



Pure Storage and VMware Storage APIs for Array Integration—VAAI

Cody Hosterman, Solutions Architect, vExpert 2013-2014

Version 1, July 2014

Table of Contents

- 1 [Executive Summary](#)
- 2 [Pure Storage Introduction](#)
- 3 [Introduction to VAAI](#)
- 4 [VAAI Best Practices Checklist](#)
- 6 [Enabling/Disabling VAAI](#)
- 7 [ATS or Hardware assisted locking](#)
- 8 [Full Copy or XCOPY](#)
- 9 [Block Zero or WRITE SAME](#)
- 10 [Dead Space Reclamation or UNMAP](#)
- 11 [Monitoring VAAI with ESXTOP](#)

Executive Summary

This document describes the purpose and performance characterizations of the VMware Storage APIs for Array Integration (VAAI) with the Pure Storage FlashArray. The Pure Storage FlashArray includes general support for VMware ESXi as well as the most important VAAI primitives that enable administrators to enhance and simplify the operation and management of VMware vSphere virtualized environments. Throughout this paper, specific best practices on using VAAI with Pure Storage will be discussed.

This document is intended for use by pre-sales consulting engineers, sales engineers and customers who want to deploy the Pure Storage FlashArray in VMware vSphere-based virtualized datacenters.

Pure Storage Introduction

Pure Storage is the leading all-flash enterprise array vendor, committed to enabling companies of all sizes to transform their businesses with flash.

Built on 100% consumer-grade MLC flash, Pure Storage FlashArray delivers all-flash enterprise storage that is 10X faster, more space and power efficient, more reliable, and infinitely simpler, and yet typically costs less than traditional performance disk arrays.

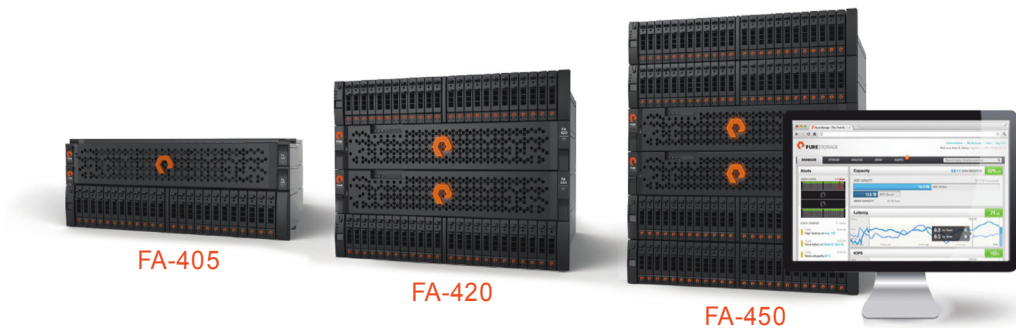


Figure 1. FlashArray 400 Series

The Pure Storage FlashArray FA-400 Series is ideal for:

Accelerating Databases and Applications Speed transactions by 10x with consistent low latency, enable online data analytics across wide datasets, and mix production, analytics, dev/test, and backup workloads without fear.

Virtualizing and Consolidating Workloads Easily accommodate the most IO-hungry Tier 1 workloads, increase consolidation rates (thereby reducing servers), simplify VI administration, and accelerate common administrative tasks.

Delivering the Ultimate Virtual Desktop Experience Support demanding users with better performance than physical desktops, scale without disruption from pilot to >1000's of users, and experience all-flash performance for under \$100/desktop.

Protecting and Recovering Vital Data Assets Provide an always-on protection for business-critical data, maintain performance even under failure conditions, and recover instantly with FlashRecover.

Pure Storage FlashArray sets the benchmark for all-flash enterprise storage arrays. It delivers:

Consistent Performance FlashArray delivers consistent <1ms average latency. Performance is optimized for the real-world applications workloads that are dominated by I/O sizes of 32K or larger vs. 4K/8K hero performance benchmarks. Full performance is maintained even under failures/updates.

Less Cost than Disk Inline de-duplication and compression deliver 5 – 10x space savings across a broad set of I/O workloads including Databases, Virtual Machines and Virtual Desktop Infrastructure.

Mission-Critical Resiliency FlashArray delivers >99.999% proven availability, as measured across the Pure Storage installed base and does so with non-disruptive everything without performance impact.

Disaster Recovery Built-In FlashArray offers native, fully-integrated, data reduction-optimized backup and disaster recovery at no additional cost. Setup disaster recovery with policy-based automation within minutes. And, recover instantly from local, space-efficient snapshots or remote replicas.

Simplicity Built-In FlashArray offers game-changing management simplicity that makes storage installation, configuration, provisioning and migration a snap. No more managing performance, RAID, tiers or caching. Achieve optimal application performance without any tuning at any layer. Manage the FlashArray the way you like it: Web-based GUI, CLI, VMware vCenter, Rest API, or OpenStack.

Pure Storage FlashArray FA-400 Series includes FA-405, FA-420, and FA-450. A FlashArray is available for any application, and any budget!







	FA-405	FA-420	FA-450
FRONT VIEW (dual controllers)			
REAR VIEW (dual controllers)			
CAPACITY	<ul style="list-style-type: none"> Up to 40+ TBs effective capacity 2.75-11 TBs raw capacity 	<ul style="list-style-type: none"> Up to 125+ TBs effective capacity 11-35 TBs raw capacity 	<ul style="list-style-type: none"> Up to 250+ TBs effective capacity 34-70 TBs raw capacity
Effective capacity assumes HA, RAID, and metadata overhead, GB-to-GiB conversion, and includes benefit of data reduction with always-on inline deduplication, compression & pattern removal. Average data reduction is calculated at 6-to-1. Some customers see data reduction in excess of 20-to-1. Effective capacity has no upper limit and will vary depending on workload.			
PERFORMANCE	<ul style="list-style-type: none"> Up to 100,000 32K IOPS @ <1ms average latency Up to 3 GB/s bandwidth 	<ul style="list-style-type: none"> Up to 150,000 32K IOPS @ <1ms average latency Up to 5 GB/s bandwidth 	<ul style="list-style-type: none"> Up to 200,000 32K IOPS @ <1ms average latency Up to 7 GB/s bandwidth
Why does Pure Storage quote 32K, not 4K IOPS? The industry commonly markets 4K IOPS benchmark to make numbers look high, but real-world environments are dominated by IO sizes of 32K or larger. Pure Storage has optimized the FlashArray for the real-world. FlashArray adapts automatically to 512B-32KB IO for superior performance, scalability, and data reduction.			
HOST CONNECTIVITY	<ul style="list-style-type: none"> 8 Gb/s Fibre Channel 10 Gb/s Ethernet iSCSI Replication ports 	<ul style="list-style-type: none"> 8 Gb/s Fibre Channel 10 Gb/s Ethernet iSCSI Expansion slot (FC or iSCSI) Replication ports 	<ul style="list-style-type: none"> 16 Gb/s Fibre Channel 10 Gb/s Ethernet iSCSI Expansion slot (FC or iSCSI) Replication ports

Figure 2. Pure Storage FlashArray 400 Series Specifications

Start Small and Grow Online

FlashArray scales from smaller workloads to data center-wide consolidation. And because upgrading performance and capacity on the FlashArray is always non-disruptive, you can start small and grow without impacting mission-critical applications. Coupled with Forever Flash, a new business model for storage acquisition and lifecycles, FlashArray provides a simple and economical approach to evolutionary storage that extends the useful life of an array and does away with the incumbent storage vendor practices of forklift upgrades and maintenance extortion.

Love Your Storage Guarantee

FlashArray is backed by the industry's broadest storage guarantee – Love Your Storage Guarantee. If for any reason, you are not delighted within the first 30 days of your FlashArray deployment experience, you can return it for a full refund.

You can learn more about Pure Storage at www.purestorage.com.

Introduction to VAAI

The VMware Storage APIs for Array Integration (VAAI) is a feature set introduced vSphere 4.1 that accelerates common tasks by offloading certain storage-related operations to compatible arrays. With the storage hardware assistance, an ESXi host can perform these operations faster and more efficiently while consuming far less CPU, memory, and storage fabric bandwidth. All VAAI primitives are enabled by default and will automatically be invoked if ESXi detects that there is support from the underlying storage. Pure Storage FlashArray supports VAAI in ESXi 5.0 and later. The following five vStorage APIs are available for block-storage hardware vendors to implement and support:

- **Hardware Assisted Locking**—commonly referred to as Atomic Test & Set (ATS), this uses the SCSI command COMPARE and WRITE (0x89), which is invoked to replace legacy SCSI reservations during the creation, alteration and deletion of files and metadata on a VMFS volume.
- **Full Copy**—leverages the SCSI command XCOPY (0x83), which is used to copy or move virtual disks.
- **Block Zero**—leverages the SCSI command WRITE SAME (0x93) which is used to zero-out disk regions during virtual disk block allocation operations.
- **Dead Space Reclamation**—leverages the SCSI command UNMAP (0x42) to reclaim previously used but now deleted space on a block SCSI device.
- **Thin Provisioning Stun and Resume**¹—allows for underlying storage to inform ESXi that capacity has been entirely consumed which causes ESXi to immediately “pause” virtual machines until additional capacity can be provisioned/installed.

Pure Storage FlashArray supports ATS, XCOPY, WRITE SAME and UNMAP in Purity release 3.0.0 onwards on ESXi 5.x. Thin provisioning Stun & Resume support is currently under development.

¹ Thin Provisioning Stun & Resume is not currently supported by the Pure Storage Flash Array.

VAAI Best Practices Checklist

The following section is intended as a quick-start guide for using VAAI functionality on Pure Storage. Refer to the relevant sections in the rest of the document for more information.

