VMware vSphere® Storage APIs – Array Integration (VAAI)

vmware®

Table of Contents

Introduction to VAAI

In a virtualized environment, storage operations traditionally have been expensive from a resource perspective. Functions such as cloning and snapshots can be performed more efficiently by the storage device than by the host.

VMware vSphere® Storage APIs – Array Integration (VAAI), also referred to as hardware acceleration or hardware offload APIs, are a set of APIs to enable communication between VMware vSphere ESXi™ hosts and storage devices. The APIs define a set of "storage primitives" that enable the ESXi host to offload certain storage operations to the array, which reduces resource overhead on the ESXi hosts and can significantly improve performance for storage-intensive operations such as storage cloning, zeroing, and so on. The goal of VAAI is to help storage vendors provide hardware assistance to speed up VMware® I/O operations that are more efficiently accomplished in the storage hardware.

Without the use of VAAI, cloning or migration of virtual machines by the vSphere VMkernel Data Mover involves software data movement. The Data Mover issues I/O to read and write blocks to and from the source and destination datastores. With VAAI, the Data Mover can use the API primitives to offload operations to the array if possible. For example, if the desired operation were to copy a virtual machine disk (VMDK) file from one datastore to another inside the same array, the array would be directed to make the copy completely inside the array. Whenever a data movement operation is invoked and the corresponding hardware offload operation is enabled, the Data Mover will first attempt to use the hardware offload. If the hardware offload operation fails, the Data Mover reverts to the traditional software method of data movement.

In nearly all cases, hardware data movement will perform significantly better than software data movement. It will consume fewer CPU cycles and less bandwidth on the storage fabric. Improvements in performance can be observed by timing operations that use the VAAI primitives and using esxtop to track values such as CMDS/s, READS/s, WRITES/s, MBREAD/s, and MBWRTN/s of storage adapters during the operation.

In the initial VMware vSphere 4.1 implementation, three VAAI primitives were released. These primitives applied only to block (Fibre Channel, iSCSI, FCoE) storage. There were no VAAI primitives for NAS storage in this initial release.

In vSphere 5.0, VAAI primitives for NAS storage and VMware vSphere Thin Provisioning were introduced.

VAAI Block Primitives

Atomic Test & Set (ATS)

In VMware vSphere VMFS, many operations must establish a lock on the volume when updating a resource. Because VMFS is a clustered file system, many ESXi hosts can share the volume. When one host must make an update to the VMFS metadata, a locking mechanism is required to maintain file system integrity and prevent another host from coming in and updating the same metadata. The following operations require this lock:

1. Acquire on-disk locks
2. Upgrade an optimistic lock to an exclusive/physical lock
3. Unlock a read-only/multiwriter lock
4. Acquire a heartbeat
5. Clear a heartbeat
6. Replay a heartbeat
7. Reclaim a heartbeat
8. Acquire on-disk lock with dead owner

It is not essential to understand all of these operations in the context of this whitepaper. It is sufficient to understand that various VMFS metadata operations require a lock.

ATS is an enhanced locking mechanism designed to replace the use of SCSI reservations on VMFS volumes when doing metadata updates. A SCSI reservation locks a whole LUN and prevents other hosts from doing metadata updates of a VMFS volume when one host sharing the volume has a lock. This can lead to various contention issues when many virtual machines are using the same datastore. It is a limiting factor for scaling to very large VMFS volumes. ATS is a lock mechanism that must modify only a disk sector on the VMFS volume. When successful, it enables an ESXi host to perform a metadata update on the volume. This includes allocating space to a VMDK during provisioning, because certain characteristics must be updated in the metadata to reflect the new size of the file. The introduction of ATS addresses the contention issues with SCSI reservations and enables VMFS volumes to scale to much larger sizes.

In vSphere 4.0, VMFS3 used SCSI reservations for establishing the lock, because there was no VAAI support in that release. In vSphere 4.1, on a VAAI-enabled array, VMFS3 used ATS for only operations 1 and 2 listed previously, and only when there was no contention for disk lock acquisitions. VMFS3 reverted to using SCSI reservations if there was a multihost collision when acquiring an on-disk lock using ATS.

In the initial VAAI release, the ATS primitives had to be implemented differently on each storage array, requiring a different ATS opcode depending on the vendor. ATS is now a standard T10 SCSI command and uses opcode **0x89** (COMPARE AND WRITE).

For VMFS5 datastores formatted on a VAAI-enabled array, all the critical section functionality from operations 1 to 8 is done using ATS. There no longer should be any SCSI reservations on VAAI-enabled VMFS5. ATS continues to be used even if there is contention. On non-VAAI arrays, SCSI reservations continue to be used for establishing critical sections in VMFS5.

ATS Only Flag

An **ATS only** flag appears on newly created VMFS5 datastores. When the host detects that the array supports ATS on the device, the **ATS only** flag is written to the datastore. From that point on, ATS always will be used on that datastore, so it can be used for all metadata operations.

The flag is not enabled, however, on VMFS5 volumes that were upgraded from VMFS3, facilitating seamless upgrades from VMFS3 to VMFS5. This is because VMFS3 was not ATS complete, and during the upgrade process, some hosts might continue to use SCSI reservations for certain metadata operations.

