Exploring VMware vSphere Storage API for Array Integration on the IBM Storwize family

Jeremy Canady

IBM Systems and Technology Group ISV Enablement
July 2013

Stay connected with IBM System Storage

Twitter | LinkedIn | Facebook | YouTube | RSS feeds | → Forum | → Community | → More communities

Table of contents

Abstract

The IBM Storwize product family includes support for the VMware vSphere Storage API for Array Integration block primitives. The vSphere Storage API for Array Integration (VAAI) block primitives, such as Atomic Test and Set, Extended Copy, and Write Same enable certain VMware vSphere storage functions to be offloaded from the vSphere ESX or ESXi host to the VAAI-enabled IBM Storwize family storage system. Functions, such as virtual machine (VM) disk creation, VM cloning, and VMware vMotion are able to run on the IBM Storwize family storage system, eliminating resource usage on the vSphere ESX or ESXi host. This white paper provides a guide for VAAI utilization on the IBM Storwize family and an examination of the benefits.

Introduction to the IBM Storwize family

The IBM® Storwize® family provides intelligent storage systems for businesses of all sizes. From the highly scalable midrange IBM Storwize V7000 line to the affordable and efficient IBM Storwize V3700 line and the unified file and block storage of the IBM Storwize V7000 Unified line, the Storwize family contains an appropriate solution.

The IBM Storwize family includes a rich set of advanced storage features that improve application flexibility, responsiveness, availability, and reduced storage utilization. All of these advanced storage features are managed through an easy-to-use interface that is used across the complete Storwize family. A highlight of the advanced storage features is provided in the following list. You can find model-specific capabilities at **ibm.com**/systems/storage/storwize/index.html

- **Metro Mirror** and **Global Mirror** perform synchronous and asynchronous data replication between compatible IBM Storwize storage systems at varying distances to protect data and keep services online in downtime situations.
- **The IBM System Storage® Easy Tier®** feature provides improved performance by migrating frequently used data to high-performance solid-state drives.
- **The IBM FlashCopy®** feature creates instant volume copies allowing for greater flexibility in data protection and testing.
- **IBM Real-time Compression**™ provides a significant reduction in storage requirements by storing more data on the same physical disk.
- **Storage virtualization** enables volume migration and mirroring between any storage that is virtualized by the IBM Storwize system.

Storwize family members

The IBM Storwize family consists of the following models.

IBM Storwize V7000 and Storwize V7000 Unified

IBM Storwize V7000 and Storwize V7000 Unified are midrange storage virtualization systems built from the IBM System Storage SAN Volume Controller virtualization technology and the Redundant Array of Independent Disks (RAID) technology of the IBM System Storage DS8000® storage system. The Storwize V7000 system provides the same advanced storage functionality of IBM SVC, such as Metro Mirror, Real-time Compression, thin provisioning, and non-disruptive data movement.

The Storwize V7000 model consists of 2U drive enclosures. One enclosure contains the storage system controllers as well as 12 or 24 drives depending on the enclosure model (as shown in Figure 1). Nine additional drive enclosures can be added allowing Storwize V7000 to scale up to 240 drives per control enclosure. Additionally, up to four control enclosures can be clustered, allowing Storwize V7000 to scale up to 960 drives.

Figure 1. Storwize V7000 24-disk enclosure

IBM Storwize V7000 Unified combines the block storage capabilities of Storwize V7000 with file storage capabilities into a single system for greater ease of management and efficiency. Storwize V7000 Unified contains the same 2U drive enclosure of Storwize V7000 along with two 2U file modules (as shown in Figure 2) that provide the file services. The complete system is managed from the same Storwize V7000 user interface.

Figure 2. Storwize V7000 Unified

For more information regarding the IBM Storwize V7000 and Storwize V7000 Unified models, refer to: **ibm.com**/systems/storage/disk/storwize_v7000/

IBM Storwize V5000

IBM Storwize V5000 is an entry-level mid-range storage virtualization system built from the IBM SVC virtualization technology and RAID technology of DS8000. The Storwize V5000 system provides optional advanced storage features at a very competitive price.

The IBM Storwize V5000 model consists of 2U drive enclosures (as shown in Figure 3). One enclosure contains the storage system controllers as well as 12 or 24 drives depending on the enclosure model. Six additional drive enclosures can be added, allowing the Storwize V5000 system to scale up to 168 drives per controller enclosure. Additionally, up to two control enclosures can be clustered, allowing Storwize V5000 to scale up to 336 drives.

Figure 3. Storwize V5000 24-disk enclosure

IBM Storwize V3700

IBM Storwize V3700 system is an entry-level storage system designed for ease of use and affordability. IBM Storwize V3700 is built from the IBM SAN Volume Controller virtualization technology and the RAID technology of the IBM System Storage DS8000 storage system. Storwize V3700 provides some of the same advanced storage functionality as IBM Storwize V7000 such as thin provisioning, FlashCopy, Metro Mirror, and a wizard-driven data migration feature to simplify the migration from existing block storage systems.

The IBM Storwize V3700 model consists of 2U drive enclosures (as shown in Figure 4). One enclosure contains the storage system controller and 12 or 24 drives depending on the enclosure model. Four additional drive enclosures can be added, allowing the Storwize V3700 model to scale up to 120 drives per controller enclosure.

Figure 4. Storwize V3700 24 disk enclosure

For more information regarding the IBM Storwize V3700 model, refer to:
ibm.com/systems/storage/disk/storwize_v3700/

Introduction to vSphere Storage API for Array Integration

The vSphere Storage API for Array Integration or VAAI is an application programming interface (API) available to VMware storage partners that allow certain VMware functions to be delegated to the storage array, enhancing performance and reducing load on server and storage area networks (SANs). The Storwize family supports three block primitives of the VAAI.: Atomic Test and Set (ATS), Extended Copy (XCOPY), and Write Same (Zero). The following sections outline the primitives in detail. You can find a graphical overview of it in Figure 5.

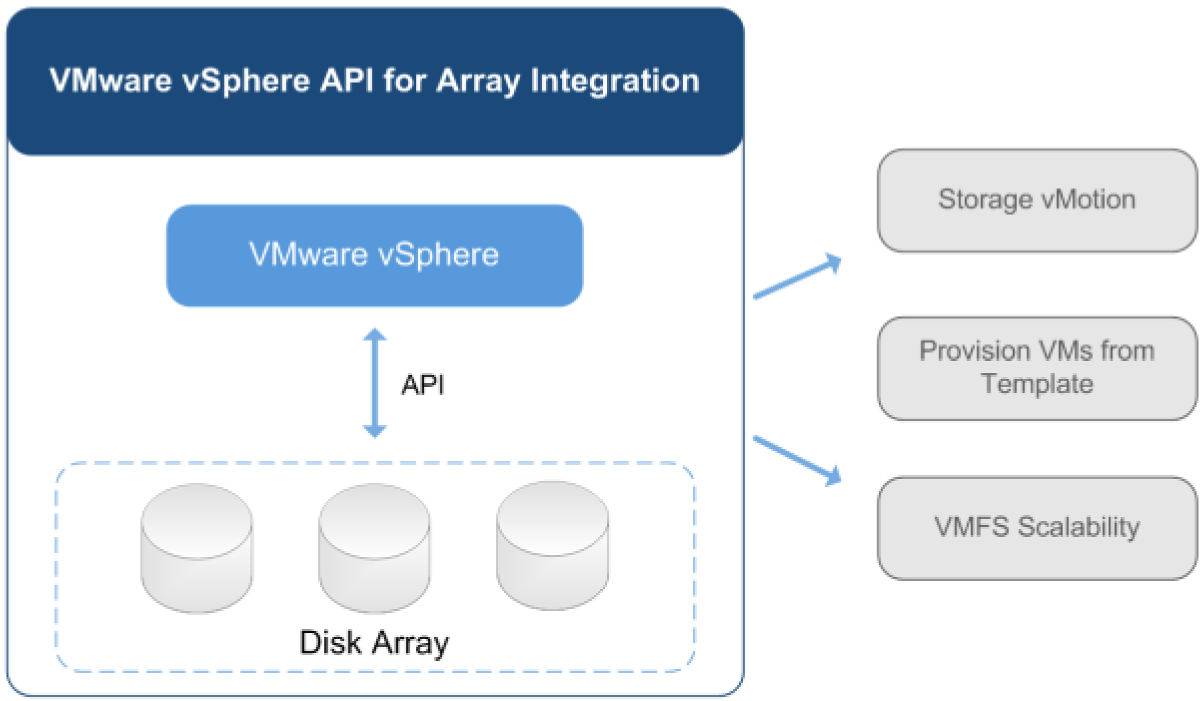


Figure 5. VMware vSphere API for Array Integration relationship to VMware functions

Atomic Test and Set (ATS)

VMware vSphere uses a clustered file system, Virtual Machine File System (VMFS), to provide simultaneous file access to multiple host servers. To prevent virtual machine files and metadata from being modified simultaneously by different host servers, a locking mechanism is required. When no advanced features are present, Small Computer System Interface (SCSI) reservations are used. SCSI reservations place a lock on the entire logical unit number (LUN). This can be seen in Figure 6.

