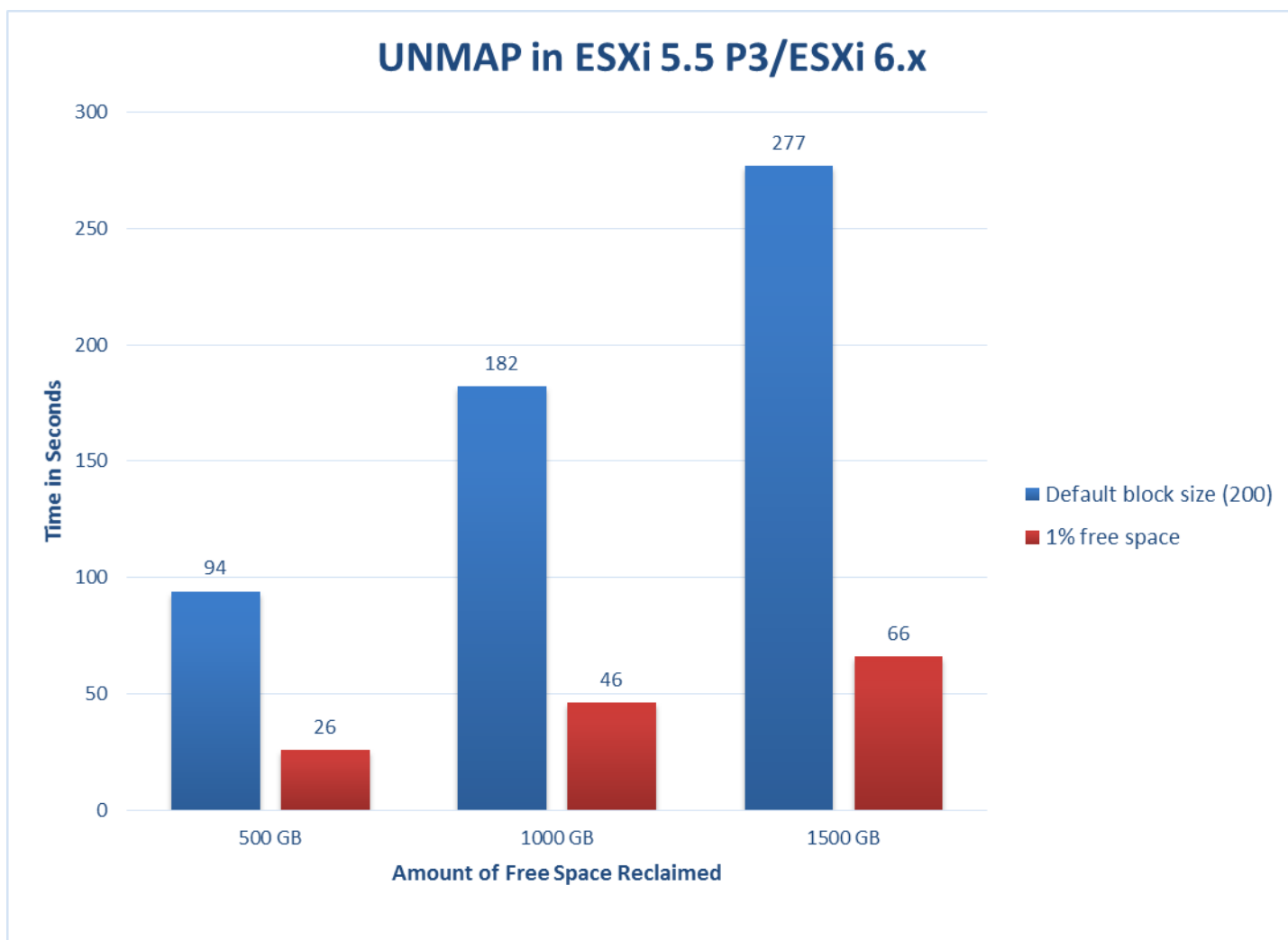


Figure 48. UNMAP in ESXi 5.5 P3+/ESXi 6.x

Note that using 1% has the same scaling as the default, e.g. double the storage reclaimed takes about double the time.

Using UNMAP with devices over 2 TB in ESXi 5.0 and 5.1

VMware has a limitation with the UNMAP mechanism prior to ESXi 5.5 that prevents vmkfstools from reclaiming more than 2 TB in a single operation. The reason for the problem appears to be that VMware cannot create a temporary balloon file beyond 2 TB. If an attempt is made to reclaim that amount of space with a single vmkfstools command, ESXi will generate errors, one of which is shown in Figure 49. This command should be cancelled with a CNTRL-C so VMware can properly clean-up any files.

```
10.108.245.115 - PuTTY
/vmfs/volumes/50a3c8e0-a147f7f0-05c6-001018cf7f48 # pwd
/vmfs/volumes/UNMAP4T
/vmfs/volumes/50a3c8e0-a147f7f0-05c6-001018cf7f48 # vmkfstools -y 99
Attempting to reclaim 99% of free capacity 3.4 TB (3.4 TB) on VMFS-5 file system 'UNMAP4T' with max
file size 64 TB.
Creating file .vmfsBalloon2TEen8L of size 3.4 TB to reclaim free blocks.
Could not set file .vmfsBalloon2TEen8L length to 3.4 TB (Cannot allocate memory).
User Interrupt: Cleaning up reclaim files.
/vmfs/volumes/50a3c8e0-a147f7f0-05c6-001018cf7f48 #
```