You cannot turn on the **ATS only** flag on VMFS5 volumes upgraded from VMFS3. The **ATS only** flag can be manually enabled or disabled on a newly created VMFS5. To manually enable it, you can use the hidden option:

```
# vmkfstools —configATSOnly 1 [device path]
```

XCOPY (Extended Copy)

Without VAAI, a clone or migrate operation must use the VMkernel software Data Mover driver. If the files being cloned are multiple GB in size, the operation might last for many minutes to many hours. This full copy consumes host resources such as CPU cycles, DMA buffers and SCSI commands in the HBA queue.

This primitive requests that the array perform a full copy of blocks on behalf of the Data Mover. It primarily is used in clone and migration operations (such as a VMware vSphere Storage vMotion® instance). ESXi hosts can be configured to enable the **EXTENDED COPY** SCSI command. The SCSI opcode for XCOPY is **0x83**.

Write Same (Zero)

When provisioning an **eagerzeroedthick** VMDK, a SCSI command is issued to write zeroes to disk blocks. Again, this initialization request consumes host resources such as CPU cycles, DMA buffers and SCSI commands in the HBA queue.

One of the most common operations on virtual disks is initializing large extents of the disk with zeroes to isolate virtual machines and promote security. ESXi hosts can be configured to enable the **WRITE_SAME** SCSI command to zero out large portions of a disk. This offload task will zero large numbers of disk blocks without transferring the data over the transport link. The **WRITE_SAME** SCSI opcode is **0x93**.

The following provisioning tasks are accelerated by the use of the **WRITE_SAME** command:

• Cloning operations for *eagerzeroedthick* target disks
• Allocating new file blocks for thin-provisioned virtual disks
• Initializing previous unwritten file blocks for *zeroedthick* virtual disks

NOTE: Some storage arrays, on receipt of the WRITE_SAME *SCSI command, will write zeroes directly to disk. Other arrays do not need to write zeroes to every location; they simply do a metadata update to write a page of all zeroes. Therefore, it is possible that you will observe significant differences in the performance of this primitive, depending on the storage array.*

VAAI NAS Primitives

Prior to vSphere 5.0, VMware supported hardware acceleration on block storage devices. As of the vSphere 5.0 release, VMware has also included a number of new NAS hardware acceleration primitives. Unlike block primitives, VAAI NAS primitives will be available only through the use of a NAS vendor plug-in, which is provided by your storage array vendor.

Full File Clone

Full File Clone, or Full Copy, is similar to the Extended Copy (XCOPY) hardware acceleration primitive provided for block arrays. This primitive enables virtual disks to be cloned by the NAS device rather than by using the Data Mover, which consumes ESXi host CPU and memory resources as well as network bandwidth. There is one noticeable difference between this primitive and the XCOPY primitive on block storage: This primitive cannot be called for Storage vMotion operations, which continue to use the Data Mover. Only virtual machines that are powered off and then migrated can offload to the storage array by using this primitive. The Storage vMotion restriction aside, the benefit is that cold clones or "deploy from template" operations can be offloaded to the storage hardware, and this reduces the resource consumption of the ESXi host.

Fast File Clone/Native Snapshot Support

Fast File Clone enables the creation of virtual machine snapshots to be offloaded to an NAS storage array. This is a primitive that forms the basis of VMware View Composer™ Array Integration (VCAI), which was released as a technical preview in VMware View™ 5.1. VMware View Composer can elect to have desktops based on linked clones created directly by the storage array rather than by using ESXi host CPU and memory resources for the same task. In vSphere 5.1, this primitive has been extended to VMware vCloud® vApps™, whereby VMware vCloud Director™ can elect to have vCloud vApps based on linked clones instantiated by the storage array rather than the ESXi host.

Extended Statistics

This primitive enables visibility into space usage on NAS datastores. This is especially useful for thin-provisioned datastores because it enables vSphere to display actual space usage statistics in situations where oversubscription was used. Previously, vSphere administrators needed to use array-based tools to manage and monitor how much space a thinly provisioned VMDK was consuming on a back-end datastore. With this new primitive, this information can be surfaced up in the VMware vSphere Client™. This enables vSphere administrators to have a much better insight into storage resource consumption, without the need to rely on third-party tools or engage with their storage array administrators.