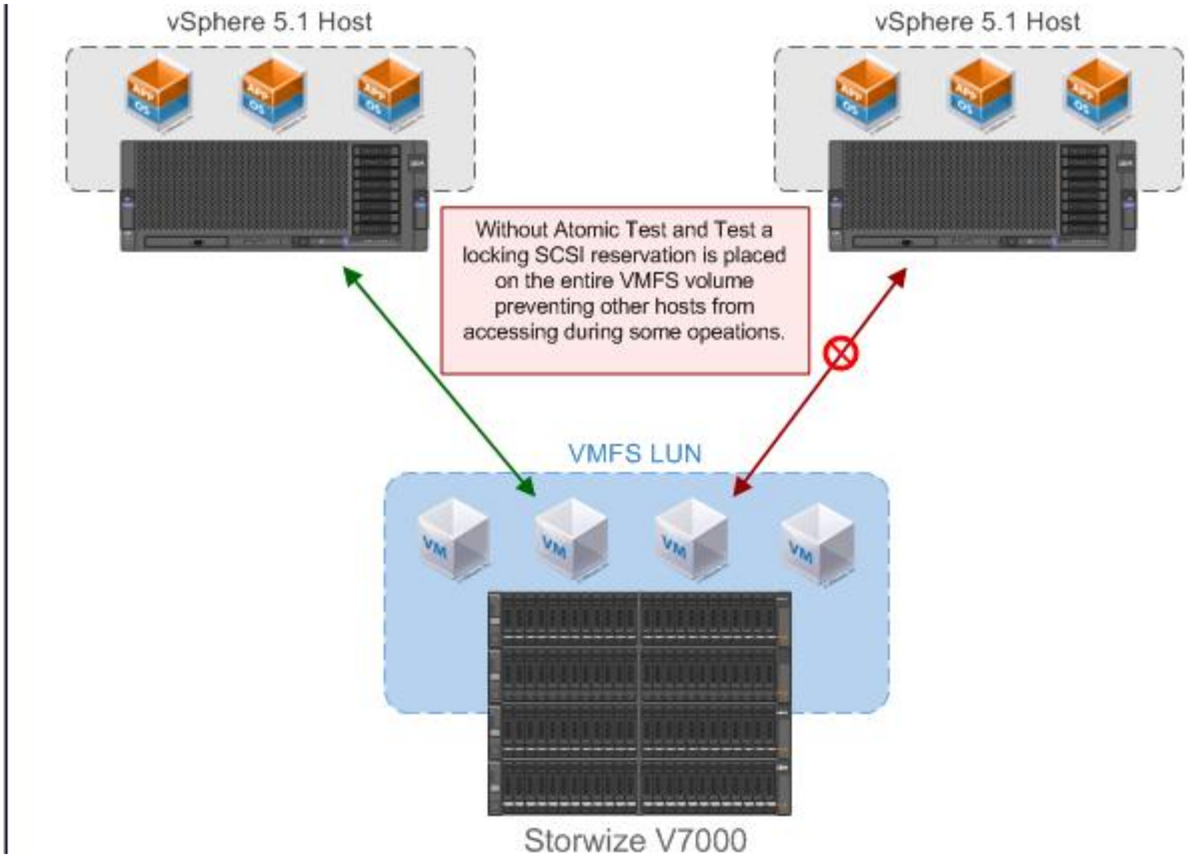


Figure 6: VMFS locking operations without Atomic Test and Set

Many operations such as VM power on, snapshot creation and deletion, and vMotion, require this locking functionality. Although effective in preventing data corruption, contention due to the SCSI reservation method of locking can result in reduced performance when VMFS volumes contain a high number of VMs. Atomic Test and Set eliminates this problem by providing a more granular way to control access to the VMFS metadata, which reduces the contention. This can be seen in Figure 7. This primitive enhances the performance in many ways and significantly increases the scalability of VMFS volumes.

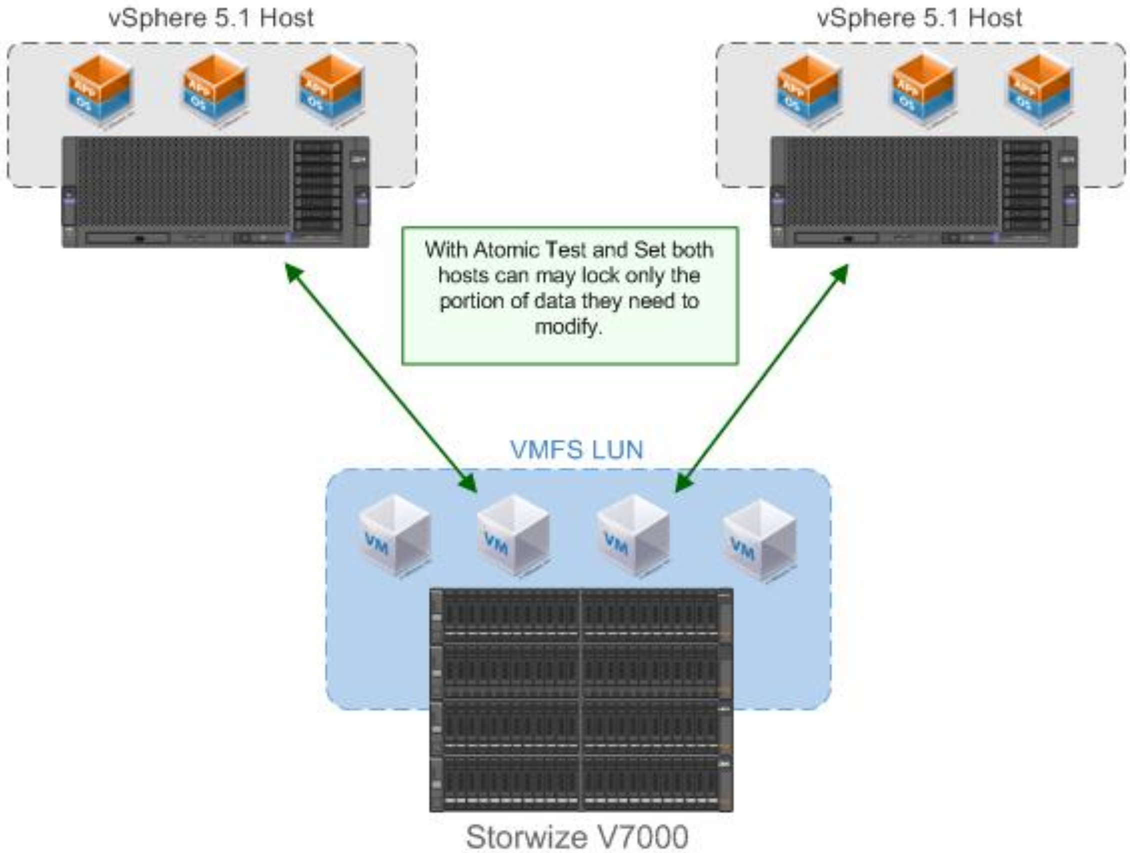


Figure 7 VMFS locking operations with Atomic Test and Set

Extended Copy (XCOPY)

When a new VM is created by cloning a template or VM, the data is read from the source up through the vSphere host and then back down to the destination, as shown in Figure 8.

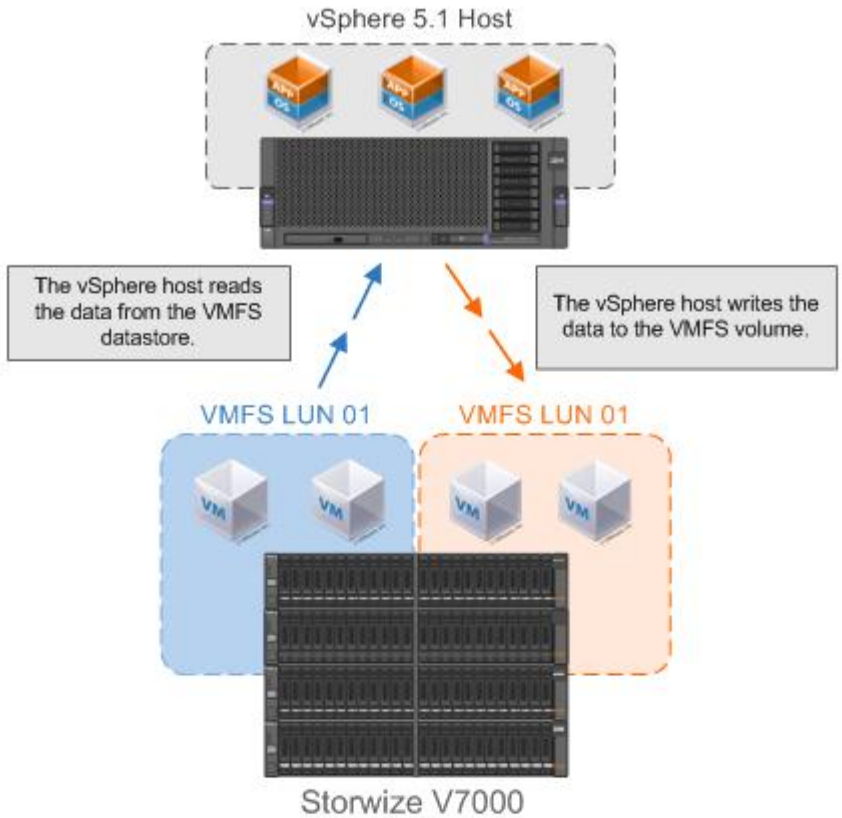


Figure 8: VM cloning without XCOPY

This process consumes both host processor resources and host storage bandwidth. The Extended Copy primitive allows the data-moving operation to run on the storage array and not through the vSphere host. The vSphere host initiates and tracks the progress of the task while the storage array performs the work of duplicating the data, as shown in Figure 9.