1. Ensure proper multipathing configuration is complete. This means more than one HBA and connections to at least four FlashArray ports. All Pure devices should be controlled by the VMware Native Multipathing Plugin (NMP) Round Robin Path Selection Policy (PSP). Furthermore, each device should be configured to use an I/O Operation Limit of 1.
2. Ensure all primitives all enabled.
3. For XCOPY, set the maximum transfer size to 16 MB.
4. For UNMAP in ESXi 5.5, use a large block count (~60,000).
5. WRITE SAME and ATS have no specific recommendations.

Enabling/Disabling VAAI

In ESXi 5.x hosts, to determine if VAAI is enabled using the service console or the vCLI, run these command to check if `Int Value` is set to 1 (enabled):

```
esxcli system settings advanced list -o /DataMover/HardwareAcceleratedMove
esxcli system settings advanced list -o /DataMover/HardwareAcceleratedInit
esxcli system settings advanced list -o /VMFS3/HardwareAcceleratedLocking
```

You will see an output similar to:

```
Path: /VMFS3/HardwareAcceleratedLocking
Type: integer
Int Value: 1          ← Value is 1 if enabled
Default Int Value: 1
Min Value: 0
Max Value: 1
String Value:
Default String Value:
Valid Characters:
Description: Enable hardware accelerated VMFS locking (requires compliant hardware)
```

Hardware acceleration is enabled by default and requires no work on the array or ESXi to use out of the box. In the case it was somehow disabled, follow these steps to re-enable the primitives:

To enable atomic test and set (ATS) AKA hardware accelerated locking:

```
esxcli system settings advanced set -i 1 -o /VMFS3/HardwareAcceleratedLocking
```

To enable Hardware accelerated initialization AKA WRITE SAME:

```
esxcli system settings advanced set --int-value 1 --option /DataMover/HardwareAcceleratedInit
```

To enable Hardware accelerated move AKA XCOPY (full copy):

```
esxcli system settings advanced set --int-value 1 --option /DataMover/HardwareAcceleratedMove
```

The figure below describes the above steps pictorially using the vSphere Web Client. Go to an ESXi host and then Settings, then Advanced System Settings and search for “Hardware”

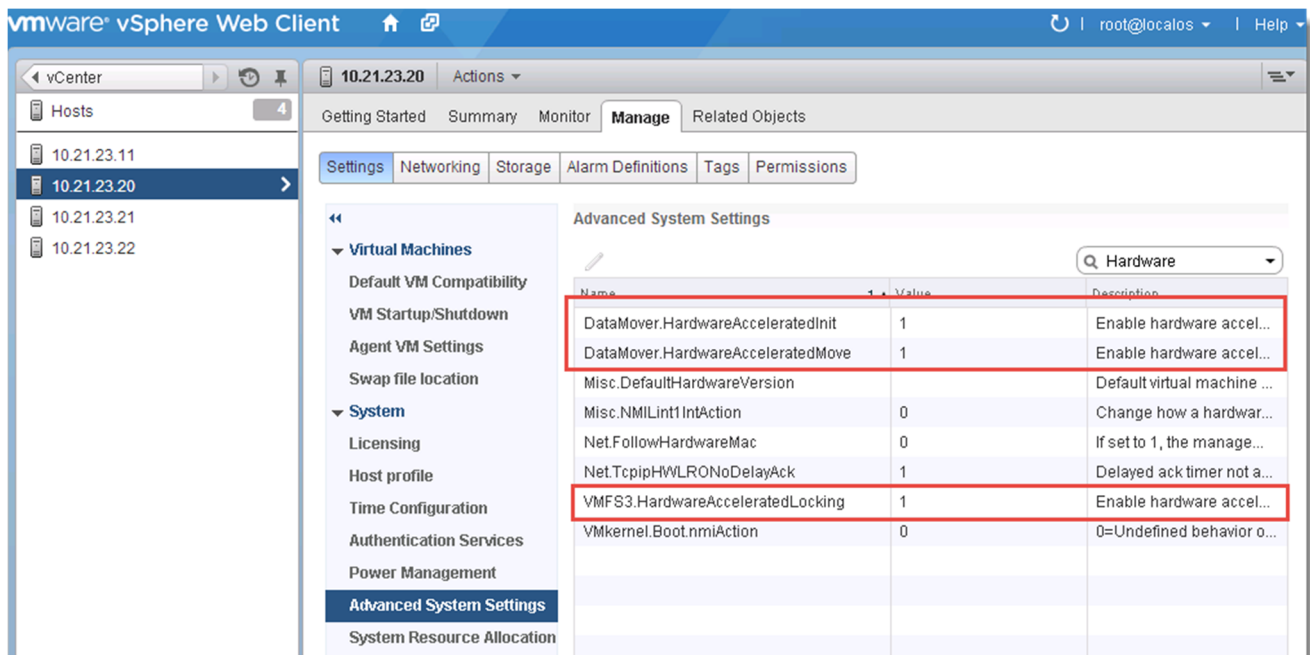


Figure 3. VAAI advanced options in the vSphere Web Client

ATS or Hardware assisted locking

Prior to the introduction of VAAI ATS, ESXi used device-level locking via acquiring full SCSI reservations to get and control access to the metadata associated with a VMFS volume. In a cluster with multiple nodes, all metadata operations were serialized and hosts had to wait until whichever host that was currently holding the lock released it. This behavior not only caused metadata lock queues which slowed down operations like virtual machine provisioning but also delayed any standard I/O to a volume from ESXi hosts not currently holding the lock until the lock was released.

With VAAI ATS, the locking granularity is reduced to a much smaller level of control by only locking specific metadata segments, instead of an entire volume. This behavior makes the metadata change process not only very efficient but importantly provides a mechanism for parallel metadata access while still maintaining data integrity. ATS allows for ESXi hosts to no longer have to queue metadata change requests which consequently speeds up operations that previously had to wait for a lock. Therefore, situations with large amounts of simultaneous virtual machine provisioning operations will see the most benefit. The standard use cases benefiting the most from ATS include:

- High virtual machine to VMFS density
- Extremely dynamic environments—numerous provisioning and de-provisioning of VMs.
- VM operations such as boot storms, or virtual disk growth

Performance Examples

Unlike some of the other VAAI primitives, the benefits of hardware assisted locking are not always readily apparent in day to day operations. That being said there are some situations where the benefit arising from the enablement of hardware assisted locking can be somewhat profound. For example, see the following case.

Hardware assisted locking provides the most assistance in situations where traditionally there would be an exceptional amount of SCSI reservations over a pro-longed period of time. The most standard example of this would be a mass power-on of a large number of virtual machines, commonly known as a boot storm. During a boot storm the host or hosts booting up the virtual machines require at least an equivalent number of locks to power on the virtual machines. This frequent and sustained locking can easily affect other workloads that share the target datastore(s) of the virtual machines in the boot storm. These volume-level locks cause other workloads to have reduced and unpredictable performance for the duration of the boot storm. Refer to the following charts that show throughput and IOPS of a workload running during a boot storm.