Figure 49. Attempting to UNMAP more than 2 TB

If the space to be reclaimed is greater than 2 TB, simply issue the `vmkfstools` command with a smaller percentage multiple times to complete the UNMAP as in Figure 50.

```
10.108.245.115 - PuTTY
/vmfs/volumes/50a3c8e0-a147f7f0-05c6-001018cf7f48 # pwd
/vmfs/volumes/UNMAP4T
/vmfs/volumes/50a3c8e0-a147f7f0-05c6-001018cf7f48 # vmkfstools -y 99
Attempting to reclaim 99% of free capacity 3.4 TB (3.4 TB) on VMFS-5 file system 'UNMAP4T' with max
file size 64 TB.
Creating file .vmfsBalloonmKkCt7 of size 3.4 TB to reclaim free blocks.
Could not set file .vmfsBalloonmKkCt7 length to 3.4 TB (Cannot allocate memory).
User Interrupt: Cleaning up reclaim files.
/vmfs/volumes/50a3c8e0-a147f7f0-05c6-001018cf7f48 # vmkfstools -y 50
Attempting to reclaim 50% of free capacity 3.4 TB (1.7 TB) on VMFS-5 file system 'UNMAP4T' with max
file size 64 TB.
Creating file .vmfsBalloonWGOBHJ of size 1.7 TB to reclaim free blocks.
Done.
/vmfs/volumes/50a3c8e0-a147f7f0-05c6-001018cf7f48 # vmkfstools -y 50
Attempting to reclaim 50% of free capacity 3.4 TB (1.7 TB) on VMFS-5 file system 'UNMAP4T' with max
file size 64 TB.
Creating file .vmfsBalloonakpDRs of size 1.7 TB to reclaim free blocks.
Done.
/vmfs/volumes/50a3c8e0-a147f7f0-05c6-001018cf7f48 #
```

Figure 50. UNMAP more than 2 TB with multiple operations

Following now are a number of use cases which are primarily based upon the device configuration on the VMAX and how the time to reclaim is impacted by it.

Use case configuration

VMAX (V2)

Each use case test in this section is run on a 500 GB thin striped metadvice which is bound to a Fibre Channel pool of disks. The virtual machines that fill the datastore are all comprised of a single, eagerzeroedthick virtual disk which was created without the use of the Block Zero primitive, forcing all zeros to be written to the array. The datastore is completely filled-up, and then a virtual machine in one of three sizes, 75 GB, 150 GB, or 300 GB is deleted. The `vmkfstools -y` command is then run using, in most use cases, 99%. Although VMware recommends using a smaller percentage

when reclaiming to prevent out-of-space conditions in the datastore, the best practice would be to run a reclaim during a maintenance window and as such using 99% would not be unwarranted if the datastore and underlying device were idle.

VMAX3/VMAX All Flash (V3)

As the V3 does not have metadevices, a single 500 GB device was created using an “Optimal” SLO. In addition, because of architectural changes on the V3, each vmdk of the VM was filled-up manually rather than using `eagerzeroedthick` without the Block Zero primitive. The V3 was also tested only on ESXi 5.5 and 6 which demonstrated similar results.

The results in the use cases are only meant as examples of reclaim operations and are dependent upon the lab environment under which they were generated. Although every effort is made to mimic the typical activity in a customer environment it is important to understand that results will vary. Note that although only the results from the pre-ESXi 5.5 `vmkfstools` command is shown for all but the first use case, the `esxcli storage vmfs unmap` command in vSphere 5 and 6 demonstrates similar scaled results, though on the V2 the new process takes longer to complete due to the asynchronous nature of VMware’s implementation. The V3 architecture means its performance exceeds both UNMAP methodologies on the V2.

Use Case 1: UNMAP baseline on a standard thin device

The first use case is to run the reclaim operation on the device without any other relationships or outside environmental impacts. This will be considered the baseline and the results for the ESXi 5.1 will be used for the purpose of comparison in the other tests. The results in Figure 51 demonstrate that the time it takes to reclaim space increases fairly consistently as the amount of space increases. This is true for the VMAX (V2) and the VMAX3/VMAX All Flash (V3) platform. For instance, to reclaim 150 GB at ESXi 5.1 takes 100 seconds while to reclaim 300 GB at takes 202 seconds, almost exactly double the time for double the size. The results for the other tests are similar.