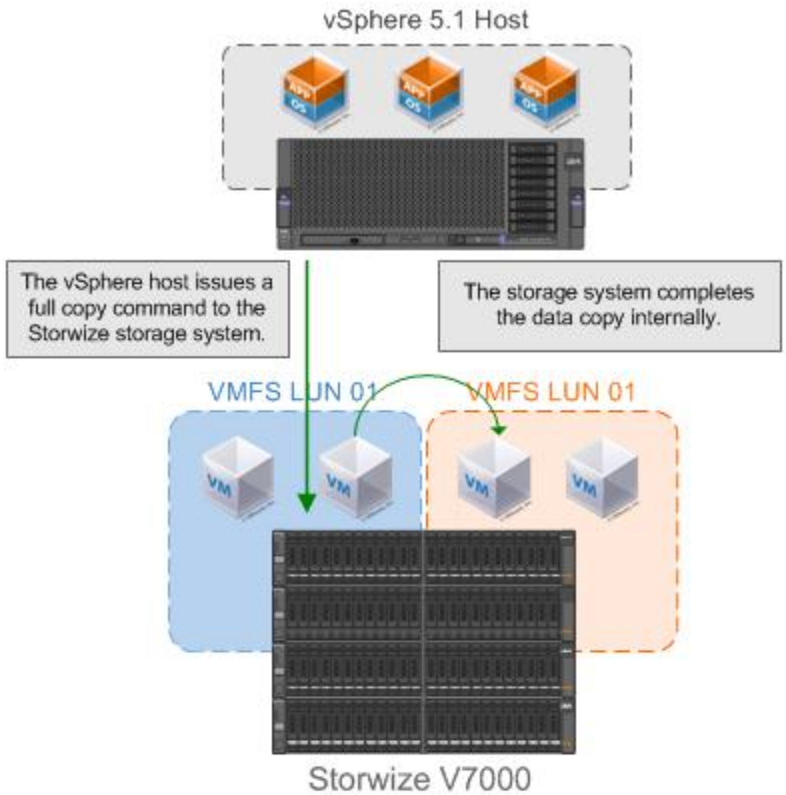


Figure 9 VM cloning with XCOPY

Advantages of XCOPY are not limited to the VM creation. Other storage tasks such as Storage vMotion can use XCOPY. When a migration task for a VM is submitted, the vSphere host can use the XCOPY primitive to perform the data copy to the new location. The only limitation is that the source and destination must be on the same array.

The XCOPY primitive greatly reduces the SAN traffic required for cloning and migration operations, while also saving processor resources on vSphere hosts. All of these efficiencies also mean that cloning and migration tasks that are run with the XCOPY primitive can be completed significantly faster.

Write Same (Zero)

Certain operations within VMware vSphere, such as deploying a VMware Fault Tolerance (FT) compliant VM or creating a new fully allocated virtual disk require that the VMDK file be provisioned as *eagerzeroedthick*. By default, the VMware VMDK files are provisioned as *zeroedthick*. In the *zerodthick* format the space for the VMDK file is fully allocated but the blocks are not zeroed until just before their first access. In the *eagerzeroedthick* format, the VMDK file is fully allocated and all blocks are zeroed immediately. As shown in Figure 10, without Write Same (Zero), commands must be issued from the vSphere host to the storage subsystem for each block of data being formatted. When enabled, the Write Same primitive allows one command to be issued from the vSphere host, and the storage subsystem formats all the blocks without further commands. This primitive eliminates redundant traffic between the vSphere host and storage subsystem and decreases the amount of time required for the task to complete.

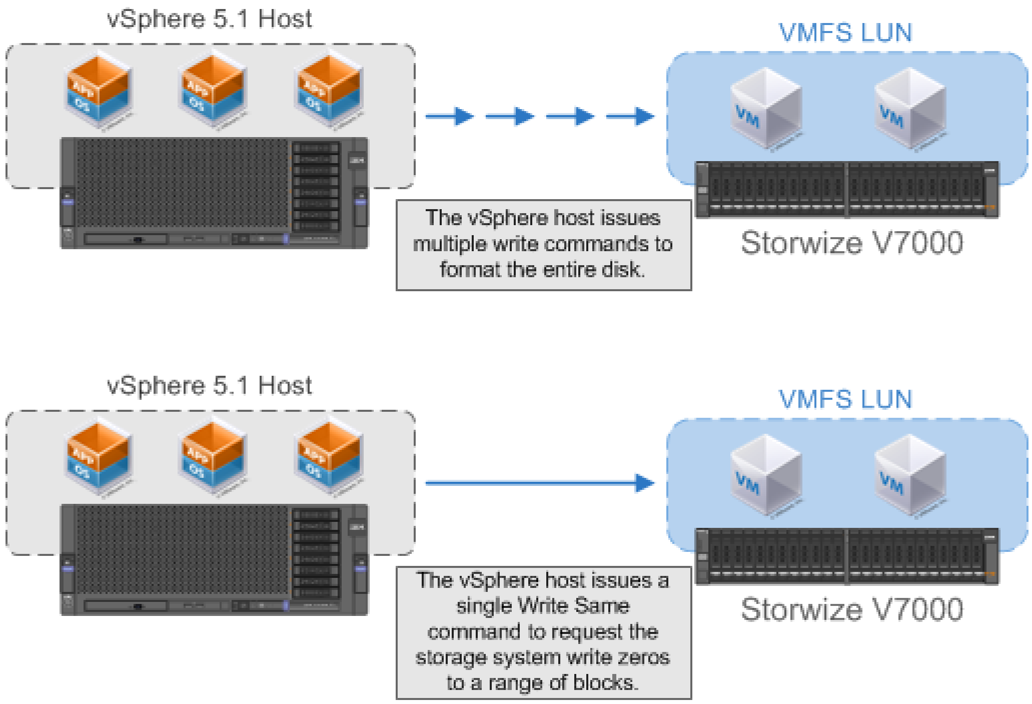


Figure 10: Formatting an eagerzerodthick disk with and without Write Same (Zero)

Enabling vSphere Storage API for Array Integration

Enabling vSphere API for Array Integration differs between different ESXi versions. The original implementation in ESXi 4.1 required a storage system specific device driver to be installed. With the release of ESXi 5.0, the device driver requirement was removed.

ESXi 5.0 and later enablement

vSphere hosts running ESXi 5.0 and later do not require administrative action to enable the vSphere Storage API for Array Integration with the IBM Storwize family. ESXi 5.0 and later uses standard SCSI primitives defined by the T10 technical committee. The Storwize family supports the primitives used by vSphere Storage API for Array Integration without any specific configuration. However, there are a few requirements that must be met.

- vSphere hosts must be running ESXi 5.0 or later.
- The Storwize V7000 storage system must be running a firmware version supported by the ESXi version. You can find the supported configurations at: vmware.com/resources/compatibility/search.php?deviceCategory=san

ESX and ESXi 4.1 enablement

vSphere hosts running ESX and ESXi 4.1 requires very little administrative action. To enable vSphere Storage API for Array Integration support for the Storwize family, the IBM Storage Device Driver for VMware vStorage API must be installed on the ESX or ESXi host.

The specific requirements are as follows:

- vSphere hosts must be running ESX or ESXi 4.1.
- vSphere hosts must have the IBM Storage Device Driver for VMware vStorage API for Array Integration installed. You can find the instructions for installing this driver at: http://delivery04.dhe.ibm.com/sar/CMA/SDA/03gql/0/IBM_DD_for_VMware_VAAI_1.2.0_IG.pdf
- The Storwize V7000 storage system must be running a firmware version supposed by the ESXi version. You can find the supported configurations at: vmware.com/resources/compatibility/search.php?deviceCategory=san

Managing vSphere Storage API for Array Integration

vSphere Storage API for Array Integration is a feature that after its enablement, should not require additional management. However, should the need arise, the three primitive configurations can be modified through the vSphere client and through the command line.