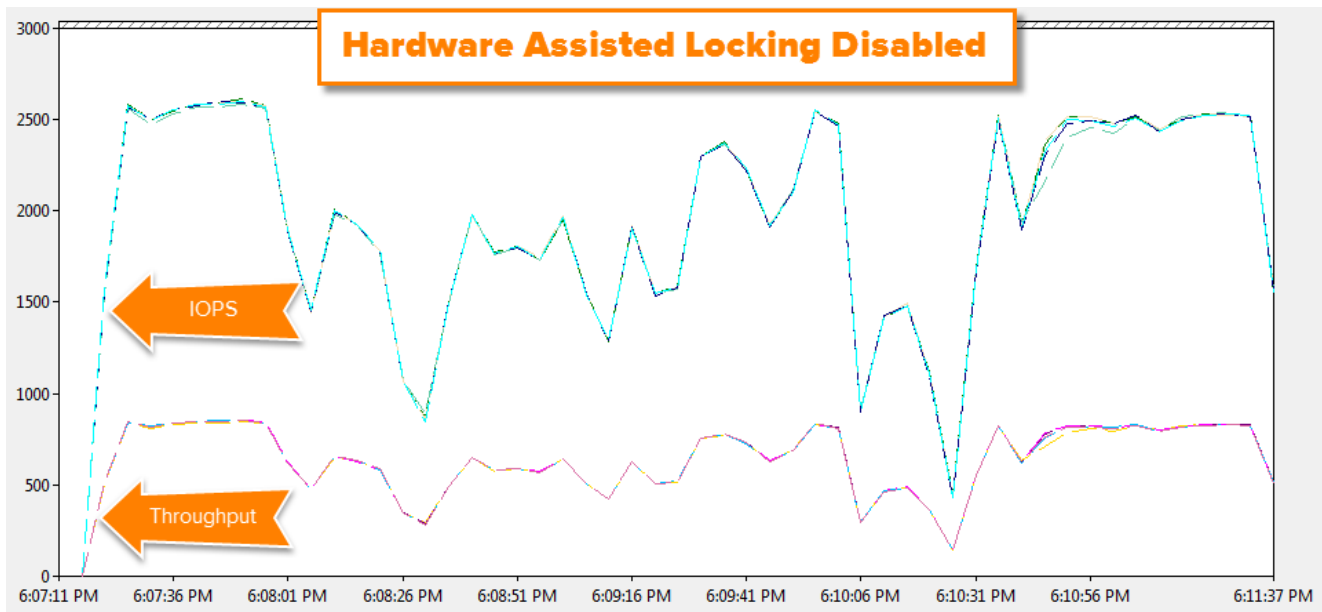


Figure 4. Performance test with hardware assisted locking disabled

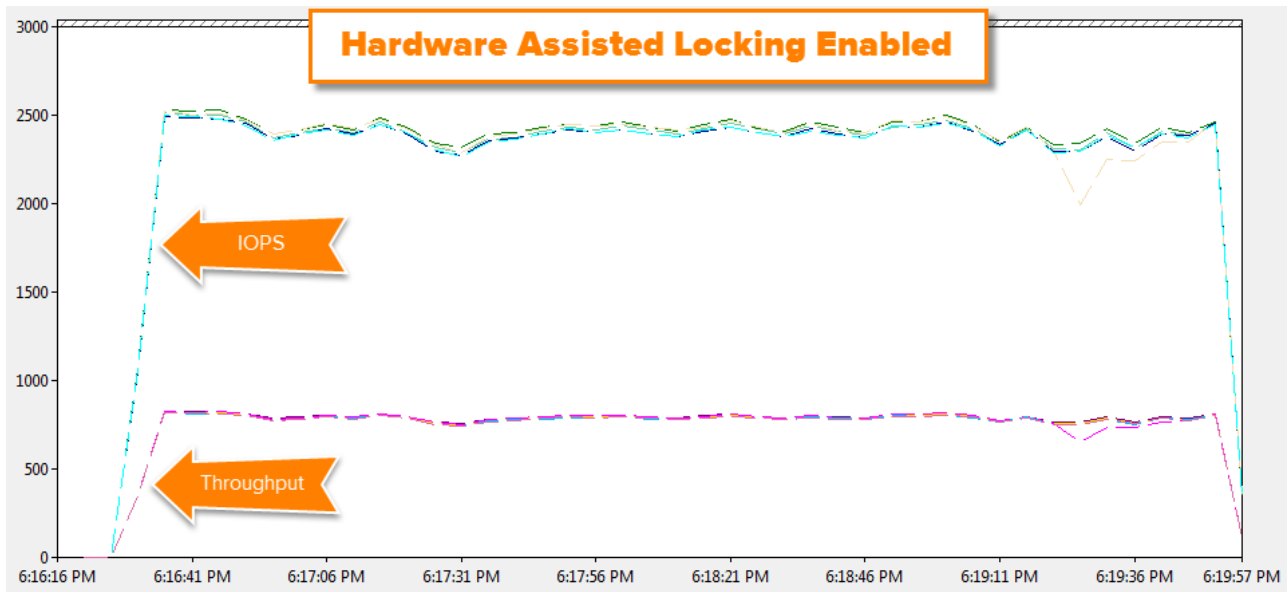


Figure 5. Performance test with hardware assisted locking enabled

In this scenario, a virtual machine ran a workload to five virtual disks that all resided on the same datastore as a 150 virtual machines that were all booted up simultaneously. By referring to the previous charts, it can be easily noted that with hardware assisted locking disabled the workload is deeply disrupted resulting in inconsistent and inferior performance during the boot storm. Both the IOPS and throughput² vary wildly throughout the test. When hardware assisted locking is enabled the disruption is almost entirely gone and the workload proceeds unfettered.

Full Copy or Hardware Accelerated Copy

Prior to Full Copy (XCOPY) API support, when data needed to be copied from one location to another such as with Storage vMotion or a virtual machine cloning operation, ESXi would issue many SCSI read/write commands between the source and target storage location (the same or different device). This resulted in a very intense and often lengthy additional workload to the target storage. This I/O consequently stole available bandwidth from more “important” I/O such as the I/O issued from virtualized applications. Therefore, copy or movement operations often had to be scheduled to occur only during non-peak hours in order to limit interference with normal production storage performance. This restriction effectively decreased the stated dynamic abilities and benefits offered by a virtualized infrastructure.

The introduction of XCOPY support for virtual machine data movement allows for this workload to be offloaded from the virtualization stack to almost entirely onto the storage array. The ESXi kernel is no longer directly in the data copy path and the storage array instead does all the work. XCOPY functions by having the ESXi host identify a region that needs to be copied. ESXi describes this space in a series of XCOPY SCSI commands and sends them to the array. The array then translates these block descriptors and copies the data at the described

² The scale for throughput is in MB/s but is reduced in scale by a factor of ten to allow it to fit in a readable fashion on the chart with the IOPS values. So a throughput number on the chart of 1,000 is actually a throughput of 100 MB/s.

source locations to the described target location entirely within the array. This architecture therefore does not require the moved data to be sent back and forth between the host and array—the SAN fabric does not play a role in traversing the data. The host only tells the array where the data that needs to be moved resides and where to move it to—it does not need to tell the array what the data actually is and consequently vastly reduces the time to move data. XCOPY benefits are leveraged during the following operations³:

- Virtual machine cloning
- Storage vMotion
- Deploying virtual machines from template

During these offloaded operations, the throughput required on the data path is greatly reduced as well as the load on the ESXi hardware resources (HBAs, CPUs etc.) initiating the request. This frees up resources for more important virtual machine operations by letting the ESXi resources do what they do best: run VMs, and lets the storage do what it does best: manage the storage.

On the Pure Storage FlashArray, XCOPY sessions are exceptionally quick and efficient. Due to the Purity FlashReduce technology (features like deduplication, pattern removal and compression) similar data is not stored on the FlashArray more than once. Therefore, during a host-initiated copy operation such as XCOPY, the FlashArray does not need to copy the data—this would be wasteful. Instead, Purity simply accepts and acknowledges the XCOPY requests and just creates new (or in the case of Storage vMotion, redirects existing) metadata pointers. By not actually having to copy/move data the offload process duration is greatly reduced. In effect, the XCOPY process is a 100% inline deduplicated operation.

A standard copy process for a virtual machine containing, for example, 50 GB of data can take many minutes or more depending on the workload on the SAN. When XCOPY is enabled and properly configured this time drops to a matter of a few seconds—usually around 10 for a virtual machine of that size.

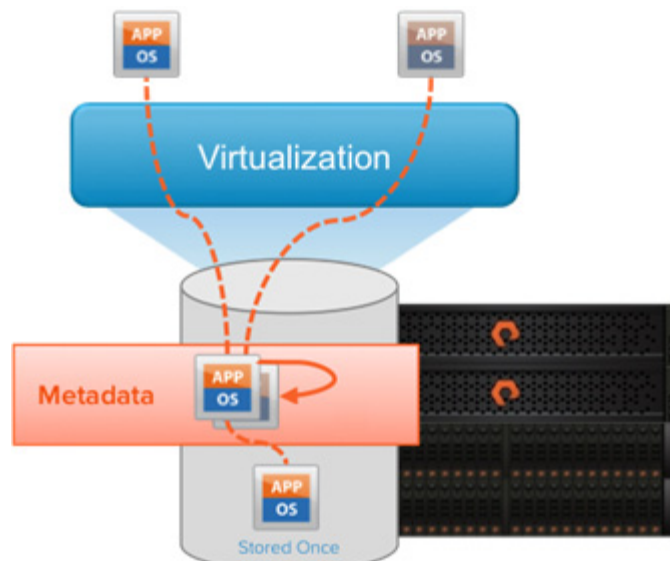


Figure 6. Pure Storage XCOPY implementation

³ Note that there are VMware-enforced caveats in certain situations that would prevent XCOPY behavior and revert to legacy software copy. Refer to VMware documentation for this information at www.vmware.com.

XCOPY on the Pure Storage FlashArray works directly out of the box without any pre-configuration required. Nevertheless, there is one simple configuration change on the ESXi hosts that can increase the speed of XCOPY operations. ESXi offers an advanced setting called the MaxHWTransferSize that controls the maximum amount of data space that a single XCOPY SCSI command can describe. The default value for this setting is 4 MB. This means that any given XCOPY SCSI command sent from that ESXi host cannot exceed 4 MB of described data.

The FlashArray, as previously noted, does not actually copy the data described in a XCOPY transaction—it just moves or copies metadata pointers. Therefore, for the most part, the bottleneck of any given virtual machine operation that leverages XCOPY is not the act of moving the data (since no data is moved), but how quickly an ESXi host can send XCOPY SCSI commands to the array. Therefore copy duration depends on the number of commands sent (dictated by both the size of the virtual machine and the maximum transfer size) and correct multi-pathing configuration.

Accordingly, if more data can be described in a given XCOPY command, less commands overall need to be sent and it subsequently takes less time for the total operation to complete. For this reason Pure Storage recommends setting the transfer size to the maximum value of 16 MB⁴.

The following commands provide for retrieval of the current value, and for setting a new one.

```
esxcfg-advcfg -g /DataMover/MaxHWTransferSize
```

```
esxcfg-advcfg -s 16384 /DataMover/MaxHWTransferSize
```

As mentioned earlier, general multipathing configuration best practices play a role in the speed of these operations. Changes like setting the Native Multipathing Plugin (NMP) Path Selection Plugin (PSP) for Pure devices to Round Robin and configuring the Round Robin IO Operations Limit to 1 can also provide an improvement in copy durations (offloaded or otherwise). Refer to the VMware and Pure Storage Best Practices Guide on www.purestorage.com for more information.

Performance Examples

The following sections will outline a few examples of XCOPY usage to describe expected behavior and performance benefits with the Pure Storage FlashArray. All tests will use the same virtual machine:

- Windows Server 2012 R2 64-bit
- 4 vCPUs, 8 GB Memory
- One zeroedthick 100 GB virtual disk containing 50 GB of data (in some tests the virtual disk type is different and is noted where necessary)

If performance is far off from what is expected it is possible that the situation is not supported by VMware for XCOPY offloading and legacy software-based copy is instead being used. The following VMware restrictions apply that could cause XCOPY to not be used:

⁴ Note that this is a host-wide setting and will affect all arrays attached to the host. If a third party array is present and does not support this change leave the value at the default or isolate that array to separate hosts.

- The source and destination VMFS volumes have different block sizes
- The source file type is RDM and the destination file type is a virtual disk
- The source virtual disk type is eagerzeroedthick and the destination virtual disk type is thin
- The source or destination virtual disk is any kind of sparse or hosted format
- Target virtual machine has snapshots
- The VMFS datastore has multiple LUNs/extents spread across different arrays
- Storage vMotion or cloning between arrays

Deploy from Template

In the first test, the virtual machine was configured as a template and resided on a VMFS on a Pure Storage volume (naa.624a9370753d69fe46db318d00011015). A single virtual machine was deployed from this template onto a different datastore (naa.624a9370753d69fe46db318d00011014) on the same FlashArray. The test was run twice, once with XCOPY disabled and one with it enabled. With XCOPY enabled, the clone operation was far faster and greatly reduced both the IOPS and throughput from the host for the duration of the operation.