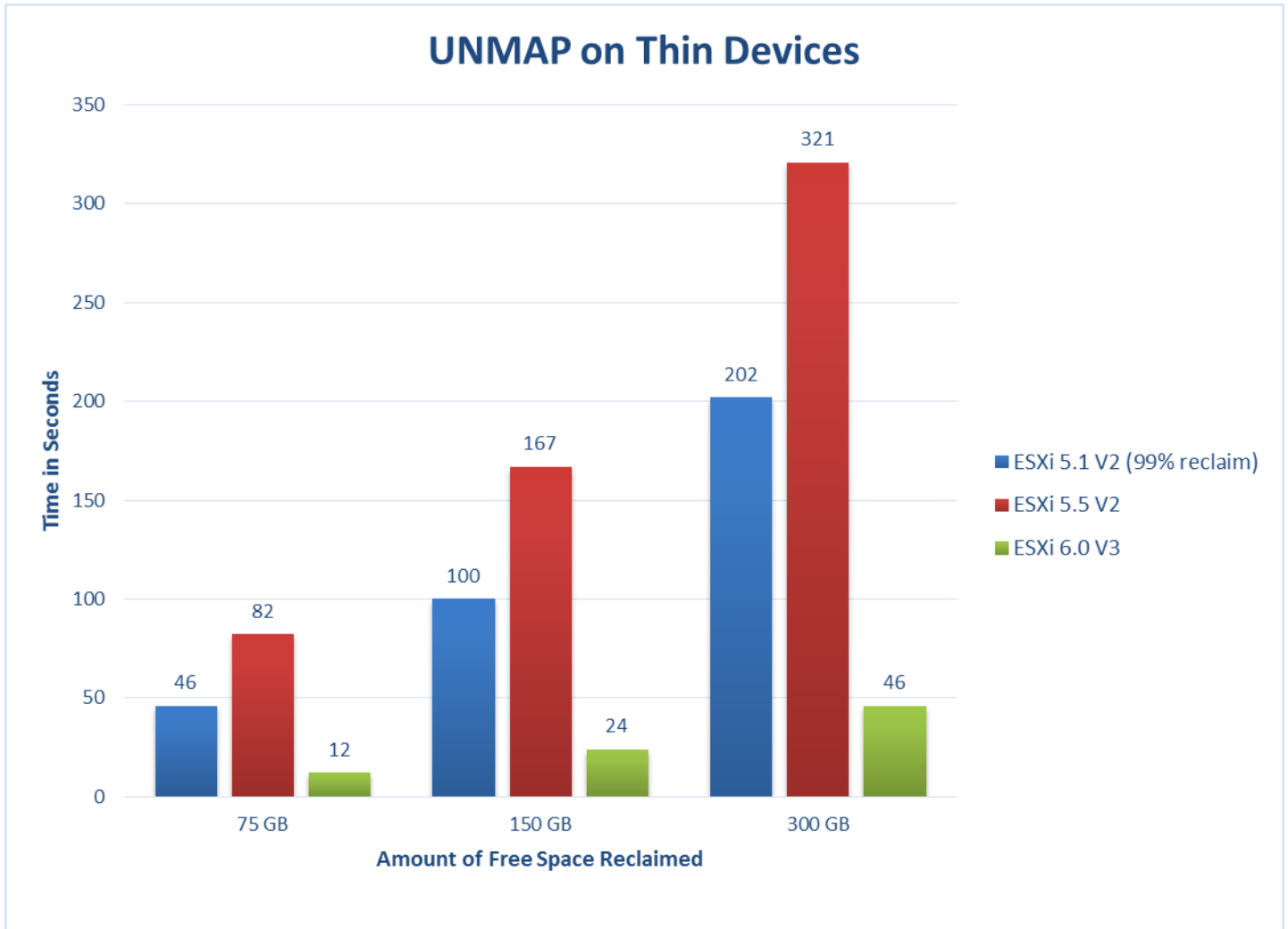


Figure 51. UNMAP on thin devices

Use Case 2: UNMAP with SRDF devices

The second use case is for devices that are in an SRDF relationship, either synchronous or asynchronous. The results are presented together in the same graph. Only the results from a 99% space reclaim are shown in Figure 52.