Reserve Space

In previous versions of vSphere, only a thin VMDK could be created on NAS datastores. This new reserve space primitive enables the creation of thick VMDK files on NAS datastores, so administrators can reserve all of the space required by a VMDK, even when the datastore is NAS. As shown in Figure 1, users now have the ability to select lazy-zero or eager-zero disks on NAS datastore.
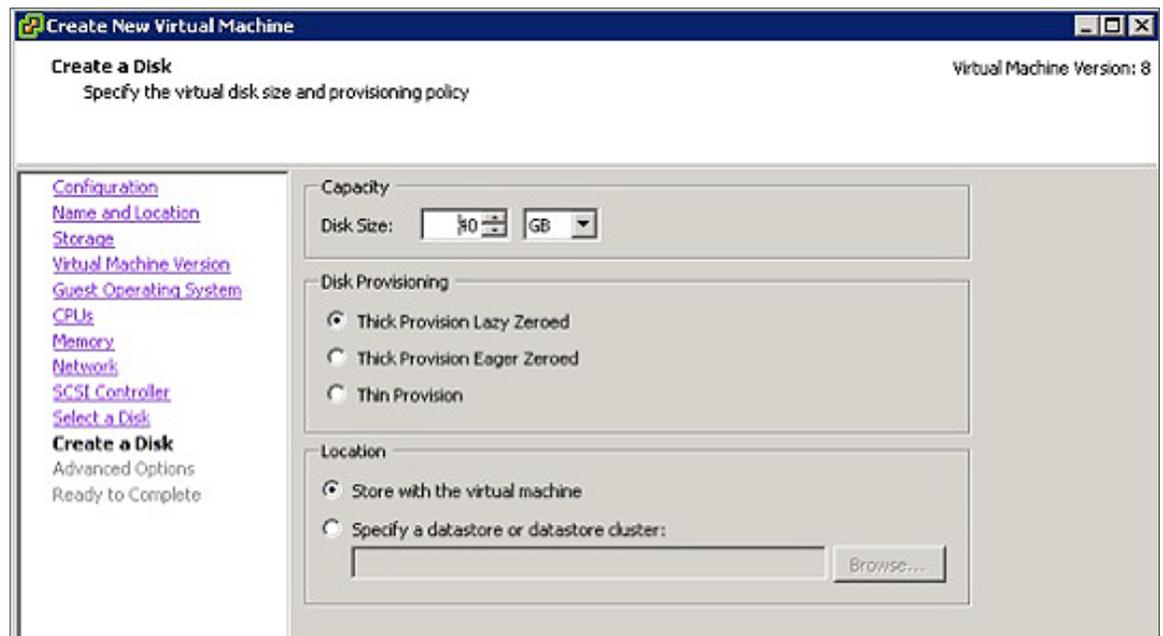

Figure 1. Thick-Disk Provisioning on NFS Datastores

VAAI Thin Provisioning Primitives

In environments where thin provisioning is used in the management and monitoring of datastores, there are frequent complaints that the array is not informed that blocks have become free when virtual machines have been deleted or migrated from a datastore. This leads to array management tools' reporting much higher space consumption than is actually the case. Out-of-space (OOS) concerns are also common. In the past, OOS conditions could impact all virtual machines on the OOS thin-provisioned datastore.

To address these concerns, multiple enhancements were made to VAAI in vSphere 5.0. In particular, several new primitives have been added to better support Thin Provisioning.

Thin Provisioning Stun

The first enhancement was introduced to address concern regarding the impact on virtual machines when thin-provisioned datastore usage reaches 100 percent of capacity. Historically, this affected all virtual machines running on the datastore. With the release of the vSphere 5.0 VAAI Thin Provisioning primitives, if a thin-provisioned datastore reaches 100 percent usage, only those virtual machines requiring extra blocks of storage space are paused; those not needing additional space continue to run. After additional space is allocated to the thin-provisioned datastore, the paused virtual machines can be resumed.

Thin Provisioning Space Threshold Warning

vSphere cannot detect how much space a thin-provisioned datastore is consuming at the back end. The first indication a vSphere administrator has that an overcommitted datastore has no available space is that no more virtual machines can be provisioned and other virtual machines stop running. With this new primitive, a warning is raised and surfaced in VMware vCenter™ via VAAI if a thin-provisioned datastore reaches a specific threshold, as is shown in Figure 2. The threshold values are set on the array, if there is a means to do so, and are defined by the storage administrator; vSphere does not set this threshold. This enables an administrator to proactively add more storage, extend the datastore, or run a Storage vMotion operation to migrate some virtual machines off of the datastore to avoid OOS conditions.

Figure 2. Thin Provisioning Out-of-Space Warning

Dead Space Reclamation

A new VAAI primitive, using the **SCSI UNMAP** command, enables an ESXi host to inform the storage array that space can be reclaimed that previously had been occupied by a virtual machine that has been migrated to another datastore or deleted. This enables an array to accurately report space consumption of a thin-provisioned datastore and enables users to monitor and correctly forecast new storage requirements.

The mechanism for conducting a space reclamation operation has changed somewhat since the primitive was introduced in vSphere 5.0. Then, the operation was automatic: When a virtual machine was migrated from a datastore or deleted, the UNMAP was called immediately and space was reclaimed on the array. There were some issues with this approach, however, primarily regarding performance and an array's ability to reclaim the space in an optimal time frame. For this reason, the UNMAP operation is now a manual process. Because it is the only manually initiated primitive, the process to reclaim dead space will be discussed in detail.