Clone virtual machine

Status: ✔ Completed
 Initiator: root
 Target: Win2012R2
 Result: XCOPY_VM1c
 Server: vcenter-csg

XCOPY Disabled: 130 seconds

Related events:

Wednesday, June 11, 2014 9:17:46 AM	Template Win2012R2 deployed on host 10.21.23.71
Wednesday, June 11, 2014 9:17:46 AM	Reconfigured XCOPY_VM1c on 10.21.23.71 in Mountain View
Wednesday, June 11, 2014 9:15:36 AM	Deploying XCOPY_VM1c on host 10.21.23.71 in Mountain View from template Win2012R2
Wednesday, June 11, 2014 9:15:36 AM	Task: Clone virtual machine

Figure 8. Deploy from template operation with XCOPY disabled

Clone virtual machine

Status: ✔ Completed
 Initiator: root
 Target: Win2012R2
 Result: XCOPY_VM1d
 Server: vcenter-csg

XCOPY Enabled: 7 seconds

Related events:

Wednesday, June 11, 2014 9:36:19 AM	Template Win2012R2 deployed on host 10.21.23.71
Wednesday, June 11, 2014 9:36:19 AM	Reconfigured XCOPY_VM1d on 10.21.23.71 in Mountain View
Wednesday, June 11, 2014 9:36:12 AM	Deploying XCOPY_VM1d on host 10.21.23.71 in Mountain View from template Win2012R2
Wednesday, June 11, 2014 9:36:11 AM	Task: Clone virtual machine

Figure 7. Deploy from template operation with XCOPY enabled

The above images show the vSphere Web Client log of the “deploy from template” operation times. The deployment operation time was reduced from over two minutes down to seven seconds. The following images show the perfmon graphs gathered from esxtop comparing total IOPS and total throughput when XCOPYY is enabled and disabled. Note that the scales are identical for both the XCOPYY-enabled and XCOPYY-disabled charts.

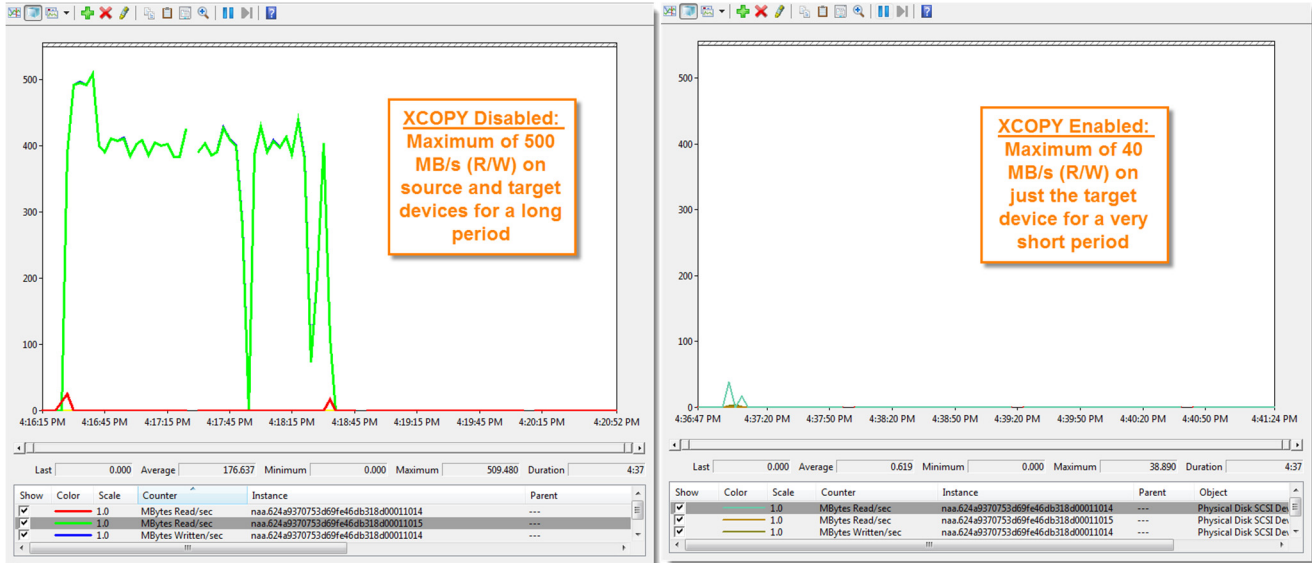


Figure 10. Deploy from template throughput improvement with XCOPY

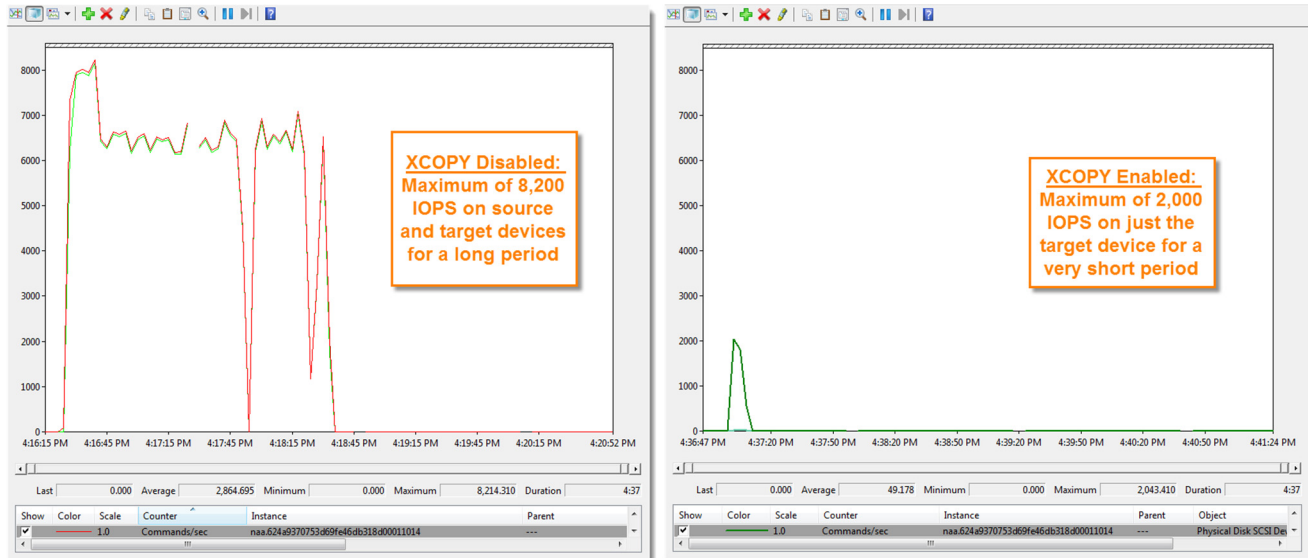


Figure 9. Deploy from template IOPS improvement with XCOPY

Simultaneous Deploy From Template Operations

This improvement does not diminish when many virtual machines are deployed at once. In the next test the same template was used but instead of one virtual machine being deployed, 8 virtual machines were concurrently deployed from the template. This process was automated using the following basic PowerCLI script.

```

for ($i=0; $i -le 7; $i++)
{
    New-vm -vmhost 10.21.23.71 -Name "XCOPY_VM1$i" -Template WIN2012R2 -Datastore
    InfrastructurePGRD2 -runasync
}

```

The preceding script deploys all 8 VMs to the same target datastore. It took 13 minutes and 22 seconds for the deployment of 8 VMs with XCOPY disabled to complete and it only took 23 seconds when XCOPY was enabled. For an improvement of about 35x. A single VM deployment improvement (as revealed in the previous example) was about 19x so the efficiency gains actually *improve* as deployment concurrency is scaled up. Non-XCOPY based deployment characteristics (throughput/IOPS/duration) increase in almost a linear fashion along with an increased VM count while XCOPY-based deployment characteristics increase at a much slower comparative rate due to the great ease at which the FlashArray can handle XCOPY operations.

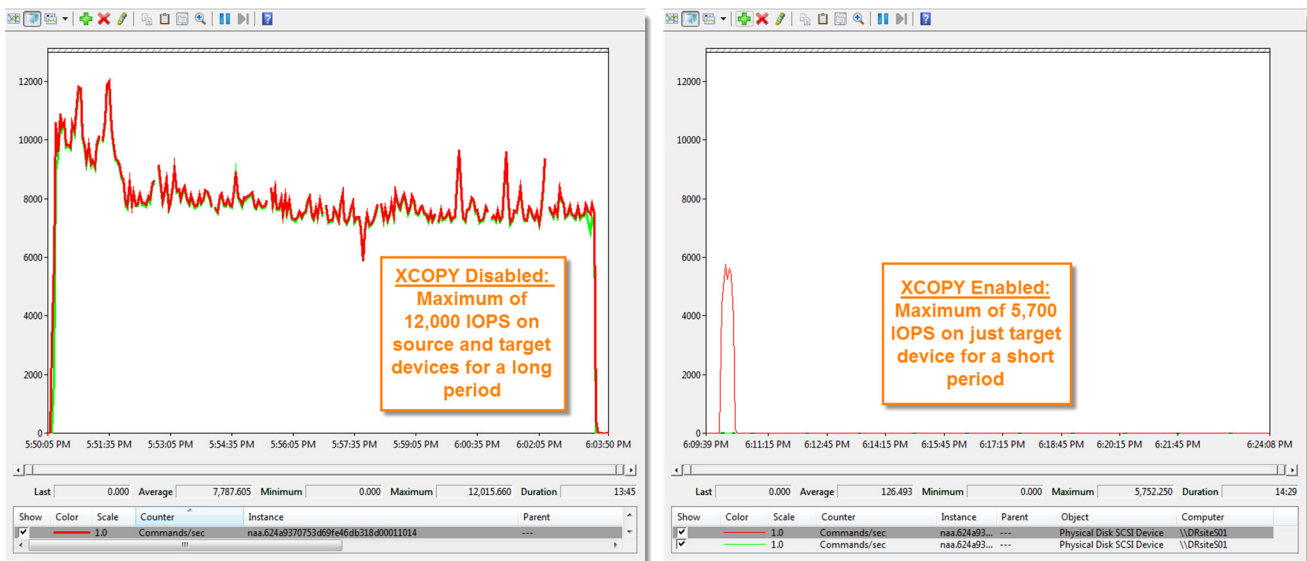


Figure 11. Total IOPS for 8 simultaneous "deploy from template" operations with and without XCOPY enabled

Storage vMotion

Storage vMotion operations can also benefit from XCOPY acceleration and offloading. Using the same VM configuration as the previous example, the following will show performance differences of migrating the VM from one datastore to another with and without XCOPY enabled. Results will be shown for three different scenarios:

1. Powered-off virtual machine
2. Powered-on virtual machine—but mostly idle
3. Powered-on virtual machine running a workload. 32 KB IO size, mostly random, heavy on writes.