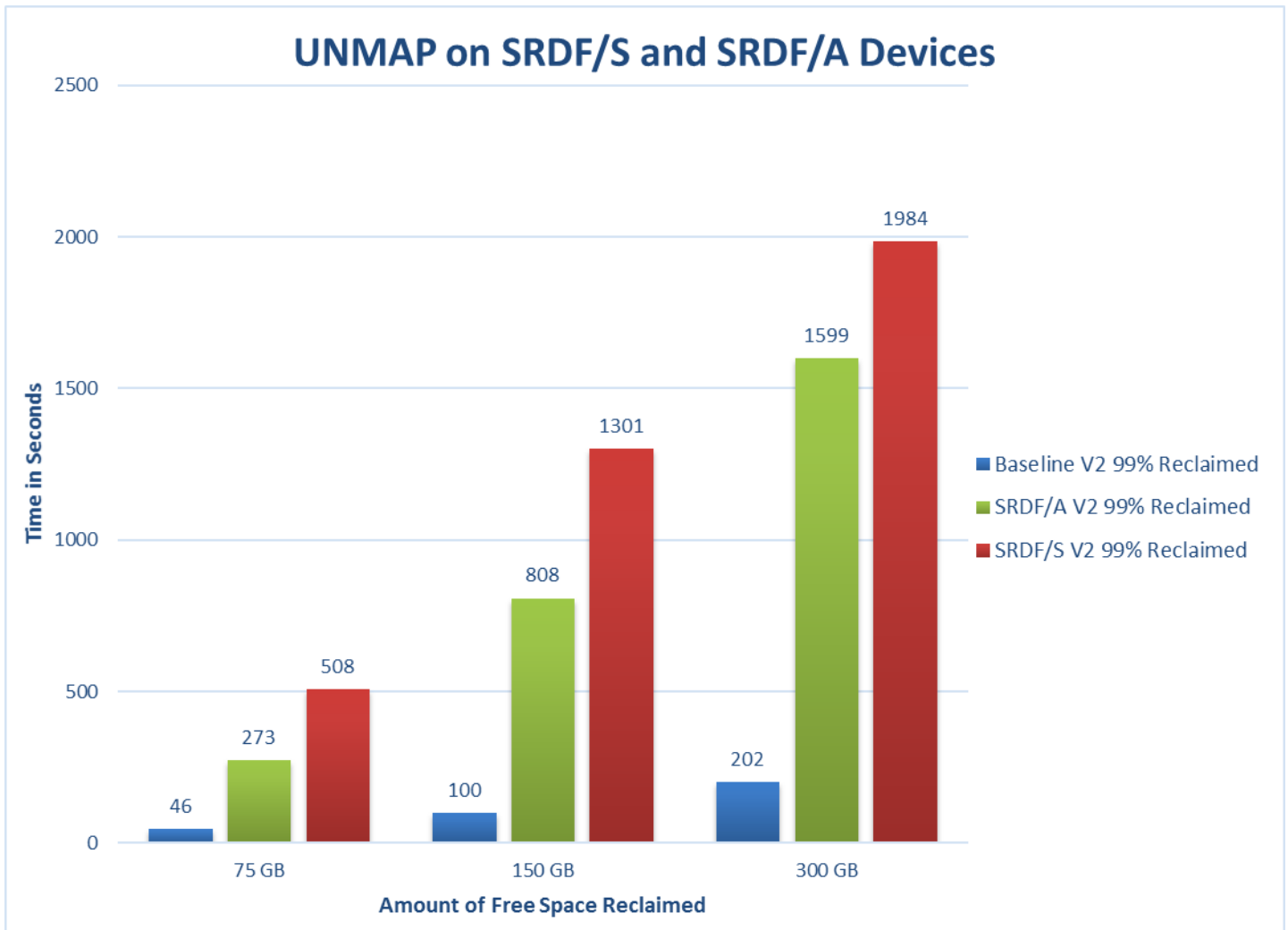


Figure 52. UNMAP on SRDF devices

As might be expected, since SRDF/A batches changes and sends them over asynchronously, the time to UNMAP takes less time on SRDF/A devices than on the synchronous SRDF/S devices; however both experience significant impact from the process in comparison to the baseline. This again points to the importance of running UNMAP during maintenance windows. Note also that unlike the baseline tests, there is more variability in the results for the different amounts of space reclaimed. This will also hold true for the next use case.

Use Case 3: UNMAP with TimeFinder/Clone device

In customer VMAX environments it is not uncommon for devices used in production VMware environments to have TimeFinder/Clone relationships. Therefore a test was run on this type of device. In general, as Figure 53 shows, the time it takes to reclaim space on a device with a TimeFinder/Clone relationship is a little more than double that of a device without the relationship.

