



# View Storage Accelerator in VMware® View™ 5.1

Leveraging the Content-Based Read Cache in  
VMware vSphere™ 5.0 within VMware View 5.1

TECHNICAL WHITE PAPER

## Table of Contents

Introduction .....	3
About VMware View .....	3
About VMware View Storage Accelerator .....	3
Advantages .....	3
Limitations .....	4
Content-Based Read Cache .....	5
CBRC Design .....	5
CBRC Filter .....	5
Offline Hashing .....	5
Digest File .....	6
Online Caching .....	6
Digest Recompute .....	7
Digest Journaling .....	7
CBRC Implementation .....	8
CBRC Work Flow .....	9
Read I/O Flow .....	10
Write I/O Flow .....	10
CBRC Parameters in vSphere .....	11
Advanced Digest Settings .....	12
View Storage Accelerator .....	13
Configuring Host Caching .....	13
vCenter Server Settings .....	13
Configuring Desktop Pool to Use Host Caching .....	14
Use Host Caching .....	15
Disk Types .....	15
Regenerate Cache .....	15
Blackout Times .....	15
Virtual Machine Host Caching Status .....	16
Advanced Host Caching Configuration .....	16
LDAP vCenter Attributes for Host Caching .....	16
LDAP Attributes for Desktop Pool Host Caching Configuration .....	17
Virtual Machine Host Caching LDAP Attributes .....	18
Useful Tips .....	19
Conclusion .....	20
References .....	20
About the Author .....	20
Acknowledgements .....	20

# Introduction

## About VMware View

VMware® View™ is the pioneer solution in desktop consolidation. VMware View virtualizes and consolidates desktops to a datacenter, eliminating the physical barriers that might be caused by personal desktop hardware. VMware View offers a rich, native personal desktop experience by accessing virtual desktops through a client-server computing model. The new VMware View 5.1 has an improved set of features and a more reliable foundation, supporting VMware vSphere™ 5.0 as the back-end infrastructure. This enables virtual desktop infrastructure (VDI) deployments that are capable of providing a robust platform for enterprises hosting virtual desktops from VMware vSphere.

## About VMware View Storage Accelerator

The goal of this paper is to provide a thorough understanding of how the content-based read cache (CBRC) functions with vSphere 5.0, and how it is integrated in VMware View 5.1. It will also provide a detailed explanation of configuring View Storage Accelerator (also known as Host Caching) for View desktops.

VDI deployments can scale up by increasing back-end computing power and by adding storage space for desktop consolidation. Storage space preserves virtual machine images, which load to the host memory to make the desktops live. A typical performance bottleneck is the I/O request issued from a virtual machine to the underlying storage for accessing the data contents from virtual machine images. View Storage Accelerator included in VMware View 5.1 addresses the bottleneck by leveraging the CBRC feature in vSphere. Though CBRC is a vSphere feature, it is uniquely leveraged by VMware View to provide the Host Caching capability. The CBRC feature provides a per-host RAM-based solution for View desktops. This considerably reduces the read I/O requests that are issued to the storage layer, and also addresses boot storm snags.

Studies show that View Storage accelerator helps to minimize the overall Total Cost of Ownership (TCO) in VMware View deployments by providing up to 80 percent reduction in peak IOPS and up to 65 percent reduction in peak throughput. Refer to the [VMware End-User Computing Blog](#) for more details.

### Advantages

1. View Storage Accelerator offers great performance improvement during boot storms when large numbers of cloned virtual machines are powered on.
2. View Storage Accelerator is highly beneficial when disks contain the same content irrespective of where it came from. In the case of View desktops, there is a greater chance of the same content appearing across disks. This can also happen with shared disks in View Composer OS disks, or for full clone pools that are cloned from the same master template.
3. View Storage Accelerator is also beneficial when users load applications like Microsoft Outlook, Adobe Reader, and other applications or data frequently. In each of these cases, read requests for identical content are served directly from the shared cache memory.
4. View Storage Accelerator is desirable when a boot storm is highly read-intensive and multiple virtual machines issue read requests for data blocks that have identical content.
5. View Storage Accelerator is supported for any vCenter-Managed View Desktops such as manual desktops, automated full clone desktops, and automated linked clone desktops.
6. Configuration of View Storage Accelerator can be easily done through View Administrator Console, and is a one-time configuration for a pool.
7. Maintaining View Storage Accelerator is quite easy by the use of automated settings for appropriate digest regeneration intervals and blackout periods.