The chart below shows the results of the tests.

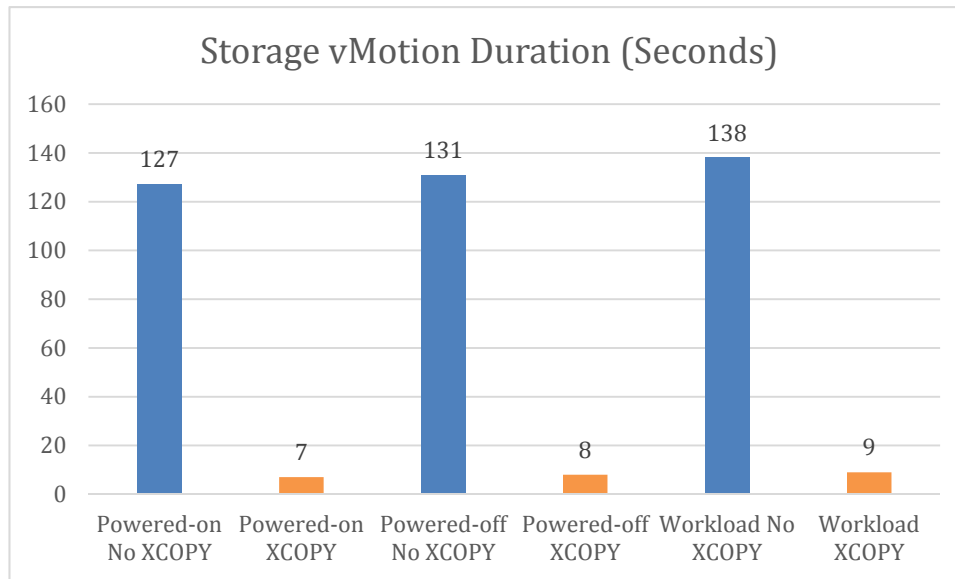


Figure 12. Storage vMotion duration

The chart shows that both a “live” (powered-on) Storage vMotion and a powered-off migration equally can benefit from XCOPY acceleration. The presence of a workload slows down the operation somewhat but nevertheless a substantial benefit can still be observed.

Virtual Disk Type Effect on XCOPY Performance

The allocation method of the source virtual disk(s) can have a perceptible effect on the copy duration of an XCOPY operation. Thick-type virtual disks (such as zeroedthick or eagerzeroedthick) clone/migrate much faster than a thin virtual disk of the same size with the same data⁵. According to VMware this performance delta is a design decision and is to be expected, refer to the following VMware KB for more information:

http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=2070607

The following chart shows the duration in seconds of the three types of virtual disks during a “deploy from template operation”. For comparative purposes it shows the durations for both XCOPY-enabled operations and XCOPY-disabled operations.

⁵ For this reason, it is recommended to never use thin-type virtual disks for virtual machine templates as it will significantly increase the deploy-from-template duration for new virtual machines.

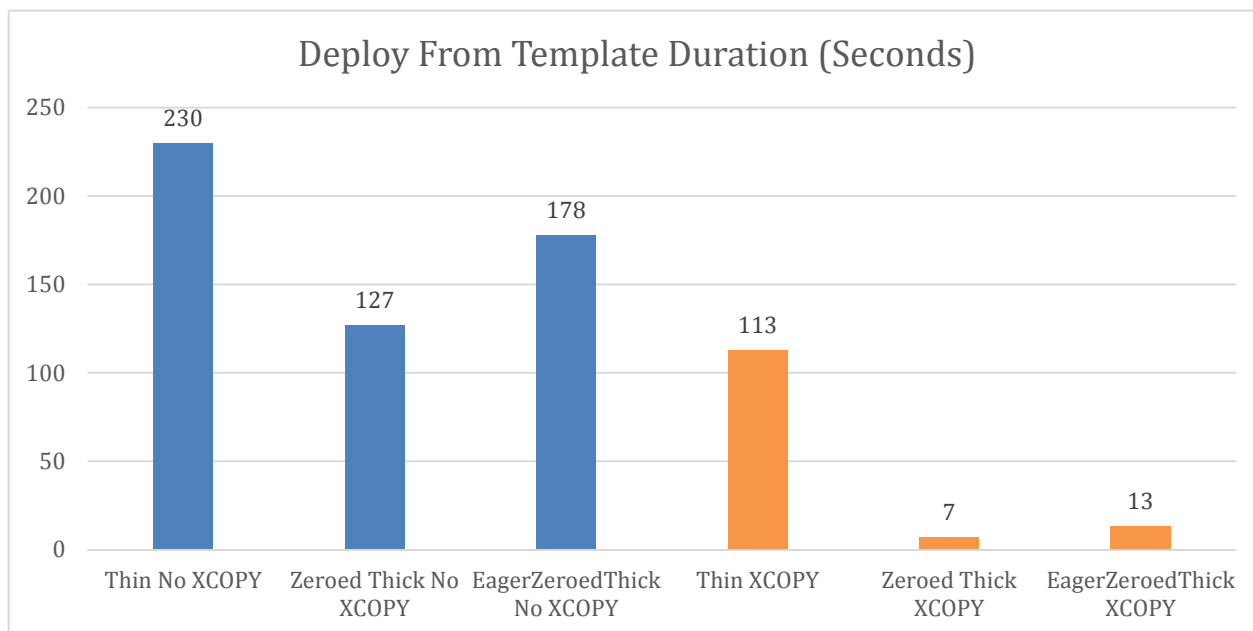


Figure 13. Source virtual disk type effect on XCOPY performance

It can be noted that while each virtual disk type benefits from XCOPY acceleration, thick-type virtual disks benefit the most when it comes to duration reduction of cloning operations. Regardless, all types benefit equally in reduction of IOPS and throughput. Also, standard VM clone or migration operations display similar duration differences as the above “deploy from template” examples.

Block Zero or WRITE SAME

ESXi supports three disk formats for provisioning virtual disks:

1. Eagerzeroedthick—the entirety of the virtual disk upon creation is completely reserved on the VMFS and pre-zeroed. This virtual disk allocation mechanism offers the most predictable performance and highest level of protection against capacity exhaustion.
2. Zeroedthick—this format reserves the space on the VMFS volume upon creation but does not pre-zero the encompassed blocks until the guest OS writes to them. New writes cause iterations of on-demand zeroing in segments of the block size of the target VMFS (almost invariably 1 MB with VMFS 5). There is a slight performance impact on writes to new blocks due to the on-demand zeroing.
3. Thin—this format neither reserves space on the VMFS volume nor pre-zeroes blocks. Space is reserved and zeroed on-demand in segments in accordance to the VMFS block size. Thin virtual disks allow for the highest virtual machine density but provide the lowest protection against possible capacity exhaustion. There is a slight performance impact on writes to new blocks due to the on-demand zeroing.

Prior to WRITE SAME support, the performance differences between these allocation mechanisms were distinct. This was due to the fact that before any unallocated block could be written to, zeroes would have to be written first, causing an allocate-on-first-write penalty. Therefore, for every new block that was to be written to there were two writes, the zeroes then the actual data. For thin and zeroedthick virtual disks this zeroing was on-

demand so the effect was observed by the virtual machine writing to new blocks. For eagerzeroedthick zeroing occurred during deployment and therefore large virtual disks took a long time to create but with the benefit of eliminating any zeroing penalty for new writes.

To reduce this latency, VMware introduced WRITE SAME support. WRITE SAME is a SCSI command that tells a target device (or array) to write a pattern (in this case zeros) to a target location. ESXi utilizes this command to avoid having to actually send a payload of zeros but instead simply communicates to an array that it needs to write zeros to a certain location on a certain device. This not only reduces traffic on the SAN fabric, but also speeds up the overall process since the zeros do not have to traverse the data path.

This process is optimized even further on the Pure Storage FlashArray. Since the array does not store space-wasting patterns like contiguous zeroes, the metadata is created or changed to simply note that these locations are supposed to be all-zero so any subsequent reads will result in the array returning contiguous zeros to the host. This additional array-side optimization further reduces the time and penalty caused by pre-zeroing of newly-allocated blocks.

Performance Examples

The following sections will outline a few examples of WRITE SAME usage to describe expected behavior and performance benefits of using WRITE SAME on the Pure Storage FlashArray.

Deploying Eagerzeroedthick Virtual Disks

The most noticeable operation in which WRITE SAME helps is with the creation of eagerzeroedthick virtual disks. Due to the fact that the zeroing process must be completed during the create operation, WRITE SAME has a dramatic impact on the duration of virtual disk creation and practically eliminates the added traffic that used to be caused by the traditional zeroing behavior.