Controlling vSphere Storage API for Array Integration through the vSphere client

You can enable and disable the vSphere Storage API for Array Integration primitives for Write Same (Zero) and Extended Copy (XCOPY), namely DataMover.HardwareAcceleratedInit and DataMOver.HardwareAcceleratedMove respectively, in the vSphere host advanced settings as show in Figure 11.

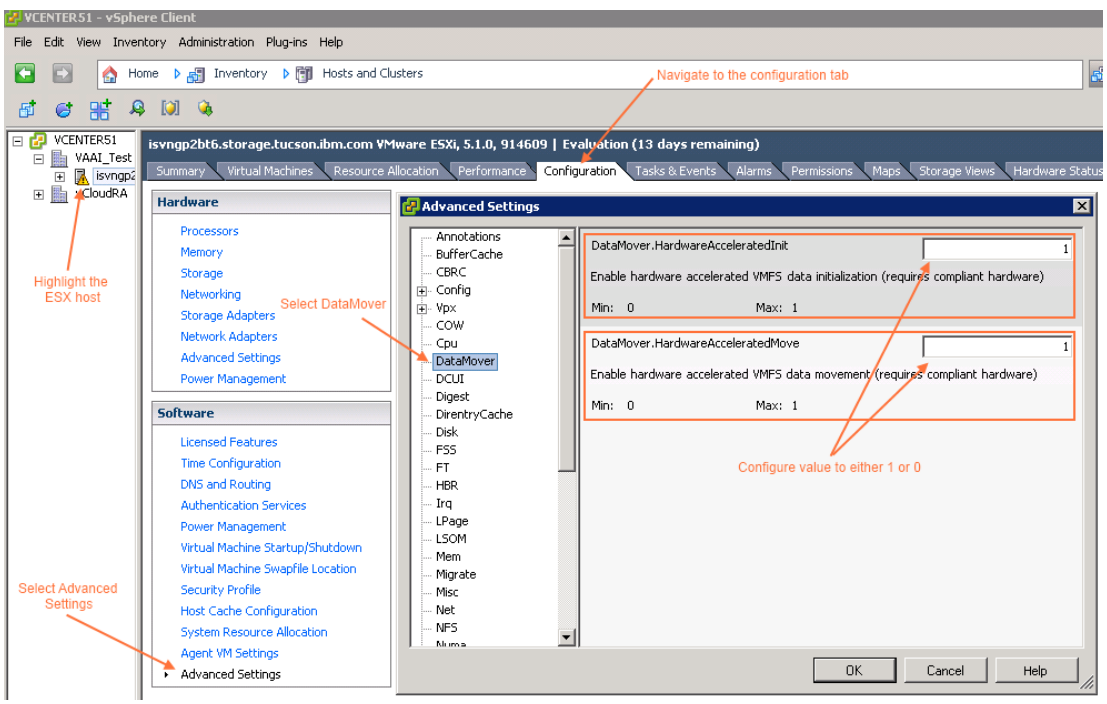


Figure 11. Controlling Extended Copy (XCOPY) and Write Same (Zero)

You can control Atomic Test and Set by changing the **VMFS3.HardwareAcceleratedLocking** setting as shown in Figure 12.

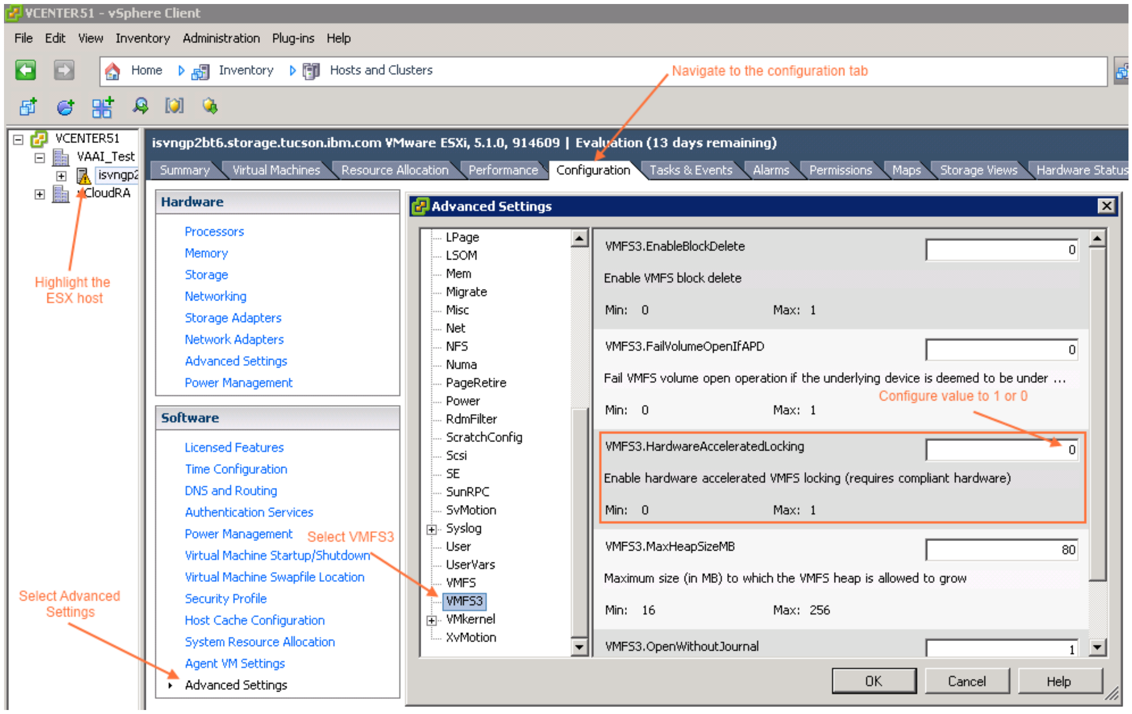


Figure 12. Controlling Atomic Test and Set

Controlling vSphere Storage API for Array Integration through the command line

The vSphere Storage API for Array Integration primitives can also be controlled through the command-line interface. The commands differ between ESXi 4.1 and ESX 5.0 and later.

ESXi 5.0 and later

- To view the status of a setting, use the **esxcli** command with the *list* option.

```
~# esxcli system settings advanced list -o
/DataMover/HardwareAcceleratedMove
Path: /DataMover/HardwareAcceleratedMove
   Type: integer
   Int Value: 0
   Default Int Value: 1
   Min Value: 0
   Max Value: 1
   String Value:
   Default String Value:
   Valid Characters:
   Description: Enable hardware accelerated VMFS data movement (requires
compliant hardware)
```

- To change a setting, use the **excli** command with the *set* option.

```
~# esxcli system settings advanced set -o
"/DataMover/HardwareAcceleratedMove" -i 1
```

```
~# esxcli system settings advanced set -o
"/DataMover/HardwareAcceleratedMove" -i 0
```

ESX and ESXi 4.1

- To view the status of a setting, use the **esxcfg-advcfg** command with the *–g* option:

```
~ # esxcfg-advcfg -g /DataMover/HardwareAcceleratedMove
Value of HardwareAcceleratedMove is 1
```

- To change a setting, use the **esxcfg-advcfg** command with the **–s** option:

```
~ # esxcfg-advcfg -s 0 /DataMover/HardwareAcceleratedMove
Value of HardwareAcceleratedMove is 0
~ # esxcfg-advcfg -s 1 /DataMover/HardwareAcceleratedMove
Value of HardwareAcceleratedMove is 1
```

Validation tests for vSphere Storage API and Array Integration

The VMware tasks that are impacted by vSphere Storage API for Array Integration typically receive significant performance and offload benefits that are easily identified in production environments. For this white paper, several test cases were constructed in a lab environment to demonstrate the benefits.

ATS test cases

The benefits of ATS can be seen in scenarios that contain multiple hosts accessing a shared VMFS data store. Some VMFS operations require updates to the metadata within the file system. To prevent multiple hosts from updating the same data at the same time, a locking mechanism must be used. Without ATS, VMFS relies upon a SCSI reservation, which essentially locks the whole data store. The locking of the data store can cause a conflict with other hosts that also need to access the data store. The tests cases in this white paper focus on the effects of creating an artificial locking demand and using snapshots to generate the locking demand. The test cases were first ran with the ATS primitive disabled and then again with the primitive enabled. The details and results of the test cases are explained in this section.

Test case 1: I/O operations during artificially generated locks

Test case 1 consisted of two ESXi 5.1 hosts, host A and host B, sharing a VMFS data store. A VM running on host A was first configured with a virtual disk located on the VMFS data store. Next, the Iometer workload tool was installed and configured to issue sequential 4 KB reads to the attached drive. The host B was configured with a Bash script designed to artificially generate locks on the shared VMFS data store. The script continually ran the touch command on a text file located on the shared VMFS data store. The touch command updates the file's access and modification timestamps, actions that require a lock. The testing process was run with ATS enabled, disabled, and with no locking generation. During the testing, the Iometer workload input/output operations per second (IOPS) and throughput were measured. Additionally, the **esxitop** tool was used to monitor and measure the reported reservation conflicts per second (CONS/s).