To reclaim dead space from a thin-provisioned datastore in vSphere 5.1, first verify that the storage array is capable of processing the SCSI UNMAP command. With **esxcli**, users can display device-specific details regarding Thin Provisioning and VAAI.

```
# esxcli storage core device list —d  naa.xxx
naa.xxx
    Display Name: NETAPP Fibre Channel Disk (naa.xxx)
    Has Settable Display Name: true
    Size: 51200
    Device Type: Direct-Access
    Multipath Plugin: NMP
    Devfs Path: /vmfs/devices/disks/naa.xxx
    Vendor: NETAPP
    Model: LUN
    Revision: 8020
    SCSI Level: 4
    Is Pseudo: false
    Status: on
    Is RDM Capable: true
    Is Local: false
    Is Removable: false
    Is SSD: false
    Is Offline: false
    Is Perennially Reserved: false
    Thin Provisioning Status: yes
    Attached Filters: VAAI_FILTER
    VAAI Status: supported
    Other UIDs: vml.xxx
```

The device is indeed thin provisioned and supports VAAI. Next, users can display the VAAI primitives supported by the array for that device, as well as whether the array supports the UNMAP primitive for dead space reclamation (referred to as the **Delete Status**). Another **esxcli** command is used for this step, as is shown in the following:

```
# esxcli storage core  device vaai status get –d  naa.xxx
naa.xxx
    VAAI Plugin Name: VMW_VAAIP_NETAPP
    ATS Status: supported
    Clone Status: supported
    Zero Status: supported
    Delete Status: supported
```

The device displays **Delete Status: supported** to indicate that it is capable of sending SCSI UNMAP commands to the array when a space reclamation operation is requested.

If a Storage vMotion operation is initiated and a virtual machine is moved from a source datastore to a destination datastore, the following actions will reclaim that space: When the Storage vMotion operation has completed, the vSphere Client will report that the VMFS5 volume now has much more free space. The issue is that when the amount of free space is checked on the thin-provisioned LUN backing this VMFS5 volume on the storage array, there is unused and stranded space. In this example, a NetApp FAS array is providing the back-end storage. Using a **lun show** CLI command on this NetApp array, we see in the following that 8.8GB of space is still being consumed, even though there are now no virtual machines or any other files on this datastore.

```
lun show -v /vol/vol2/thin-lun
/vol/vol2/thin-lun      50g (53687091200)   (r/w, online, mapped)
    Serial#: xxx
    Share: none
    Space Reservation: disabled
    Multiprotocol Type: vmware
    Maps: unmap=51 issi=51
    Occupied Size: 8.8g (9473908736)
    Creation Time: Tue Apr 24 15:16:52 BST 2012
    Cluster Shared Volume Information: 0x0
```

This is the crux of the issue that we are trying to solve with the VAAI UNMAP primitive. To run the UNMAP command, change the directory to the root of the VMFS volume that you want to reclaim space from. The following command is run:

vmkfstools –y <% of free space to unmap>

In this example, we attempted a reclamation of 60 percent of free space. The **vmkfstools -y** command displays the following:

```
Attempting to reclaim 60% of free capacity 48.8 GB (29.3 GB) on VMFS-5 file system
'source-datastore' with max file size 64 TB.
Create file .vmfsBalloontsWt8w of size 29.3 GB to reclaim free blocks.
Done.
```

vmkfstools -y created a balloon file of 29.3GB, which is 60 percent of the free capacity (48.8GB). This temporary balloon file is equal to the size of the space to be unmapped/reclaimed.

NOTE: Caution – If a percentage value in the high 90s or 100 is specified, the temporary balloon file that is created during the reclamation operation might fill up the VMFS volume. Any growth of current VMDK files due to running virtual machines writing to their disks or the creation of new files, such as snapshots, might fail due to unavailable space. Care should be taken when calculating the amount of free space to reclaim.

Finally, query the status of the thin-provisioned LUN on the storage array. It should show a difference in **Occupied Size**, as shown in the following:

```
lun show -v /vol/vol2/thin-lun
/vol/vol2/thin-lun    50g (53687091200)   (r/w, online, mapped)
    Serial#: xxx
    Share: none
    Space Reservation: disabled
    Multiprotocol Type: vmware
    Maps: unmap=51 issi=51
    Occupied Size: 76.3m (79966208)
    Creation Time: Tue Apr 24 15:16:52 BST 2012
    Cluster Shared Volume Information: 0x0
```

The dead space successfully has been reclaimed for the thin-provisioned datastore; the array now can correctly report the space usage.

Disabling the UNMAP primitive does not affect any of the other Thin Provisioning primitives such as Thin Provisioning Stun and the space threshold warning. All primitives are orthogonal.

VAAI Implementation

Pluggable Storage Architecture

There are several components required in the VMkernel to make VAAI work correctly. The first of these is the Pluggable Storage Architecture (PSA) device filter framework. This is a feature that is intended to facilitate device-level value-add into the PSA I/O stack. It enables VMware to introduce new value-add into the PSA stack asynchronously, decoupling it from the vSphere release cycle. The primary motivation is to support VAAI.

Without the PSA device filter framework, VAAI would have to be implemented in the storage array type policy (SATP) on a per-array basis. Implementation in the PSA device filter framework reduces duplicate code and saves memory space in the VMkernel. Also, if VAAI were implemented in the SATP, it would prevent other third-party plug-ins, such as EMC PowerPath, from using the offload primitives.