The following chart shows the deployment time of four differently sized eagerzeroedthick virtual disks when WRITE SAME was enabled (in orange) and disabled (in blue). The enabling of WRITE SAME on average reduces the deployment time of these types of virtual disks to about 6x faster regardless of the size.

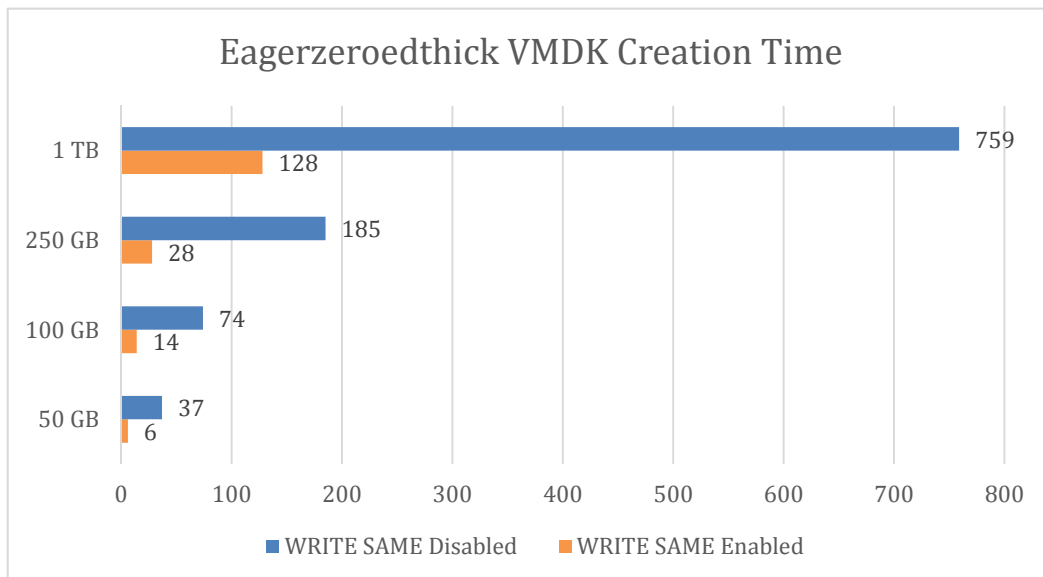


Figure 14. Eagerzeroedthick virtual disk deployment time differences

WRITE SAME also works well in scale on the FlashArray. Below are the results of a test when four 100GB eagerzeroedthick virtual disks were deployed (with vmkfstools) simultaneously.

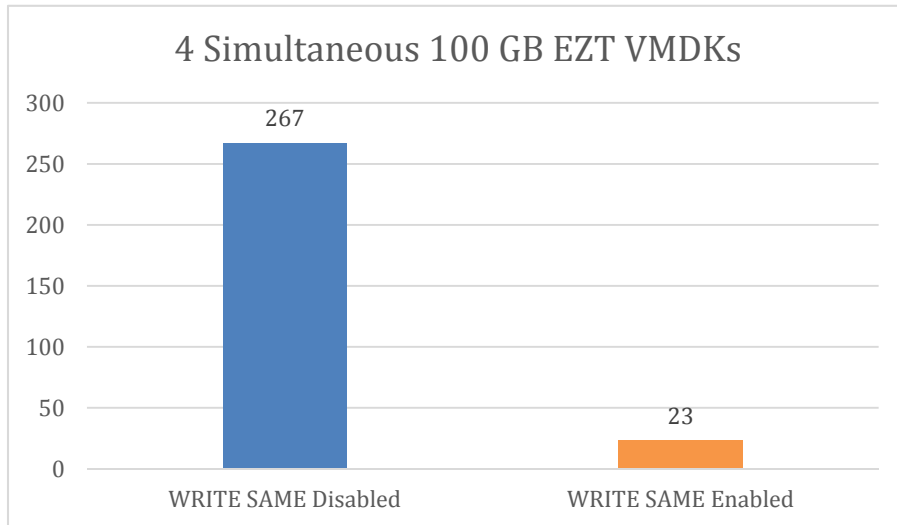


Figure 15. Total simultaneous deployment time for eagerzeroedthick virtual disks

In comparison to the previous chart where only one virtual disk was deployed at a time, the deployment duration of an eagerzeroedthick virtual disk without WRITE SAME increased almost linearly with the added number of virtual disks (took 3.5x longer with 4x more disks). When WRITE SAME was enabled, the increase wasn't even twofold (took 1.6x times longer with 4x more disks). It can be concluded that the Pure Storage FlashArray can easily handle and scale with additional simultaneous WRITE SAME activities.

Zeroedthick and Thin Virtual Disks Zero-On-New-Write Performance

In addition to accelerating up eagerzeroedthick deployment, WRITE SAME also improves performance within thin and zeroedthick virtual disks. Since both types of virtual disks zero-out blocks only upon demand (new writes to previously unallocated blocks) these new writes suffer from additional latency when compared to over-writes. The introduction of WRITE SAME reduces this latency by speeding up the process of initializing this space.

The following test was created to ensure that a large proportion of the workload was new writes so that the write workload always encountered the allocation penalty from pre-zeroing (with the exception of the eagerzeroedthick test which was more or less a control). Five separate tests were run:

1. Thin virtual disk with WRITE SAME disabled.
2. Thin virtual disk with WRITE SAME enabled.
3. Zeroedthick virtual disk with WRITE SAME disabled.
4. Zeroedthick virtual disk with WRITE SAME enabled.
5. Eagerzeroedthick virtual disk

The workload was a 100% sequential 32 KB write profile in all tests. As expected the lowest performance (lowest throughput, lowest IOPS and highest latency) was with thin or zeroedthick with WRITE SAME disabled (zeroedthick slightly out-performed thin). Enabling WRITE SAME improved both, but eagerzeroedthick virtual disks out-performed all of the other virtual disks regardless of WRITE SAME use. With WRITE SAME enabled eagerzeroedthick performed better than thin and zeroedthick by 30% and 20% respectively in both IOPS and throughput, and improved latency from both by 17%. The following three charts show the results for throughput, IOPS and latency.

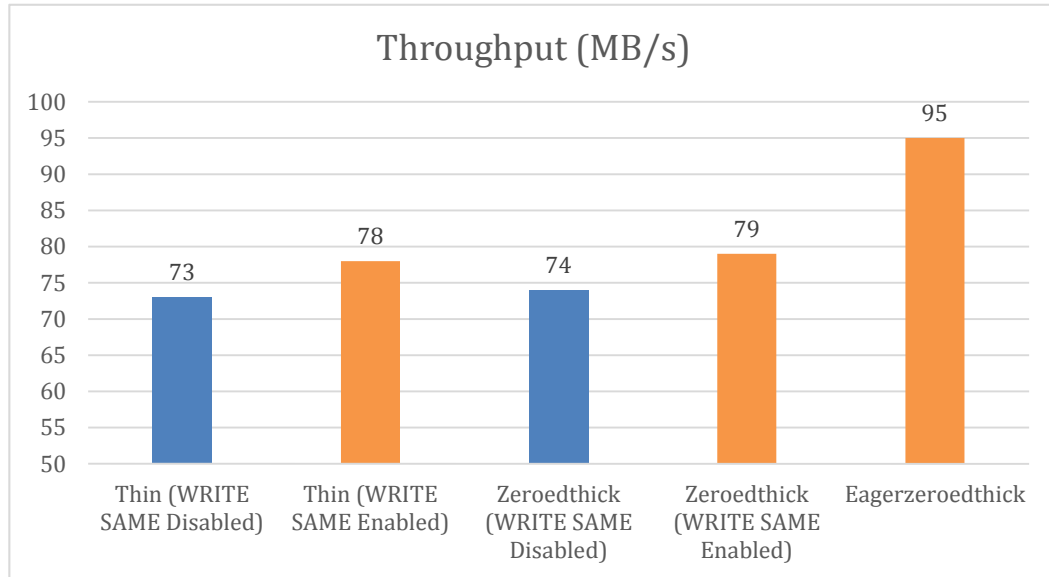


Figure 16. Throughput differences of virtual disk types

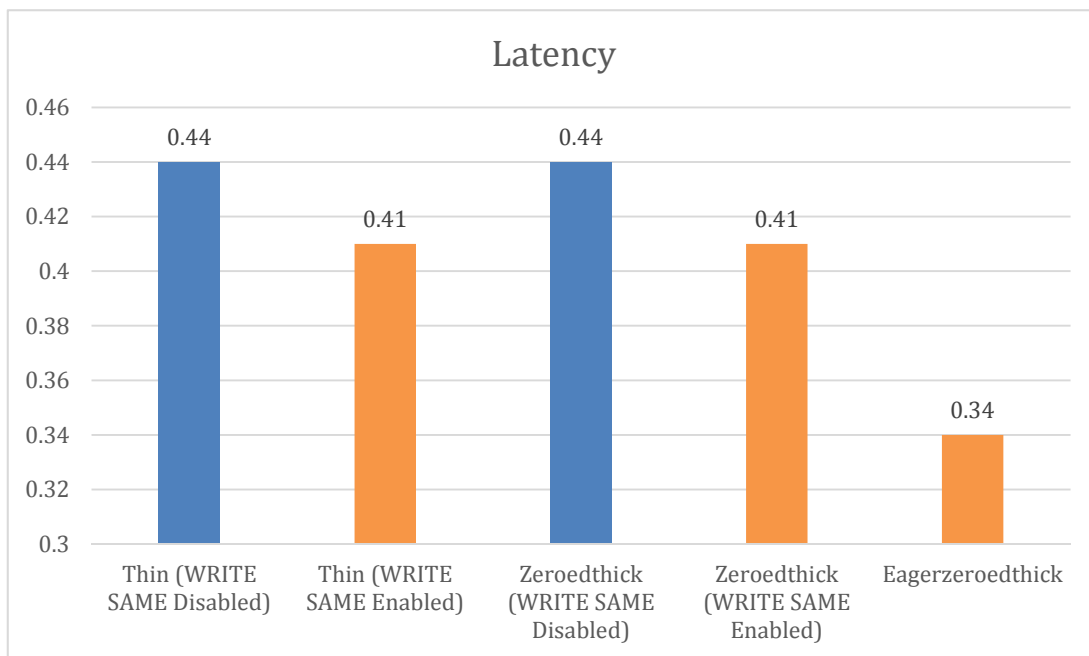


Figure 17. Latency difference of virtual disk type



Figure 18. IOPS differences of virtual disk type

Note that all of the charts do not start the vertical axis at zero—this is to better illustrate the deltas between the different tests.