Testing revealed a significant improvement in performance with ATS enabled. With ATS disabled and the artificial workload applied, the average IOPS generated by the Iometer workload tool was found to be 1345 IOPS and the throughput was 5.51 MBps. With ATS enabled, the average IOPS increased by over 400% to 6758.94 IOPS while the throughput also increased by over 400% to 27.68 MBps. This can be seen in Figure 13 and Figure 14. The severity of the artificially generated locking can be seen in the resulting number of reservation conflicts per second.

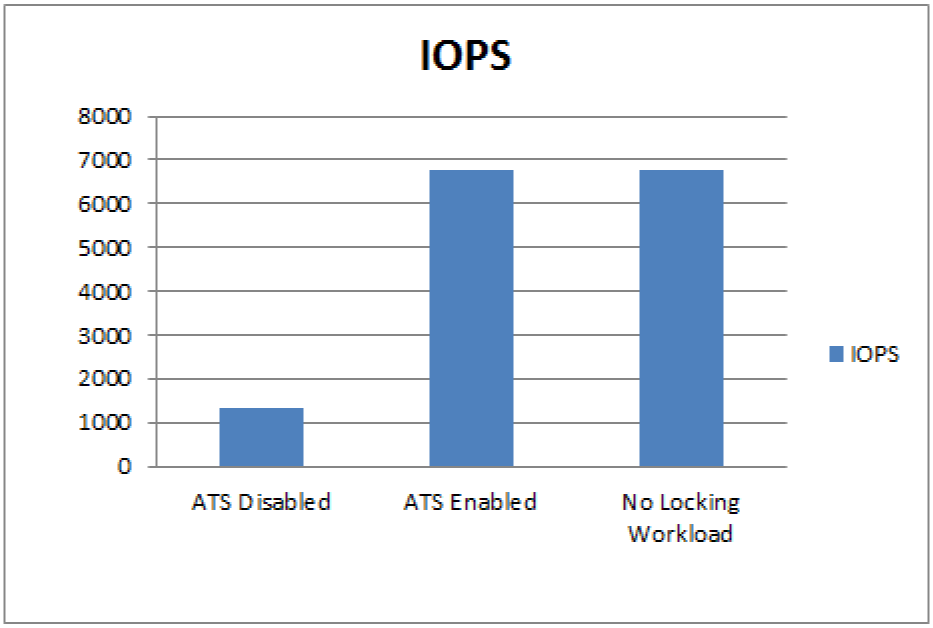


Figure 13. ATS impact on IOPS with an artificial workload

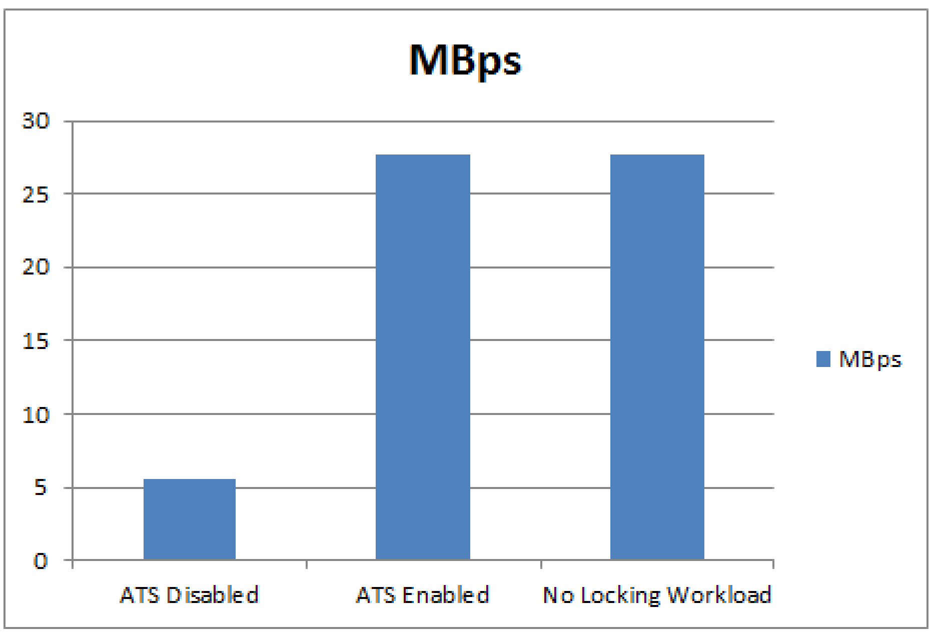


Figure 14. ATS impact on throughput with an artificial workload

During the ATS disabled testing, the reservation conflicts per second were found to be 32 CONS/s as shown in Figure 15. ATS is design to eliminate conflicts so it is no surprise to see zero conflicts with ATS enabled.

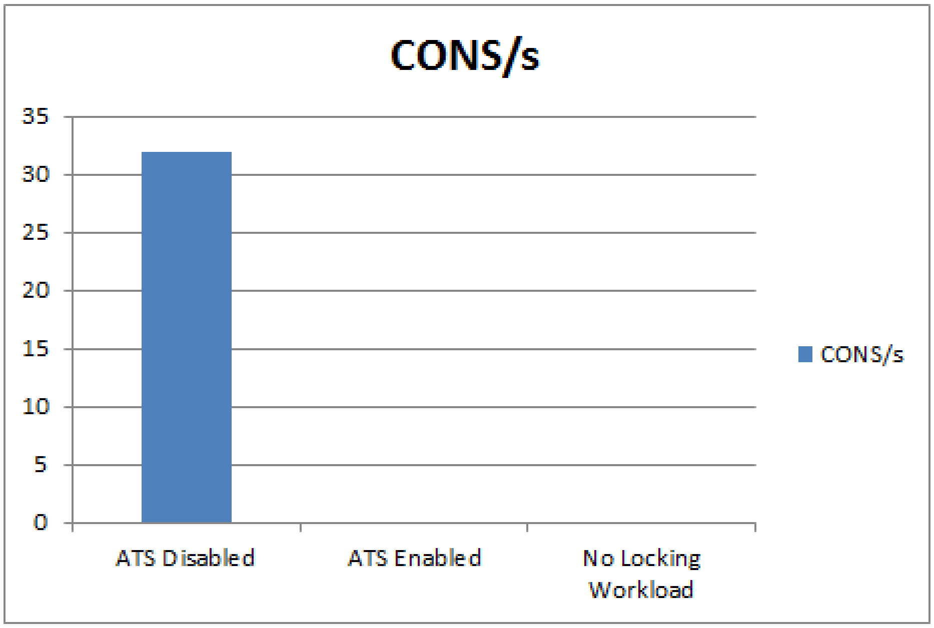


Figure 15. ATS impact on reservation conflicts per second with an artificial workload

Test case 2: I/O simulated backup snapshot workload

VMware vSphere VM backups generally rely upon the VM snapshot feature of vSphere. Snapshots create a point-in-time copy of the VM, allowing the backup software to perform the backup on a consistent set of data. The backup process generally follows three steps:

1. Create virtual snapshot
2. Create virtual machine backup
3. Remove virtual machine snapshot

The snapshot creation and removal process require locks to be placed on various portions of data on the VMFS data store. These locks can become a source of contention resulting in decreased performance. Test case 2 was created to simulate a snapshot workload to determine the effect of Atomic Test and Set.

Test case 2 consisted of two ESXi 5.1 hosts sharing a VMFS data store. A VM running on host A was configured with Iometer and had an attached drive located on the shared VMFS data store. Iometer was configured to issue sequential 4 KB reads to the attached drive. Host B had ten running VMs located on it. All ten virtual machine virtual disks were located on the shared VMFS datastore. To generate the locking load, a PowerShell script was created that used the VMware vSphere PowerCLI to continually issue snapshots to the 10 VMs. The testing process was run with ATS enabled, disabled, and with no locking generation. During the testing, the Iometer workload IOPS and the throughput were measured. Additionally, the severity of the reservation conflicts was measured by the **esxitop** tool.

Testing revealed a notable improvement in performance with ATS enabled. With ATS disabled and the snapshot workload applied, the average IOPS was found to be 4380 IOPS and throughput was 17 MBps. This can be seen in Figure 16 and Figure 17. With ATS enabled, the average IOPS increased by 56% and the throughput increased by 64%.

The resulting increase in performance is less dramatic than test case 1; however that is to be expected. Test case 2 represents a more *real-world* example of how ATS reduces contention. Test case 1 should be viewed as a worst-case scenario.