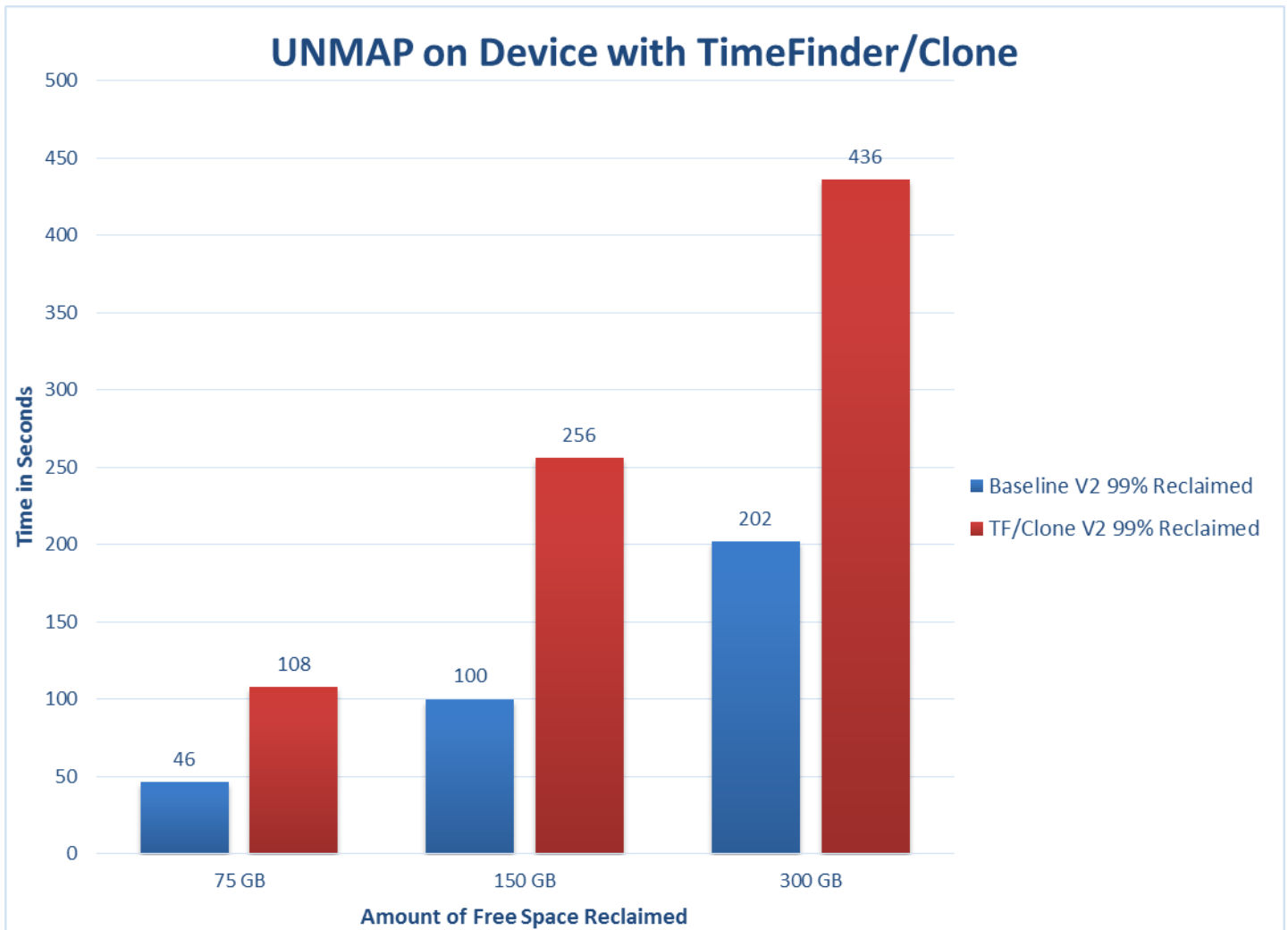


Figure 53. UNMAP with TimeFinder/Clone device

Use Case 4: UNMAP with device under high FA load

The final use case utilizes the baseline thin device but loads the front-end ports to see if there is any noticeable impact to the reclaim process. The FAs were loaded to about 50% as seen in Figure 54.

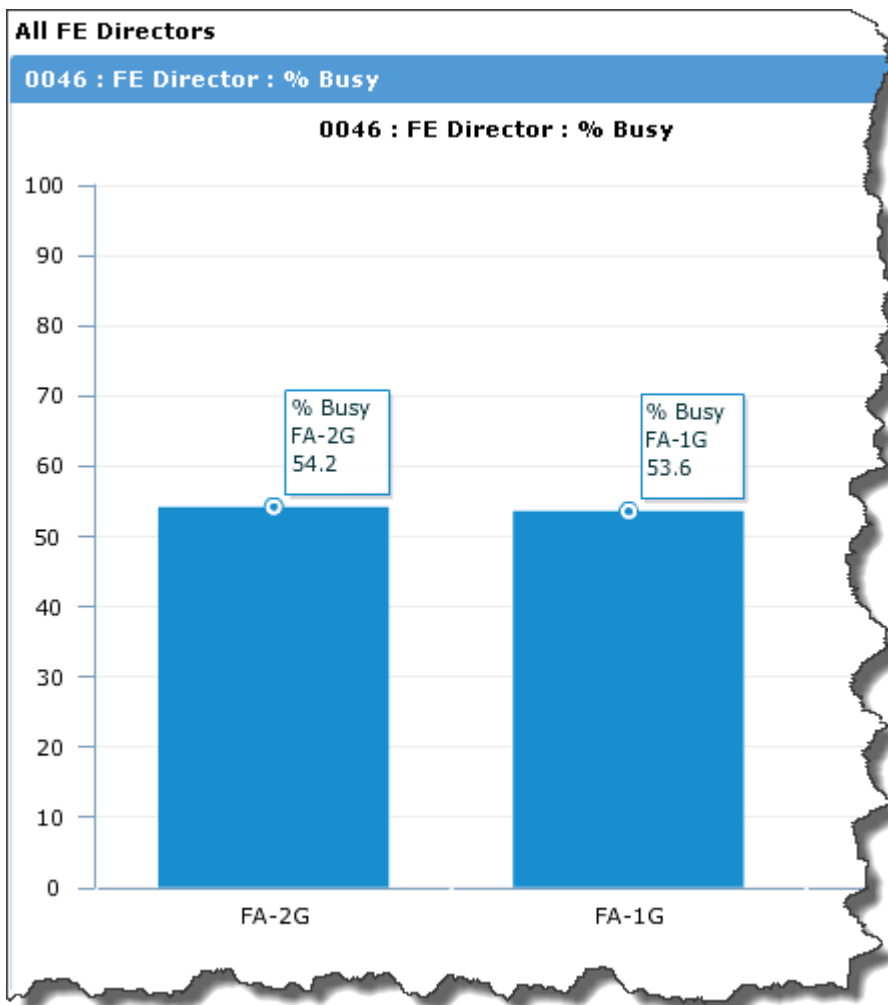


Figure 54. Unisphere for VMAX showing FA load

While the FAs remain loaded, the reclaim process was run. The results in Figure 55 demonstrate a relatively minor impact, which nevertheless one could expect to become more significant if the FA load continued to rise.