It is important to understand that these tests are not meant to authoritatively describe performance differences between virtual disks types—instead they are meant to express the performance improvement introduced by WRITE SAME for writes to uninitialized blocks. Once blocks have been written to, the performance difference between the various virtual disk types diminishes. Furthermore, as workloads become more random and/or more read intensive, this overall performance difference will become less perceptible.

From this set of tests we can conclude:

1. Regardless of WRITE SAME status, eagerzeroedthick virtual disks will always out-perform the other types for new writes.
2. The latency overhead of zeroing-on-demand with WRITE SAME disabled is about 30% (in other words the new write latency of thin/zeroedthick is 30% greater than with eagerzeroedthick).
 - a. The latency overhead is reduced from 30% to 20% when WRITE SAME is enabled.
3. The IOPS and throughput reduction caused by zeroing-on-demand with WRITE SAME disabled is about 23% (in other words the possible IOPS/throughput of thin/zeroedthick to new blocks is 23% lower than with eagerzeroedthick).
 - a. The possible IOPS/throughput to new blocks is reduced from 23% to 17% when WRITE SAME is enabled.

Dead Space Reclamation or UNMAP

In block-based storage implementations, the file system is managed by the host, not the array. Because of this, the array does not typically know when a file has been deleted or moved from a storage volume and therefore does not know when or if to release the space. This behavior is especially detrimental in thinly-provisioned environments where that space could be immediately allocated to another device/application or just returned to the pool of available storage.

In vSphere 5.0, VMware introduced Dead Space Reclamation which makes use of the SCSI UNMAP command to help remediate this issue. UNMAP enables an administrator to initiate a reclaim operation from an ESXi host to compatible block storage devices. The reclaim operation instructs ESXi to inform the storage array of space that previously had been occupied by a virtual disk and is now freed up by either a delete or migration and can be reclaimed. This enables an array to accurately manage and report space consumption of a thinly-provisioned datastore and enables users to better monitor and forecast new storage requirements.

To reclaim space in vSphere 5.0 U1 through 5.1, SSH into the ESXi console and run the following commands:

1. Change into the directory of the VMFS datastore you want to run a reclamation on:

```
cd /vmfs/volumes/<datastore name>
```

2. Then run `vmkfstools` to reclaim the space by indicating the percentage of the free space you would like to reclaim (up to 99%):

```
vmkfstools -y 99
```

To reclaim space in vSphere 5.5, the `vmkfstools -y` option has been deprecated and UNMAP is now available in `esxcli`. UNMAP can be run anywhere `esxcli` is installed and therefore does not require an SSH session:

1. Run `esxcli` and supply the datastore name. Optionally a block iteration count can be specified, otherwise it defaults to reclaiming 200 MB per iteration:

```
esxcli storage vmfs unmap -l <datastore name> -n (blocks per iteration)
```

The `esxcli` option can also be leveraged from the VMware vSphere PowerCLI using the cmdlet `GetEsxcli`:

```
$esxcli=get-esxcli -VMHost <ESXi host>
```

```
$esxcli.storage.vmfs.unmap(60000, "<datastore name\"", $null)
```

It should be noted that while UNMAP on the FlashArray is a quick and unobtrusive process, ESXi does create a balloon file when using the `vmkfstools` method to fill the entire specified range of free space for the duration of the reclaim process. This could possibly lead to a temporary out-of-space condition on the datastore if there are thin virtual disks that need to grow. When a datastore contains a large amount of thin virtual disks, large UNMAP reclaim percentages should be entered with care or avoided entirely.

In ESXi 5.5 the new `esxcli` process does not have this issue. The `esxcli` version of UNMAP allows the user to delineate the number of blocks unmapped per iteration of the process. The default value is 200 (which would be 200 MB in most cases) and can be increased or decreased as necessary. Since the UNMAP process is so low

impact on the Pure Storage FlashArray, this value can be increased dramatically to hasten the reclaim process. The time to reclaim typically decreases in a linear fashion as the block count increases. Therefore increasing the block count to something sufficiently higher is recommended. There is no hard and fast rule on what count size to use, though 60,000 is high enough for most volume sizes and anything larger begins to provide diminishing returns.

```
esxcli storage vmfs unmap -l Pure_Datastore -n 60000
```

The UNMAP procedure (regardless of the ESXi version) causes Purity to remove the metadata pointers to the data for that given volume and if no other pointers exist, the data is also tagged for removal. Therefore, used capacity numbers may not change on the array space reporting after an UNMAP operation. Since the data is often heavily deduplicated, it is highly possible that the data that was reclaimed is still in use by another volume or other parts within the same volume on the array. In that case, the specific metadata pointers are only removed and the data itself remains since it is still in use by other volume metadata pointers. That being said, it is still important to UNMAP regularly to make sure stale pointers do not remain in the system. Regular reclamation in the long term allows this data to eventually be removed as the remaining pointers are deleted.

Performance Examples

The following section will outline a few examples of UNMAP usage to describe expected behavior and performance characteristics of using UNMAP on the Pure Storage FlashArray.

For the esxcli UNMAP method the only configurable option is the block count per iteration—and as previously mentioned Pure Storage recommends setting this to a higher number in order to complete the UNMAP operation as fast as possible. The below chart shows the indirect relationship between the duration of the UNMAP operation and the number of blocks per iteration.