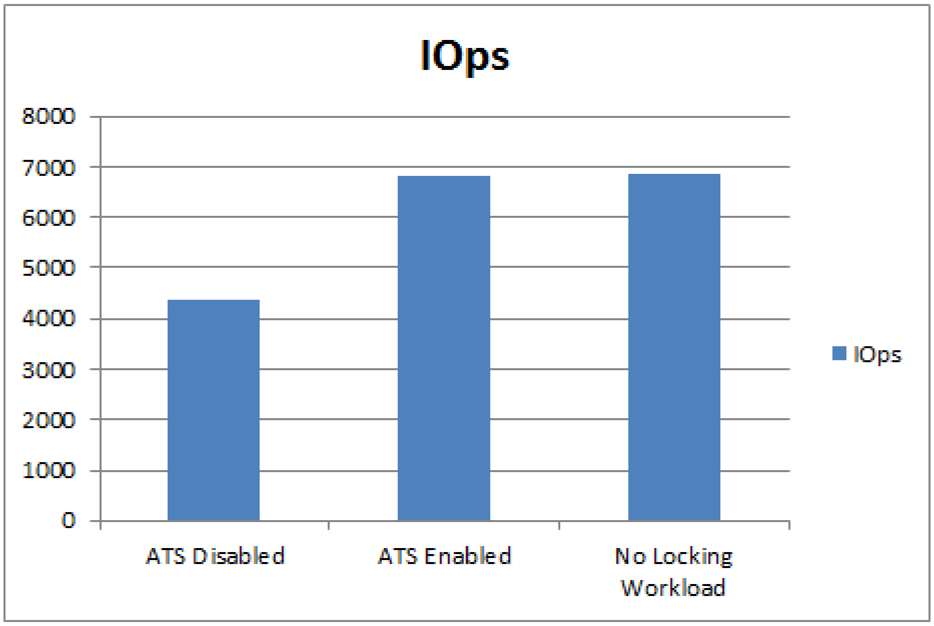


Figure 16. ATS impact on IOPS with a snapshot workload

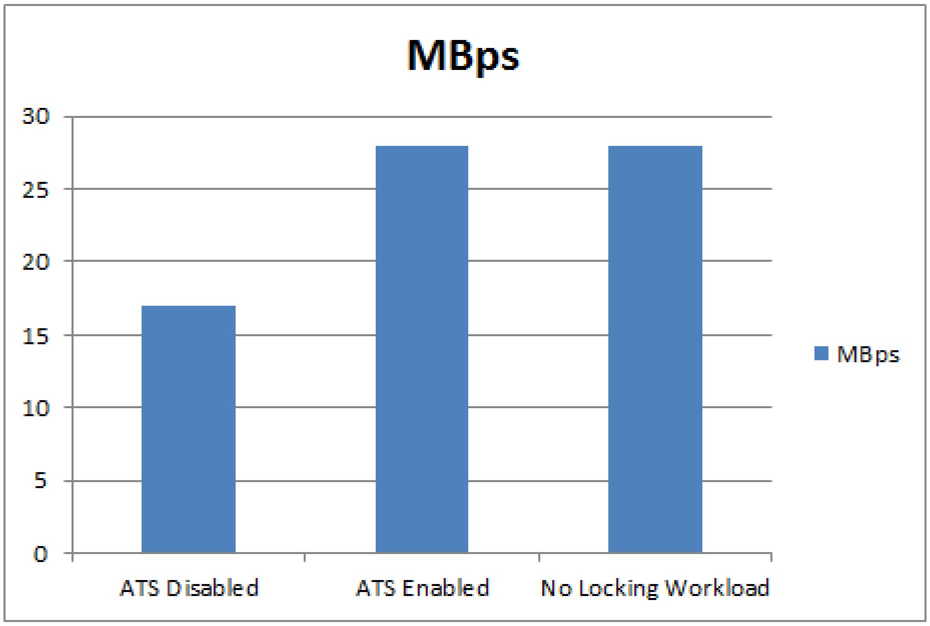


Figure 17. ATS impact on throughput with a snapshot workload

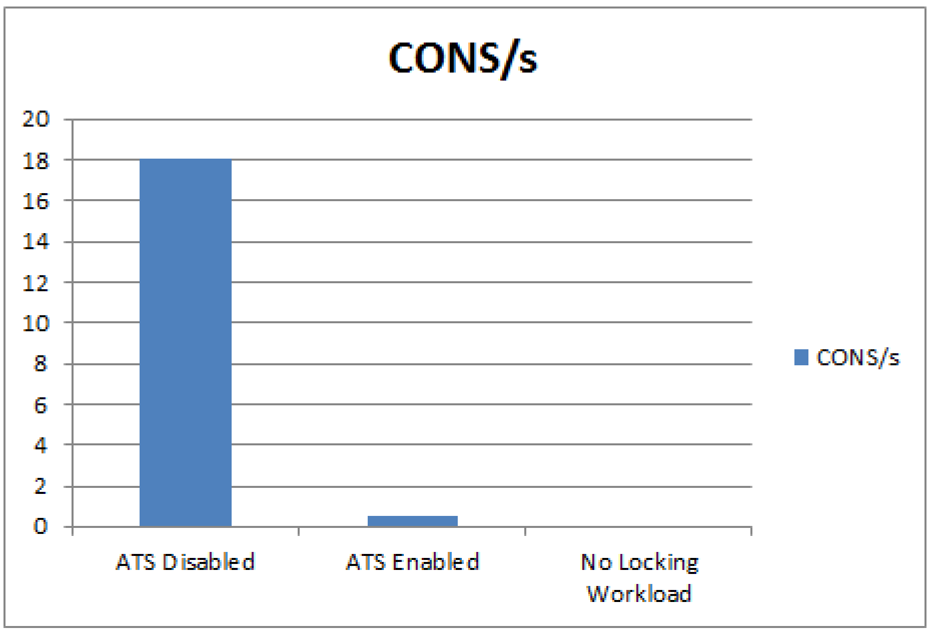


Figure 18. ATS impact on reservation conflicts per second with a snapshot workload

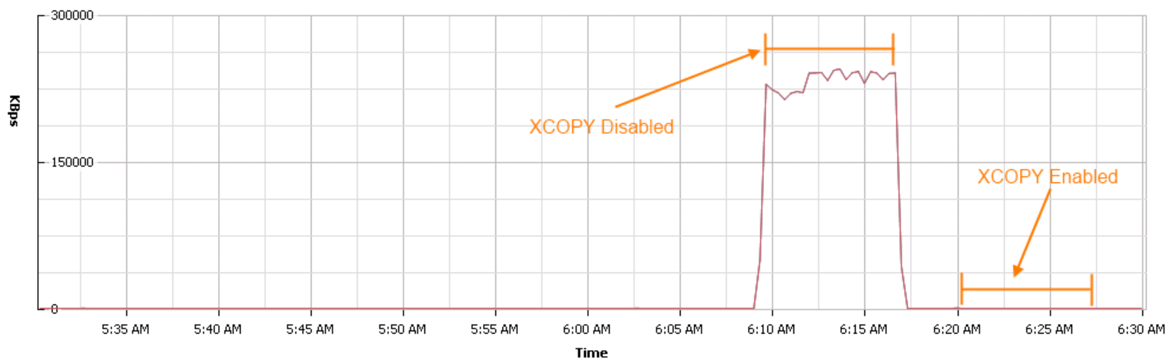
Extended Copy (XCOPY) test cases

The primary scenarios that demonstrate the benefits of Extended Copy (XCOPY) are cloning and migration. When a VM is cloned or migrated between VMFS data stores, the XCOPY primitive allows the data copy operation to be offloaded to the storage system. The test cases in this white paper focus on deploying a new VM from a template and performing a Storage vMotion of a VM. The tests were run first with XCOPY disabled, and then again with the XCOPY primitive enabled. The results of the tests were then compared.

Test case 1: Deploying a VM from a template

For test case 1, a VM was created with an attached 100 GB *eagerzeroedthick* virtual disk. The VM was then converted to a template. A new VM was deployed from this template using the **same format as source** option selected. This ensures that the resulting VM would also be created with an *eagerzeroedthick* disk.

The first benefit observed was the offload of the cloning operation to the storage system. With XCOPY disabled, the vSphere host must read from the source template and write to the destination VM. Figure 19 shows the vSphere host read and write rates on the Fibre Channel host bus adapters (HBAs) used for the deployment operation. The first clone was created with XCOPY disabled and the host generated traffic across the adapter. The second clone was created with XCOPY enabled and the host traffic was eliminated, freeing host resources for running VMs.



Performance Chart Legend

Key	Object	Measurement	Rollup	Units	Latest	Maximum	Minimum	Average
■	vmhba2	Write rate	Average	KBps	67	245559	0	29150.339
■	vmhba2	Read rate	Average	KBps	1	245467	0	29143.083

Figure 19. ESXi host adapter bandwidth utilization during VM cloning

The other benefit observed by using XCOPY was a decrease in the amount of time for the tasks to complete. On average, the clone tasks completed 14% faster with XCOPY enabled. Figure 20 shows the average duration for three vSphere cloning tasks.