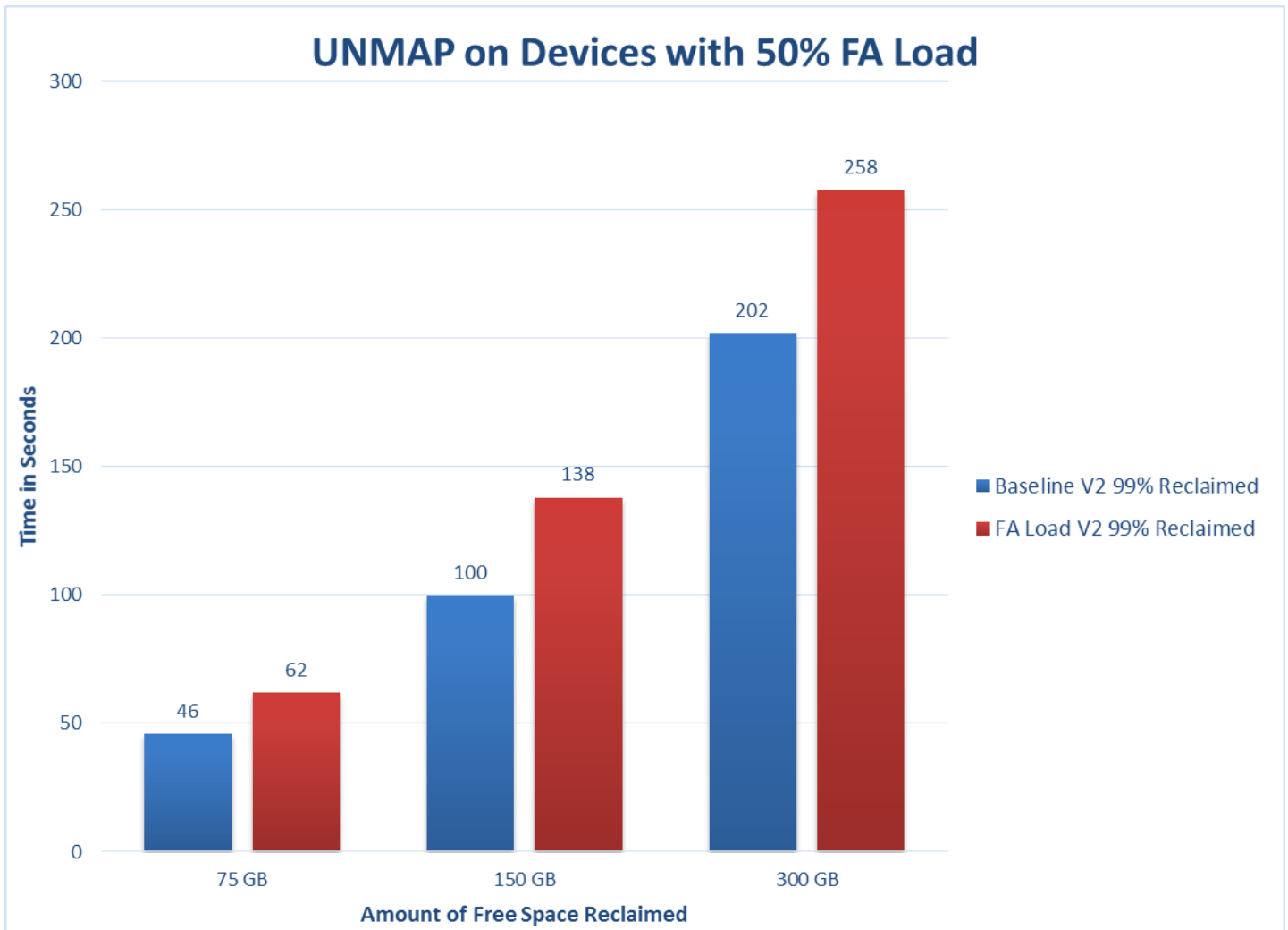


Figure 55. UNMAP on device while under FA load

Caveats for using UNMAP

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

- Not supported on devices in use by Open Replicator for VMAX (ORS)
- Not supported with devices in use by RecoverPoint
- Not supported for encapsulated FTS devices
- Can be partially blocked by thin devices that have persistent allocations. The VMAX accepts the UNMAP command but respects persistent allocations. If the specified UNMAP range has persistent and non-persistent allocations, the VMAX will de-allocate the non-persistent allocations and leave the persistent allocations.
- Not supported with virtual devices (TimeFinder/Snap targets)

- Not supported with devices currently involved in duplicate write sessions (this is a temporary state; these sessions are used during meta reconfiguration and expansion)
- Not supported with TimeFinder/VP Snap targets

UNMAP to a thin SRDF device is accepted and supported on the R1 and is propagated to its remote replicas (R21, R2) as long as the links are active and the remote boxes are also running a minimum of Engenuity 5876.159.102.

In addition to these caveats, here are a few particular situations to be aware of when using UNMAP:

- Although there are no support issues using the Full Copy primitive with the UNMAP primitive, if there is an active XCOPY session from a source to target device and a reclaim is run on the source device, operational limits may be reached which will throttle the UNMAP process, increasing the time to complete it. Such a situation could occur if a Storage vMotion is run and then a reclaim is issued immediately afterwards on the source device now that the space is released.
- Attempting to reclaim a large amount of free space may result in the memory heap error shown in Figure 56:

```

10.108.245.116 - PuTTY
/vmfs/volumes/50a236c1-12b5bbee-c37e-001018cf7f48 # vmkfstools -y 99
Attempting to reclaim 99% of free capacity 3.7 TB (3.7 TB) on VMFS-5 file system 'UNMAP_4TB_DS' with max file size 64 TB
Creating file .vmfsBalloonCluQ1x of size 3.7 TB to reclaim free blocks.
Could not set file .vmfsBalloonCluQ1x length to 3.7 TB (Cannot allocate memory).

User Interrupt: Cleaning up reclaim files.
/vmfs/volumes/50a236c1-12b5bbee-c37e-001018cf7f48 #

```

Figure 56. Experiencing a memory heap error during an UNMAP operation

In order to resolve this follow VMware KB article 1004424: An ESXi/ESX host reports VMFS heap warnings when hosting virtual machines that collectively use 4 TB or 20 TB of virtual disk storage.