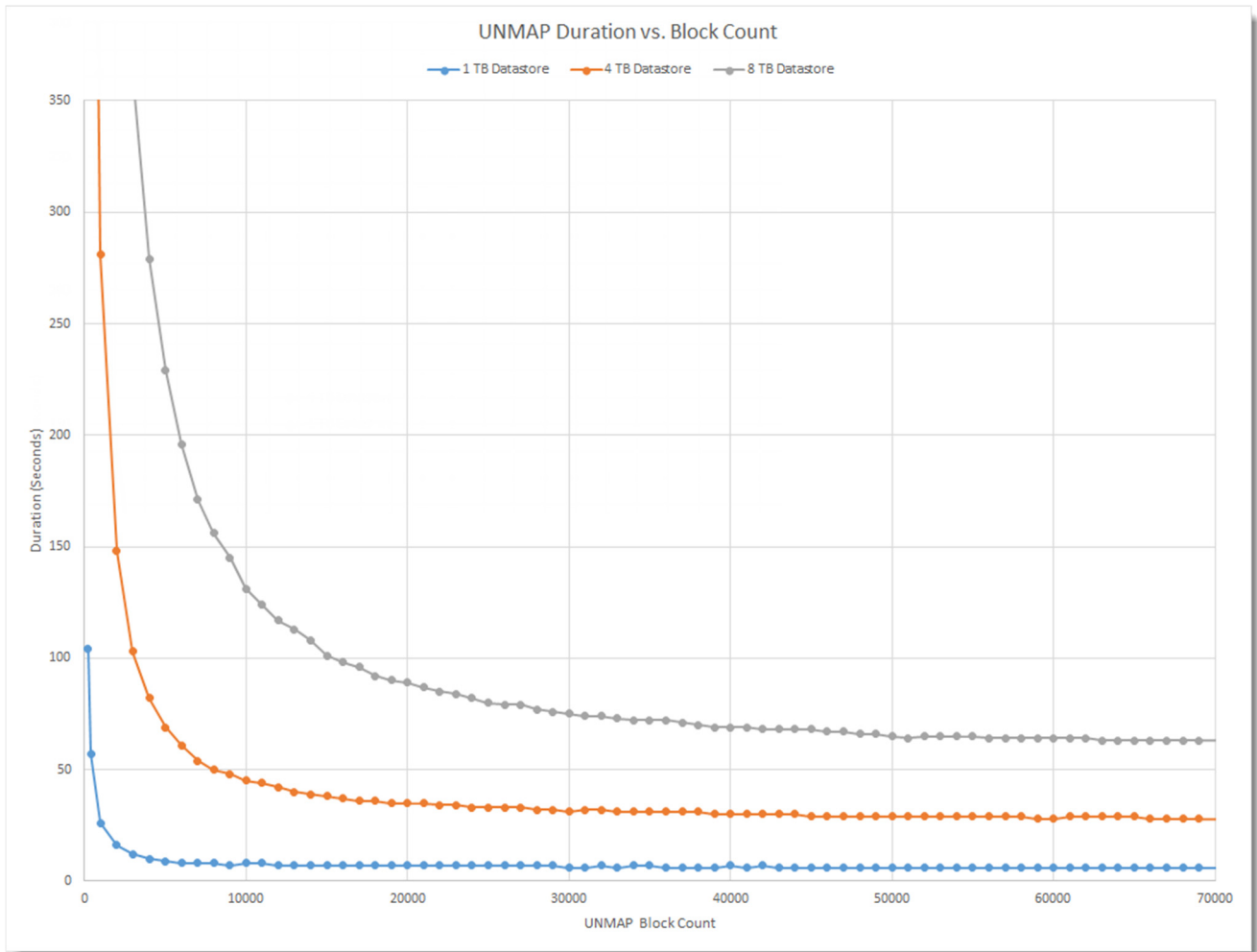


Figure 19. Relationship of block counts and UNMAP duration

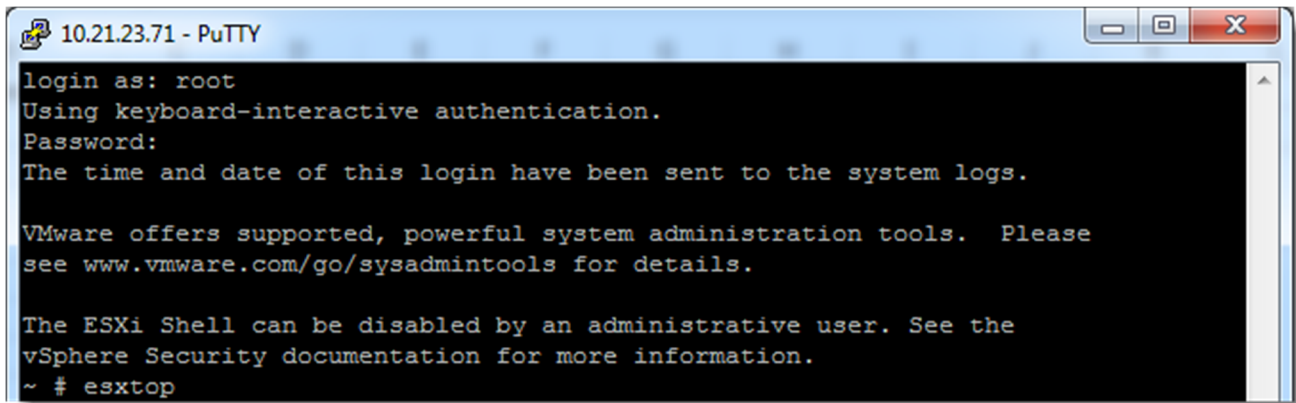
In the tests shown in the above charts three differently-sized datastores were tested with a large variety of block counts. As can be noted, all sizes show great improvement in UNMAP operation duration as the block count increases from 200 (default) to 10,000.

In conclusion, while the size of a datastore does factor in, a block count per iteration of 60,000 should suffice for most datastores. It should be noted though that high levels of fragmentation on a datastore can increase duration times and block count increases may be less effective than what the above chart shows.

Monitoring VAAI with ESXTOP

The simplest way to specifically monitor VAAI activity is through the ESXi performance monitoring tool ESXTOP. ESXTOP is both a real-time monitoring tool and a time-period performance gathering tool.

To monitor activity in real-time, log into the ESXi shell via SSH (SSH is not enabled by default—this can be done through the console or from within the vSphere Web Client in the security profile area) and run the ESXTOP command.



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

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

The ESXi Shell can be disabled by an administrative user. See the
vSphere Security documentation for more information.
~ # esxtop
```

Figure 20. Starting ESXTOP

ESXTOP offers a variety of different screens for the different performance areas it monitors. Each screen can be switched to by pressing a respective key. See the table below for the options.

c	CPU resources
p	CPU power
m	Memory resources
d	Disk adapters
u	Disk devices
v	Virtual machine disks
n	IP Network resources
i	Interrupts

In most cases for VAAI information the disk devices screen is the pertinent one, so press “u” to navigate to that screen. By default, VAAI information is not listed and must be added. To add VAAI information press “f” and then “o” and “p”. To deselect other columns simply type their corresponding letter. Press enter to return.

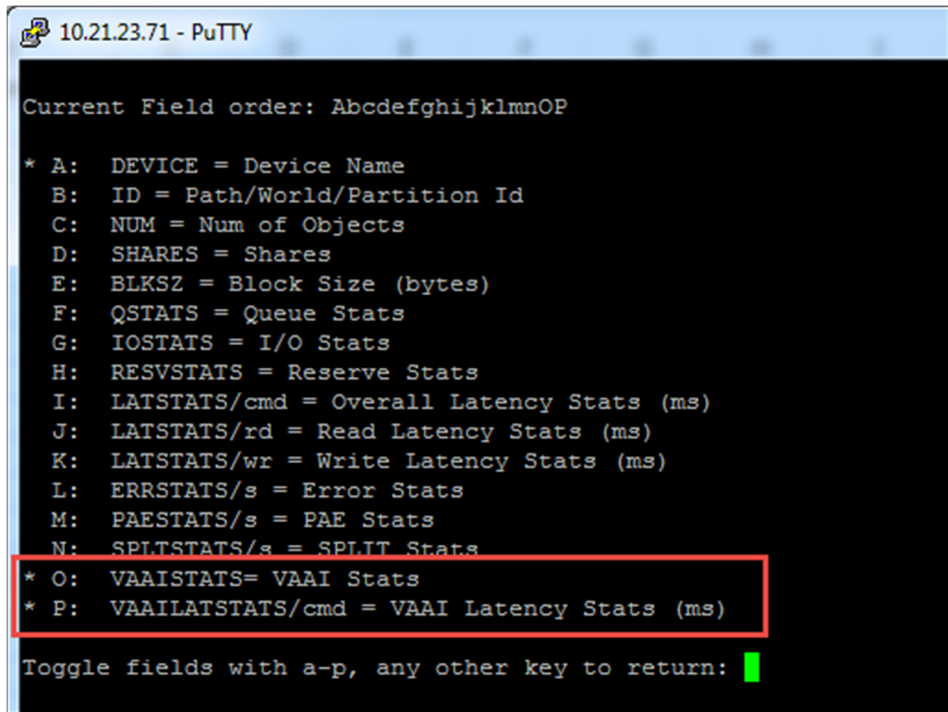


Figure 21. Selecting VAAI statistic columns.

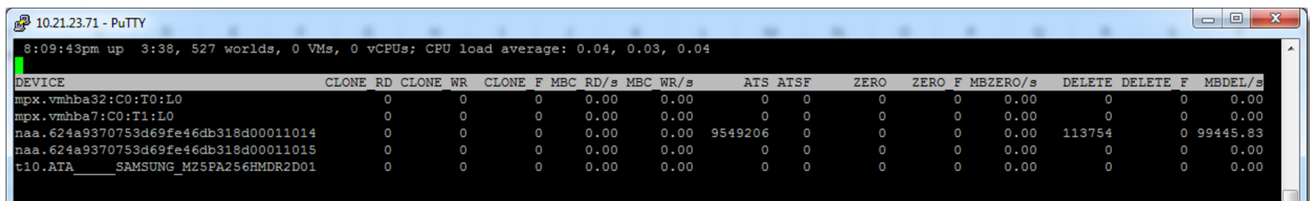


Figure 23. VAAI latency statistics

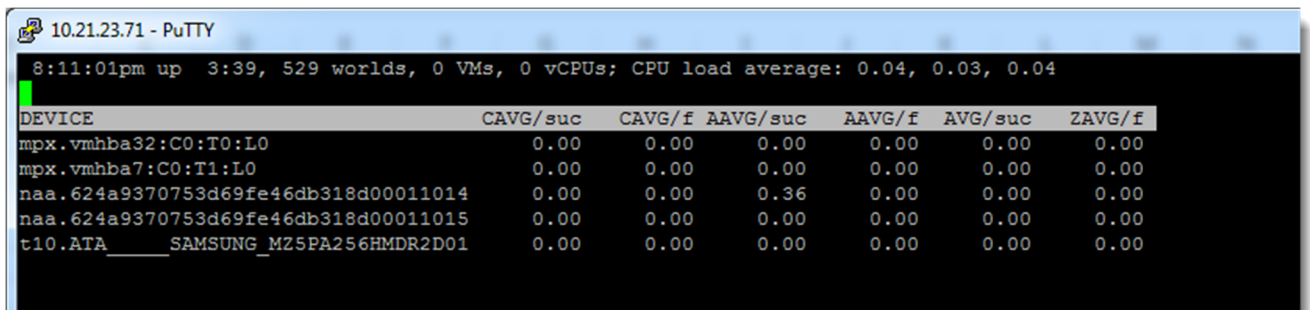


Figure 22. VAAI throughput/OPS statistics

The CLONE or MBC counters refer to XCOPY, ATS refers to Hardware Assisted Locking, ZERO refers to WRITE SAME and DELETE refers to UNMAP. If a collection of this data for later review is preferred over real time analysis esxtop can be gathered for a specific amount of time and saved as a CSV file.

Batch mode, as it is referred to, takes a configuration of ESXTOP and runs for a given interval. This can create a tremendous amount of data so it is advised to remove any and all counters that are not desired using the “f” option and saving the configuration with “W”.

Once the configuration is complete (usually just VAAI info and possibly some other storage counters) ESXTOP can be run again in batch mode by executing:

```
esxtop -b -d 5 -n 50 > esxtopresults.csv
```

The “-b” indicates batch mode, “-d” is a sample period and “-n” is how many intervals should be taken, so in this example it would take counters every 5 seconds, 50 times. So a total capture period of 250 seconds.

Reference

1. Interpreting esxtop statistics - <http://communities.vmware.com/docs/DOC-11812>
2. Disabling VAAI Thin Provisioning Block Space Reclamation (UNMAP) in ESXi 5.0 - kb.vmware.com/kb/2007427
3. Pure Storage and VMware vSphere Best Practices Guide http://info.purestorage.com/WP-PureStorageandVMwarevSphereBestPracticesGuide_Request.html
4. VMware Storage APIs –Array Integration <http://www.vmware.com/files/pdf/techpaper/VMware-vSphere-Storage-API-Array-Integration.pdf>
5. Frequently Asked Questions for vStorage APIs for Array Integration http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1021976



Pure Storage, Inc.
650 Castro Street, Suite #400
Mountain View, CA 94041

T: 650-290-6088
F: 650-625-9667

Sales: sales@purestorage.com
Support: support@purestorage.com
Media: pr@purestorage.com
General: info@purestorage.com