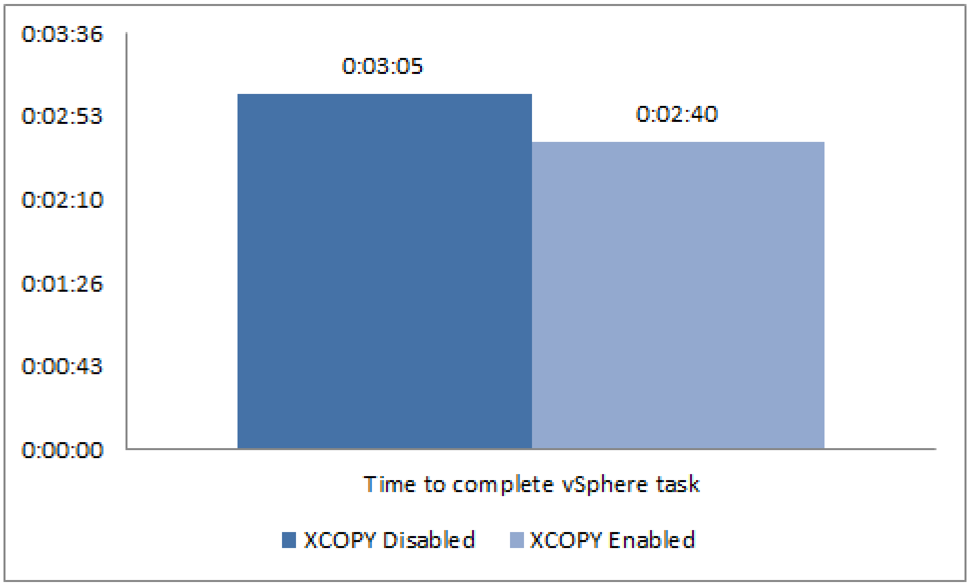


Figure 20. Average duration to complete vSphere template deployment task

Test case 2: Running a Storage vMotion task

Test case 2 focused on measuring the impact of XCOPY on a Storage vMotion task. Storage vMotion is the nondisruptive migration of a running VM between two VMFS data stores. Instead of data being copied, as in the cloning test case, Storage vMotion moves the data. The procedure included four different testing scenarios:

- Storage vMotion with XCOPY disabled
- Storage vMotion with XCOPY enabled
- Storage vMotion with XCOPY disabled and I/O load placed on the host's HBA
- Storage vMotion with XCOPY enabled and I/O load placed on the host's HBA

With no load applied to the HBAs of the ESXi host, enabling XCOPY resulted in a 13% decrease in the time to complete the Storage vMotion task. This is shown in Figure 21. When competing load was applied to the HBA of the ESXi host, enabling XCOPY resulted in a 44% decrease in the time to complete the Storage vMotion task. This can be seen in Figure 22.

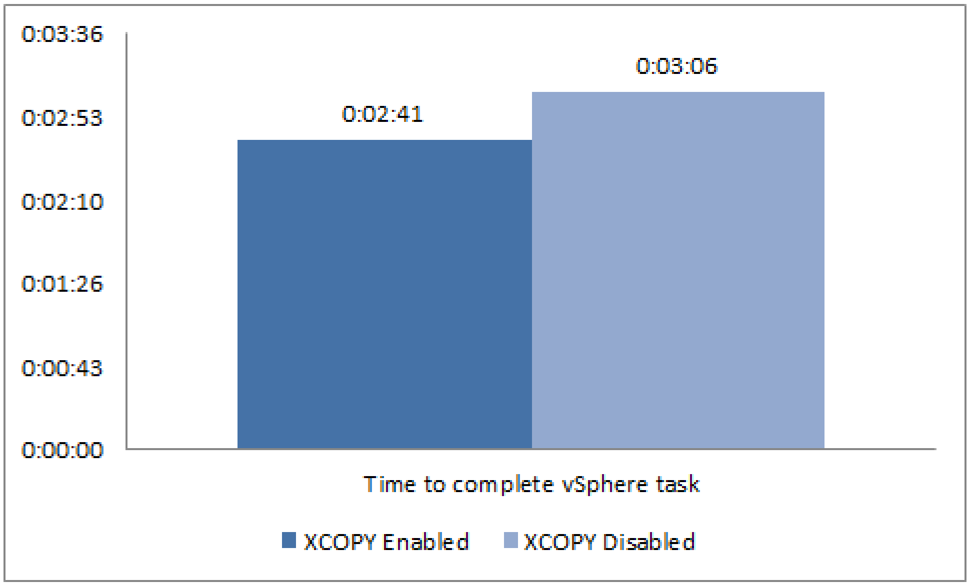


Figure 21. Duration of time required for Storage vMotion task to complete

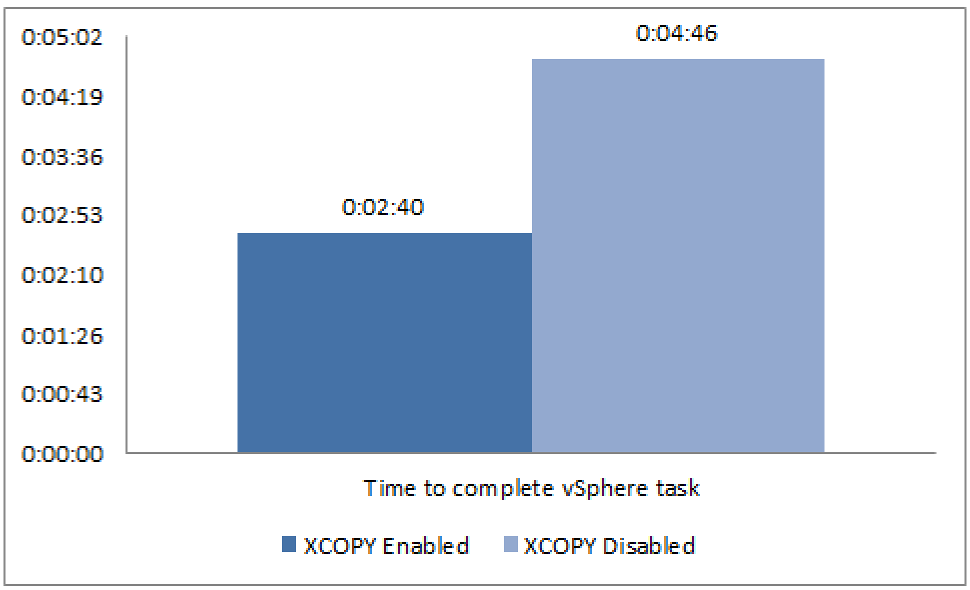


Figure 22. Duration of time required for Storage vMotion task to complete with a competing workload.

Write Same (Zero) test cases

Write Same (Zero) increases performance in several scenarios. The test cases in this white paper focus on the scenarios of creating a VM with a *zeroedthick* disk and the performance increases gained by doing so. Tests were run first with Write Same (Zero) disabled and then again with the primitive enabled. The results between the tests were compared and are as follows.

Test case 1: Creating a VM with an eagerzeroedthick disk

For test case 1, a new VM was created with a 100 GB disk. During the VM creation process, the **Thick Provision Eager Zeroed** option was selected for the 100 GB disk. This option ensures that the disk created was *eagerzeroedthick*. The VM was first created with the Write Same primitive disabled and then again with the Write Same primitive enabled. The tasks to create were monitored and then compared.

The benefit of Write Same (Zero) was the offloading of the repetitive write operations from the vSphere host to the storage subsystem. By offloading the write operations, the storage interfaces of both the host and storage system are not saturated with write operations. This leaves both interfaces free to handle requests from VMs. Figure 23 shows the storage system interface activity during the VM creation process. Note the high bandwidth utilization. Figure 24 shows the storage system interface activity during the VM creation process but this time with Write Same (Zero) enabled. Note that there is no activity on the storage system interface, but there is still write activity on the volumes as the storage system performs the requested task.