- If you must cancel an UNMAP job (vmkfstools -y) after it has started, only use CNTRL-C which will signal VMware to clean-up the temporary file it creates during the reclaim process. If the UNMAP job is killed or cancelled in any other manner, the hidden file will not be removed. More importantly, however, VMware will keep the datastore locked even if the file is manually removed and will not release the space occupied by that hidden file. The only resolution to this is to reboot the ESXi host.

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 of time. 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 57) which will list the storage devices and their live statistics.

```
svpg-dell-c2-s03.ebc.emc.local - PuTTY
11:15:12pm up 13 days 1:03, 275 worlds, 1 VMs, 2 vCPUs; CPU load average: 0.03,
0.03, 0.03
DEVICE                                PATH/WORLD/PARTITION  DQLEN  WQLEN  ACTV  QU
mpx.vmhba32:CO:T0:L0                  -                    1      -      0
mpx.vmhba33:CO:T0:L0                  -                    1      -      0
naa.5000cca0004a7ca8                  -                    64     -      0
naa.60000970000195701238533030303644 -                    64     -      0
naa.60000970000195701238533030303744 -                    64     -      0
naa.60000970000195701238533030303844 -                    64     -      0
naa.60000970000195701238533030303944 -                    64     -      0
naa.60000970000195701238533030304144 -                    64     -      0
naa.60000970000195701238533030304145 -                    64     -      0
naa.60000970000195701238533030304146 -                    64     -      0
naa.60000970000195701238533030304230 -                    64     -      0
naa.60000970000195701238533030304231 -                    64     -      0
```

Figure 57. 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 but unnecessary fields for VAAI purposes were disabled by toggling their asterisk by typing their respective lower-case letters. These selections can be seen in Figure 58.

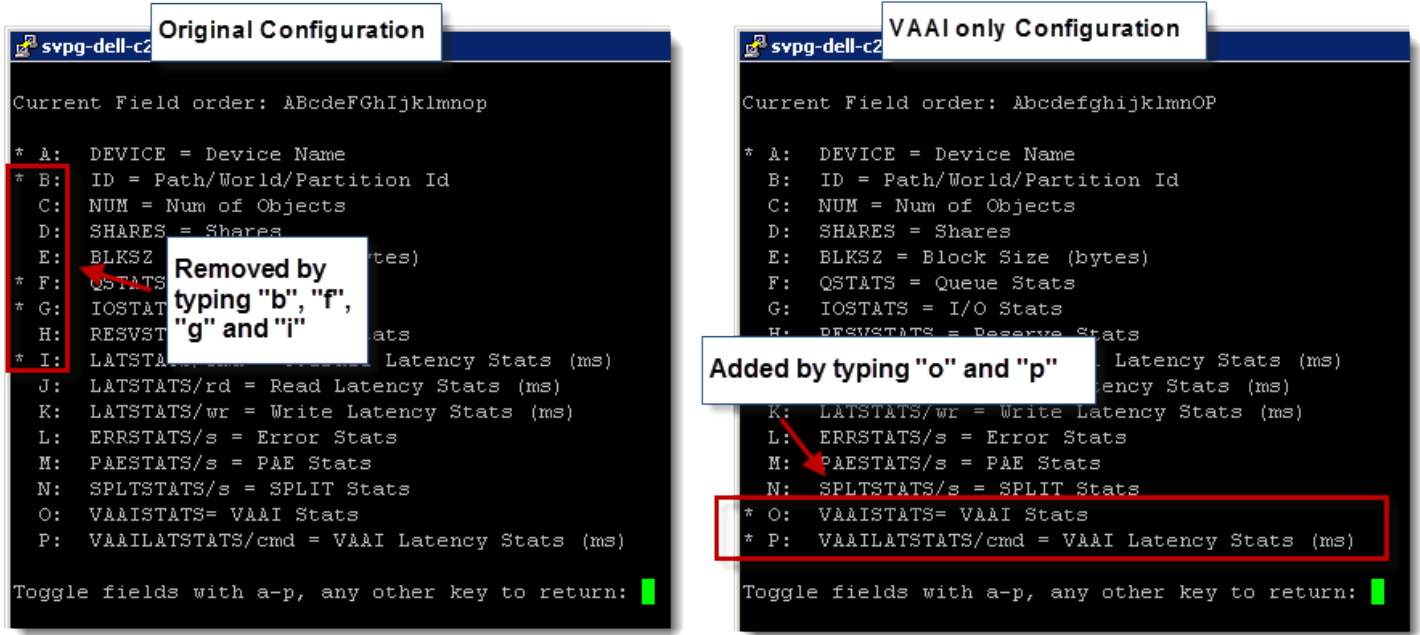


Figure 58. Adding VAAI statistics

Once the statistics desired have been added and others removed, return to the original screen by pressing return or enter on the keyboard. Figure 59 displays the VAAI statistic columns while

Table 2 defines them.