The term "filter" is somewhat misleading because the PSA device filter framework enables specific value-added software to be inserted between VMFS and the PSA device layer.

The second required component can be referred to as a VAAI plug-in specific to the VAAI filter. It implements vendor-specific VAAI functions such as ATS, XCOPY and WRITE_SAME. There were different implementations of the VAAI block primitives in vSphere 4.1, but all of the primitives in vSphere 5.0 have been ratified by T10, so any array that is T10 compliant should be able to use VAAI.
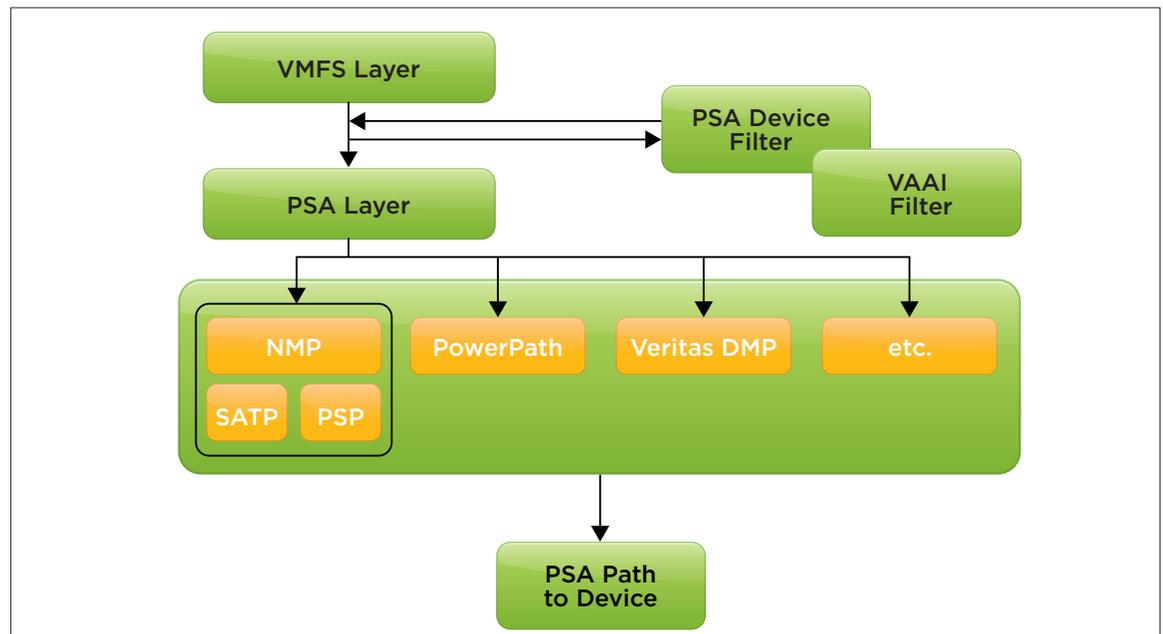
Figure 3. Pluggable Storage Architecture

To leverage VAAI functionality on a LUN, you must have both a PSA device filter and a vendor-specific VAAI plug-in for each device. VMware has a single PSA device filter plug-in called VAAI_FILTER. If a device supports VAAI offloads, it first will be claimed by the VAAI filter, VAAI_FILTER. After it has been claimed by the filter, a vendor-specific VAAI plug-in will claim and then manage the device.

The following commands display the claim rules that are associated with device filters in the PSA.

By using the following command, you can check whether an individual device has been claimed by the VAAI filter. As is shown in the following, the Attached Filters and VAAI Status fields are located toward the end of the output:

```
~ # esxcli storage core device list –d naa.xxx
naa.xxx
    Display Name: NETAPP iSCSI Disk (naa.xxx)
    Has Settable Display Name: true
    Size: 20480
    Device Type: Direct-Access
    Multipath Plugin: NMP
    Devfs Path: /vmfs/devices/disks/naa.xxx
    Vendor: NETAPP
    Model: LUN
    Revision: 8020
    SCSI Level: 4
    Is Pseudo: false
    Status: degraded
    Is RDM Capable: true
    Is Local: false
    Is Removable: false
    Is SSD: false
    Is Offline: false
    Is Perennially Reserved: false
    Thin Provisioning Status: yes
    Attached Filters: VAAI_FILTER
    VAAI Status: supported
    Other UIDs: vml.xxx
```

If a device supports VAAI and has been claimed successfully by the VAAI filter, the filter will appear in the plug-in list command:

```
~ # esxcli storage core plugin list
Plugin name       Plugin class
----------------  -----------
VMW_VAAIP_NETAPP  VAAI
VAAI_FILTER       Filter
NMP               MP
```

In this example, the VAAI filter and the vendor-specific VAAI plug-in are displayed.