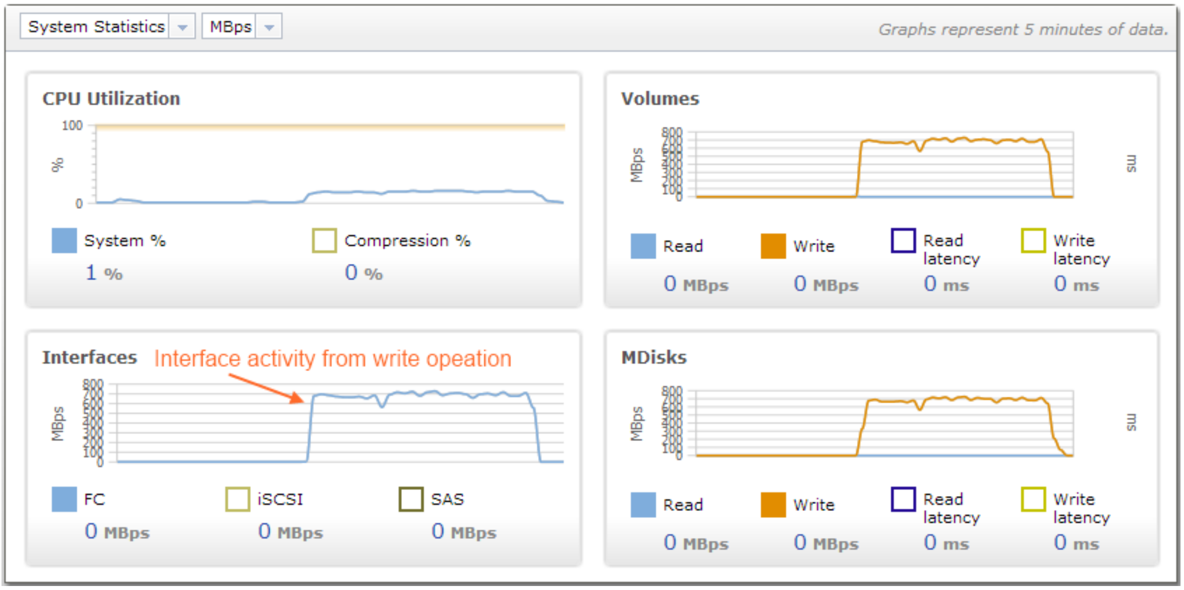


Figure 23. Storage interface activity during the VM creation with Write Same (Zero) disabled

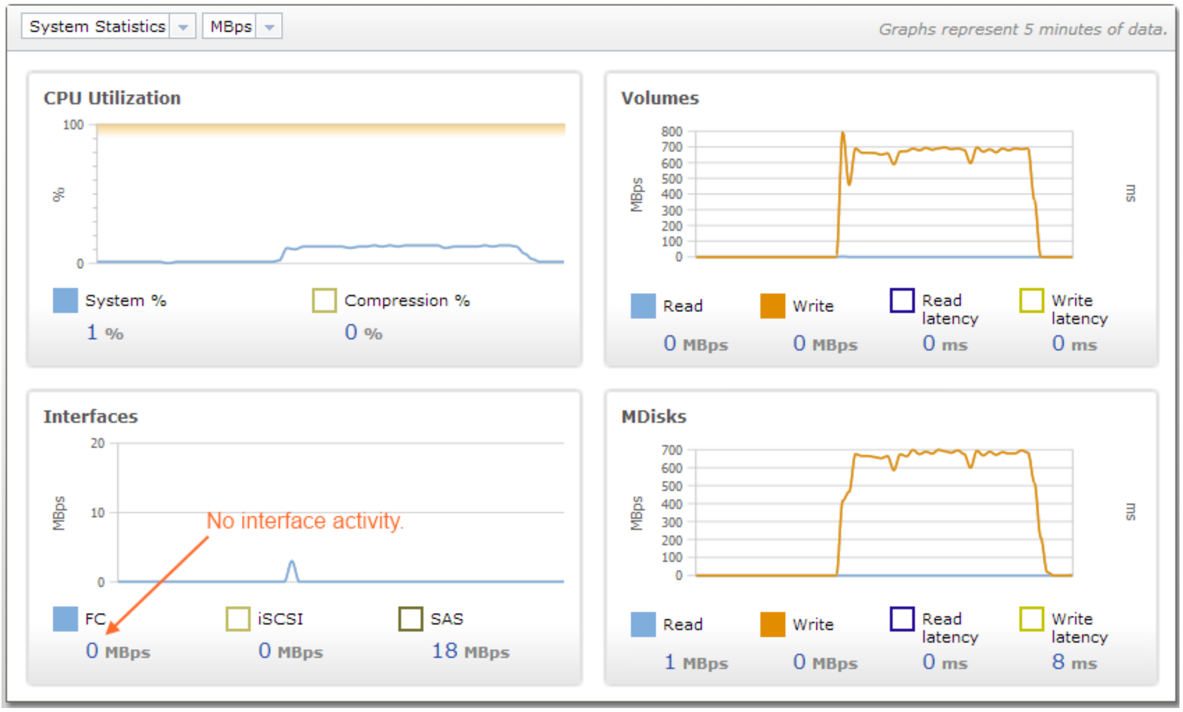


Figure 24. Interface activity during the VM creation with Write Same (Zero) enabled

Test case 2: Writing data to a zeroedthick disk

In test case 2, a VM was set up with the Iometer workload tool. A new VMDK file was created and attached to the VM using the default parameters. The default disk type for vSphere is zeroedthick. With this disk type, disk format capacity is reserved on the VMFS data store, but data blocks for the virtual disk are not formatted until data is written to them. This typically means that throughput to unwritten blocks on the *zeroedthick* disk is impacted until data has been written. After adding the VMDK file, an Iometer workload profile was configured to issue 1 MB sequential writes to the new volume. The process was first run with Write Same (Zero) enabled and then again with the primitive disabled.

Write Same (Zero) enhances the thoughput performance of *zeroedthick* disks by offloading the block format task from the vSphere host to the storage system. In this test case, a throughput increase of 100% was observed when data was continually written to a VMDK file with Write Same (Zero) enabled. The performance numbers from the test can be seen in Figure 25. Every write in the workload is sent to an unused block requiring the block to be formatted before being written to. This is the worst-case scenario for zeroedthick disks.

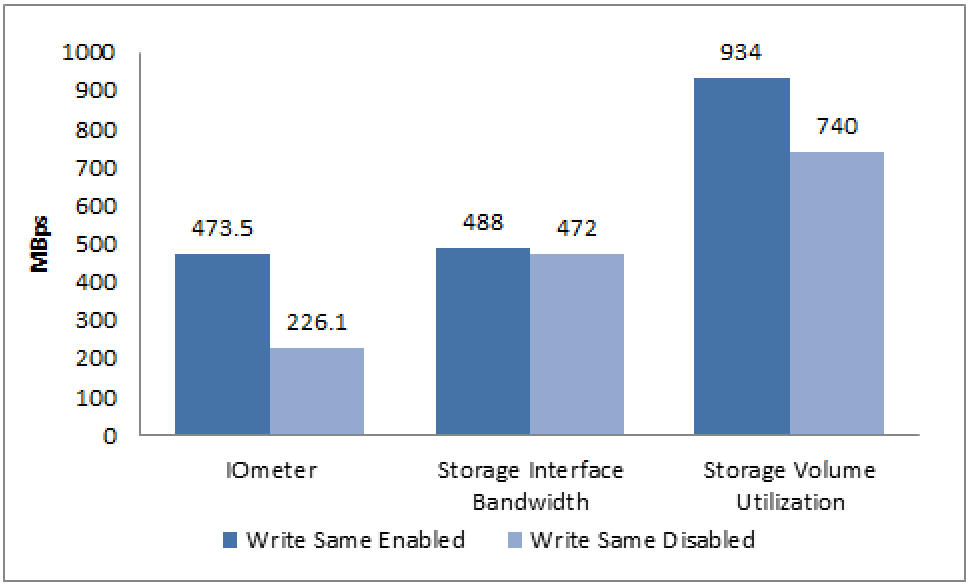


Figure 25. Bandwidth utilization during an active Iometer workload

A secondary benefit discovered is the reduction in wasted interface bandwidth on both, the vSphere host and storage system. Figure 25 shows the interface bandwidth utilization with Write Same (Zero) enabled and disabled. With Write Same (Zero) disabled, the Iometer workload is generating 226.1 MBps of throughput while the interface is reporting 742 MBps of throughput. With Write Same (Zero) enabled, the Iometer workload generated 473.5 MBps of throughput and the interface reported 488 MBps. This difference can be accounted for by the need to send a format command before a write command when Write Same (Zero) is disabled. The format command uses interface bandwidth that can be used for VM workloads.

Summary

The no-charge addition of VMware vSphere Storage API for Array Integration to the Storwize family enhances the value of the system for VMware environments. The vSphere Storage API for Array Integration primitives, such as Write Same (Zero) and Extended Copy (XCOPY) reduce the vSphere host resource consumption otherwise needed for common VMware vSphere tasks, and the Atomic Test and Set (ATS) primitive aids in large storage scaling and enhanced performance. The addition of the support for the API to the Storwize family increases the overall system performance and efficiency for VMware vSphere environments.

Resources

The following websites provide useful references to supplement the information contained in this paper:

- IBM Systems on PartnerWorld
  ibm.com/partnerworld/systems
- IBM Redbooks
  ibm.com/redbooks
- IBM System Storage SAN Volume Controller
  ibm.com/systems/storage/software/virtualization/svc/
- IBM Storwize V7000
  ibm.com/storage/storwizev7000
- IBM Storwize V3700
  ibm.com/systems/storage/disk/storwize_v3700/
- IBM TechDocs Library
  ibm.com/support/techdocs/atsmastr.nsf/Web/TechDocs
- VMware Documentation Center
  vmware.com/support/pubs/

Trademarks and special notices

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.