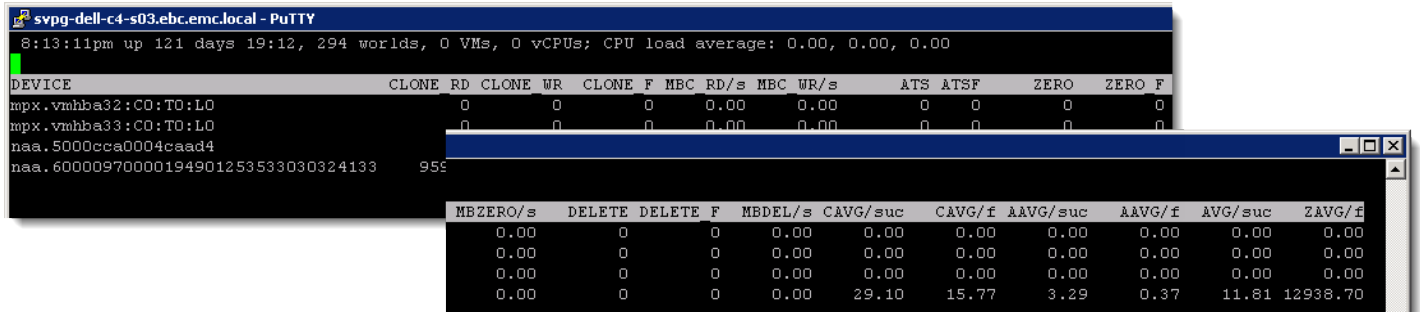


Figure 59. ESXTOP VAAI Statistic counters

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 ²³	Successful UNMAP commands
DELETE_F	Failed UNMAP commands
MBDEL/s	UNMAP throughput in MB per second

It is important to note that with the “VAAIStats” counters some 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 typing another time in seconds and pressing enter) so if a VAAI command has not been run in the last interval the

²³ Note that since UNMAP was introduced in ESX 5.x there are no statistics in ESX 4.x ESXTOP for UNMAP and therefore these columns do not exist.

number will be zero. All of the other commands are running totals since the last reboot.

A common use case for using ESXTOP VAAI statistics is to find out if a XCOPY session has been reverted to software copy due to an exhaustion of resources for queued data at a VMAX device. By watching the CLONE_F column for the given device in a XCOPY session a user can watch for failed XCOPY commands to appear²⁴. If the CLONE_F number increments and the CLONE_RD/CLONE_WR statistics stop incrementing for that device it is likely the resource ceiling has been breached. If the CLONE_RD/CLONE_WR stops incrementing and the CLONE_F does not increase this means the entire XCOPY session completed successfully. Figure 60 shows an ESXTOP example of a XCOPY failure. 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
DEVICE          CLONE RD CLONE WR CLONE F MBC RD/s MBC WR/s
mpx.vmhba32:CO:TO:LO      0      0      0      0.00      0.00
mpx.vmhba33:CO:TO:LO      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 60. ESXTOP XCOPY Statistics

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 available statistics.

Table 3. ESXTOP VAAI Latency Statistics

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

²⁴ 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 Plugin (MPP). In general though this number will be low (single digits or low double digits).

	milliseconds.
AAVG/f	The average latency of failed hardware-assisted locking/ATS commands in milliseconds.
ZAVG/suc ²⁵	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 61) 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 normal this could indicate resource contention on the SAN fabric or the frontend of the VMAX array. At that point the System Administrator would need to troubleshoot the problem through Unisphere for VMAX or other SAN fabric tools.

```

svpg-dell-c4-s03.ebc.emc.local - PuTTY
11:07:34pm up 117 days 22:07, 296 worlds, 0 VMs, 0 vCPUs; CPU load average: 0.01, 0.01, 0.01
DEVICE                CAVG/suc  CAVG/f  AAVG/suc  AAVG/f  AVG/suc  ZAVG/f
mpx.vmhba32:CO:TO:LO  0.00     0.00    0.00     0.00    0.00     0.00
mpx.vmhba33:CO:TO:LO  0.00     0.00    0.00     0.00    0.00     0.00
naa.5000cca0004caad4  0.00     0.00    0.00     0.00    0.00     0.00
naa.60000970000194901253533030324133  29.10    15.77    3.48     0.37    11.81    12938.70

```

Figure 61. ESXTOP VAAI latency statistics

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 62 has a sample output of an environment during a clone.

²⁵ There is a display issue with the 5.x version of ESXTOP where ZAVG/suc is actually displayed as AVG/suc. This is fixed in vSphere 6.

```
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...
2015-02-27T17:01:08.147Z [FFFA21A0 info 'Libs' opID=9F570797-000008A8-90-b4-91] EMCNasPlugin: Vst
orage task is in-progress...
2015-02-27T17:01:28.159Z [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 62. Monitoring VAAI commands on NFS

Conclusion

Utilizing VMware's Storage API integrations which permit the offloading of specific VMware operations to EMC's VMAX benefit 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/VMAX configurations, all these primitives are enabled by default and will be utilized on the VMAX without any user configuration or intervention.³

References

EMC

- *Using EMC VMAX Storage in VMware vSphere Environments* TechBook
<http://www.emc.com/collateral/hardware/solution-overview/h2529-vmware-esx-svr-w-symmetrix-wp-ldv.pdf>
- *Implementing VMware Storage API for Storage Awareness with Symmetrix Storage Arrays*
<http://www.emc.com/collateral/software/white-papers/h10630-vmware-vasa-symmetrix-wp.pdf>
- *Virtual Storage Integrator 5.x - 7.x Product Guide* (support.EMC.com)
- *Using VMware Virtual Volumes with EMC VMAX3 and VMAX All Flash*

<http://www.emc.com/collateral/white-papers/h14576-vmware-virtual-volumes-emc-vmax3-vmax-all-flash.pdf>

- *Using EMC VNX Storage with VMware vSphere*

https://support.emc.com/docu33753_Using_VNX_Storage_with_VMWare_vSphere.pdf

VMware

- *vSphere Documentation*

<https://www.vmware.com/support/pubs>