Finally, another CLI command can be used to verify which vendor-specific VAAI plug-in has claimed a device. This command shows both the VAAI plug-in name and the primitives that it supports.

```
~ # esxcli storage core device vaai status get –d naa.xxx
naa.xxx
    VAAI Plugin Name: VMW_VAAIP_NETAPP
    ATS Status: supported
    Clone Status: supported
    Zero Status: supported
    Delete Status: supported
```

VAAI Tuning and Monitoring

This section describes some of the options available for fine-tuning and monitoring the VAAI primitives. This will help administrators determine whether VAAI is working correctly in their respective environments.

We first will discuss XCOPY. The default XCOPY size is 4MB. With a 32MB I/O, the expectation would be to see this counter in esxtop incrementing in batches of eight, the number of work items that will be created to deliver a 32MB I/O. The default XCOPY size can be incremented to a maximum value of 16MB if required, but that should be done only on the advice of your storage array vendor. The following command shows how to query the current transfer size and how to change it:

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

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

NOTE: Please use extreme caution when changing this advanced parameter. This parameter is a global parameter, so it will impact ALL storage arrays attached to the host. While a storage array vendor might suggest making a change to this parameter for improved performance on their particular array, it may lead to issues on other arrays which do not work well with the new setting, including degraded performance.

The data-out buffer of the **WRITE_SAME** command contains all zeroes. A single zero operation has a default zeroing size of 1MB. The maximum number of outstanding **WRITE_SAME** commands is 32.We currently do not support changing the WRITE_SAME size of 1MB.

One can observe the clone operations (XCOPY), ATS operations, and zero operations (WRITE_SAME) in an esxtop output. The following is an esxtop output showing the various block primitives:

FIgure 4. esxtop Output – Block Primitives

To see this output in esxtop, first select **u** for device view, and then **f** to change which fields are displayed. Options **o** and **p** display VAAI statistics. Each primitive also has a column for failure statistics, that is, CLONE_F, ATSF and ZERO_F. These should be monitored for any failed operations.

One side effect is that VAAI offloads can lead to distorted kernel average latencies (KAVGs). When VAAI commands are issued via the VAAI filter, there actually are two commands sent. These are top-layer commands that are issued but never sent to the actual device (they stay within the VMkernel). These commands are intercepted by the VAAI filter and the VAAI plug-in and are replaced by the vendor-specific commands, which are issued to the device.

For this reason, esxtop shows device statistics for only the top-level commands. As a result, the values for latencies seem unusual when compared to results obtained when VAAI is not enabled.

Figure 5. esxtop Output – Latency Values

In this—and only this—instance, the latencies observed in esxtop should not be interpreted as a performance issue, unless there are other symptoms present.

Checking VAAI Support

To check VAAI status, a number of **esxcli** commands are available. For example, to see which primitives are supported for a particular device, the following command can be used:

```
# esxcli storage core  device vaai status get –d  naa.xxx
naa.xxx
    VAAI Plugin Name: VMW_VAAIP_NETAPP
    ATS Status: supported
    Clone Status: supported
    Zero Status: supported
    Delete Status: supported
```

In the preceding output, the **ATS Status, Clone Status** (XCOPY), **Zero Status** (WRITE_SAME) and **Delete Status** (UNMAP) are supported. The vSphere UI also can be used to verify VAAI support. The overall VAAI status is displayed as **Supported, Not supported** or **Unknown** under **Hardware Acceleration** in the UI, as is shown in the following:
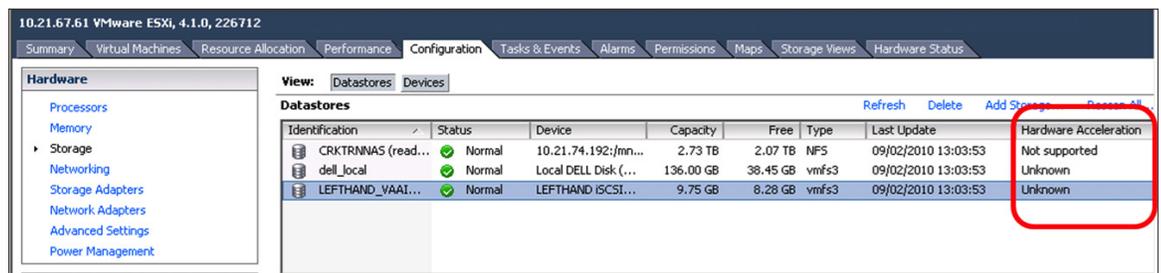

Figure 6. VAAI Status

Hardware acceleration status is determined by the support state of various primitives. If ATS is supported, the UI displays **Hardware Acceleration** as **Supported**. If, however, ATS, XCOPY and zero all are unsupported, **Hardware Acceleration** displays as **Unsupported**. All other support states set the hardware acceleration status to **Unknown**, which typically is a state that is displayed until the first clone and zero operation is initiated. Many users initiate an eagerzeroedthick clone of a file to test the VAAI status.

Enabling and Disabling VAAI

To disable some of the VAAI primitives—for troubleshooting purposes, for example—see the following CLI commands, which detail how to enable and disable VAAI primitives.

ATS

To check the status of the ATS primitive and to turn it on and off at the command line, the following commands can be used:

```
# esxcli system settings advanced list --option /VMFS3/HardwareAcceleratedLocking
Value of HardwareAcceleratedLocking is 1

# esxcli system settings advanced set --int-value 0 --option /VMFS3/
HardwareAcceleratedLocking

# esxcli system settings advanced list --option /VMFS3/HardwareAcceleratedLocking
Value of HardwareAcceleratedLocking is 0
```

XCOPY

To turn the Extended Copy primitive for cloning on and off, use the previous command with the following advanced setting:

```
/DataMover/HardwareAcceleratedMove
```

WRITE_SAME

To turn the Write Same primitive for zeroing blocks on and off, use the following advanced setting:

```
/DataMover/HardwareAcceleratedInit
```

UNMAP

To turn the UNMAP primitive for space reclamation on and off, use the following advanced setting:

/VMFS3/EnableBlockDelete

In vSphere 4.1, it was possible to define how often an ESXi host verified whether hardware acceleration was supported on the storage array.

```
# esxcfg-advcfg -g /DataMover/HardwareAcceleratedMoveFrequency
Value of HardwareAcceleratedMoveFrequency is 16384
```

This means that if at initial deployment, an array does not support the offload primitives, but at a later date the firmware on the arrays is upgraded and the offload primitives become supported, nothing must be done regarding ESXi—it automatically will start to use the offload primitives. The value relates to the number of I/O requests that occur before another offload is attempted. With an I/O size of 32MB, 512GB of I/O must flow before a VAAI primitive is retried.

The same is true for an offload failure. If the array returns a failure status for an offload operation, the software Data Mover is used to continue the operation. After another 16,384 I/O requests using the software Data Mover, the VAAI offload is once again attempted.

HardwareAcceleratedMoveFrequency exists only in vSphere 4.1. In vSphere 5.0 and later, the parameter was replaced with a periodic VAAI state evaluation that runs every 5 minutes.

Reverting to the Software Data Mover

Before VAAI, migrations of virtual machines, and their associated disks, between datastores was done by a component called the software Data Mover, a loadable VMkernel module accessible via special library calls. A data move request, in the form of a data structure, is submitted to the Data Mover for asynchronous completion. The Data Mover takes this data structure and divides it into smaller I/O requests that it sends to the appropriate back end.

The Data Mover relies on a constantly filled queue of outstanding I/O requests to achieve maximum throughput. Incoming Data Mover requests are broken down into smaller chunks, and asynchronous I/Os are simultaneously issued for each chunk until the configured Data Mover queue depth is filled. Then the asynchronous Data Mover request returns to the caller who would normally proceed with what it was doing or wait for I/O completion. Meanwhile, as and when an asynchronous request completes, the next logical request for the overall Data Mover task is issued, whether it is for writing the data that was just read or to handle the next chunk.

In the following example, a clone of a 64GB VMDK file is initiated. The Data Mover is asked to move data in terms of 32MB transfers. The 32MB is then transferred in "PARALLEL" as a single delivery but is divided up into a much smaller I/O size of 64KB, using 32 work items that are all queued in parallel (because they execute asynchronously). To transfer this 32MB, a total of 512 I/Os of size 64KB are issued by the Data Mover. By comparison, a similar 32MB transfer via VAAI uses eight work items, because XCOPY uses 4MB transfer sizes.

VMkernel can choose from the following three Data Mover types:

• **fsdm** – This is the legacy Data Mover, the most basic version. It is the slowest because the data moves all the way up the stack and then down again.
• **fs3dm** (software) – This is the software Data Mover, which was introduced with vSphere 4.0 and contained some substantial optimizations whereby data does not travel through all stacks.
• **fs3dm** (hardware) – This is the hardware offload Data Mover, introduced with vSphere 4.1. It still is the fs3dm, but VAAI hardware offload is leveraged with this version. fs3dm is used in software mode when hardware offload (VAAI) capabilities are not available.

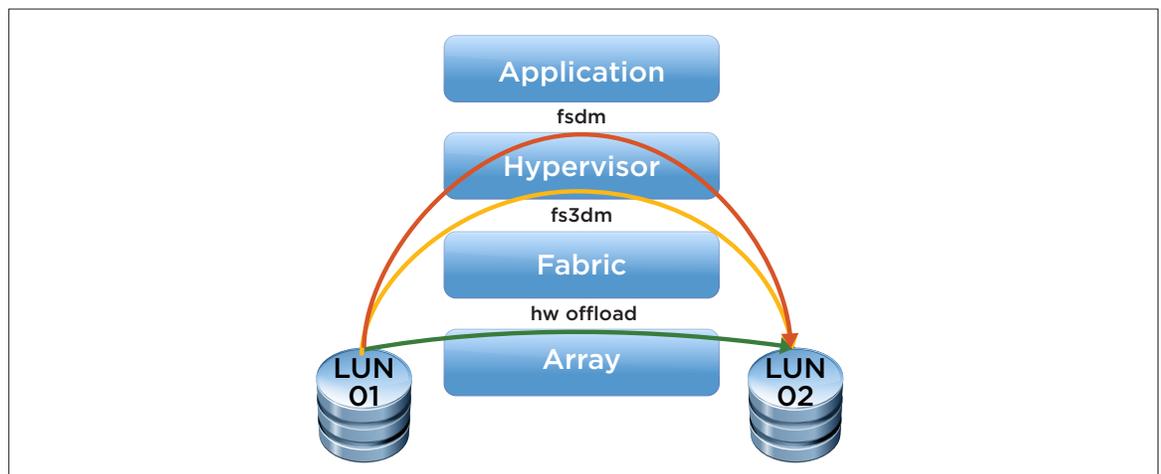The following diagram shows the various paths taken by the various Data Movers:

Figure 7. Data Mover Types

The decision to transfer using the software Data Mover or using hardware offloading to the array with VAAI is taken upfront by looking at storage array hardware acceleration state. If we decide to transfer using VAAI and then FAIL—a scenario called degraded mode—the VMkernel will try to accomplish the transfer using the software Data Mover. It should be noted that the operation is not restarted; rather it picks up from where the previous transfer left off as we do not want to abandon what could possibly be very many gigabytes of copied data because of a single transient transfer error.

When reverting to the software Data Mover, we cannot operate in terms of the XCOPY transfer size of 4MB, due to memory buffer requirement, so the transfer size is reduced to 64KB.

Differences in VAAI Between vSphere 4.x and 5.x

There are a number of differences between the first phase/release of VAAI in vSphere 4.1 and the second phase of VAAI, which was released with vSphere 5.0. The following is a list of those differences:

• VAAI in 5.0 now uses standard T10 primitives rather than per-array manufacturer commands.

• There is now full ATS support for all metadata operations with VMFS5.

• vSphere 5.0 introduced support for NAS primitives, including VCAI.

• vSphere 5.0 introduced support for Thin Provisioning primitives, including UNMAP.

• VMware HCL now requires performance of primitives before an array receives VAAI certification. This is important because certain storage arrays that appear in the vSphere 4.1 HCL might not appear in the vSphere 5.0 HCL if the performance of the offload primitives does not meet requirements.

VAAI Caveats

In cases involving the following known caveats related to VAAI, hardware offloads to the array will not be leveraged; the Data Mover will be used instead:

VMFS Block Sizes

If the source and destination VMFS volumes have different block sizes, data movement will revert to the generic FSDM layer, which will move onlysoftware data.

Raw Device Mappings

If the source file type is raw device mapping (RDM) and the destination file type is non-RDM (regular file), migration occurs via the Data Mover.

Sparse/Hosted Format VMDK

If either the source or destination VMDK involved in the operation is of sparse or hosted format, the Data Mover is used.

Misalignment

The software Data Mover is used if the logical address and/or transfer length in the requested operation is not aligned to the minimum alignment required by the storage device.

If the VMFS datastores are created using vSphere Client or vCenter, it is likely that all requests will be aligned. If a LUN is incorrectly partitioned using low-level tools such as *fdisk* or *partedUtil*, the operation might revert to using the software Data Mover.

Because customers typically use vSphere Client and vCenter exclusively for a created datastore, it is unlikely that this issue would be encountered. If a customer's environment was migrated from ESX 2.x/VMFS2 in the distant past, however, the volumes might be misaligned.

This can be an issue on some VAAI arrays that have a high alignment requirement.

VMFS Extents

VMware supports VAAI primitives on VMFS with multiple LUNs/extents, if they all are on the same array and the array supports offloading. There are some limitations on how ATS can be used on multiextent VMFS volumes, but that impact is relatively low.

On VAAI Hardware	New VMFS5	Upgraded VMFS5	VMFS3
Single-extent datastore	ATS only[1]	ATS, but reverts to SCSI-2 reservations	ATS, but reverts to SCSI-2 reservations
Multiextent datastore	Allows spanning only on ATS hardware[2]	ATS, except when locks on non-head	ATS, except when locks on non-head

Table 1. ATS Limitations on Multiextent VMFS Volumes

Different Destination Storage Array

VMware does not support VAAI primitives on VMFS with multiple LUNs/extents if they all are on different arrays, even if all arrays support offloading. Hardware cloning between arrays—even if it is within the same VMFS volume—will not work, so that would revert to software data movement.

---

1. If a new VMFS5 is created on a non-ATS storage device, SCSI-2 reservations will be used.

2. When creating a multiextent datastore in which ATS is used, the VMware vCenter Server™ will filter out non-ATS devices, so only devices that support the ATS primitive can be used.

Acknowledgments

Thank you to James Senicka, Paudie O'Riordan, Duncan Epping, Tejasvi Aswathanarayana and Ilia Sokolinski for reviewing the contents of this paper.

About the Author

Cormac Hogan is a senior technical marketing architect with the Cloud Infrastructure Product Marketing group at VMware. He is responsible for storage in general, with a focus on core VMware vSphere storage technologies and virtual storage, including the VMware vSphere Storage Appliance. He has been in technical marketing since 2011.

• Follow Cormac's blogs at http://blogs.vmware.com/vsphere/storage and http://cormachogan.com.

• Follow Cormac on Twitter: @VMwareStorage.

vmware®