8. In case of desktops where a persistent disk contains user data, Administrator can configure View Storage Accelerator for both OS and persistent disks.
9. CBRC configuration is applied to vSphere hosts in a dynamic way based on the possible placement of host caching-enabled View Desktops. This ensures no unnecessary cache memory reservation on hosts that will never run cached virtual machines.

### **Limitations**

1. Virtual machine host caching is applied only when the virtual machine is powered off. Configuring host caching for a powered-on virtual machine requires virtual machine shutdown to apply the configuration. In the case of linked clone virtual machines, caching cannot be enabled when any of the virtual machines based on the same shared base disk are powered on.
2. Host caching uses the host's RAM, which reduces the virtual machine consolidation ratio. A larger cache size leads to a lower consolidation ratio.
3. Enabling host caching will create additional disks for each boot VMDK and snapshot VMDK, which results in increased storage utilization.
4. If the administrator changes the advanced configuration parameters, some changes might require the vSphere CBRC module to be unloaded and reloaded for the changes to take effect.
5. Host caching cannot be configured for non-vCenter managed pools, or terminal server pools.

## Content-Based Read Cache

Before discussing the integration aspects of VMware View 5.1, a detailed explanation of CBRC design and work flow in vSphere 5.0 will help to provide a basic understanding of this core feature.

### CBRC Design

VMware introduced CBRC in VMware vSphere 5.0. Backend IOPS reduction, the key benefit of this feature, is achieved by a solution that comprises two stages, Offline Hashing and Online Caching. The core module of the feature, the CBRC VSCSI Filter (which is also referred to as the CBRC Filter) connects to a virtual machine disk file (VMDK), which provides the interface for hashing and caching. As Figure 1 illustrates, the CBRC Filter uses a RAM-based cache to manage the cached disk blocks.

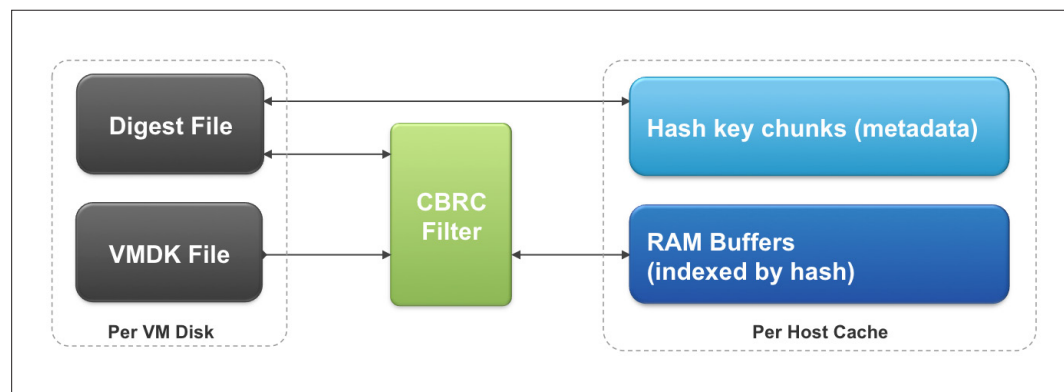


Figure 1: CBRC Architecture

### CBRC Filter

The core of the CBRC implementation is a VSCSI Filter, as shown in Figure 2. The CBRC filter is transparently attached to the CBRC-enabled VMDK. This maintains a global RAM cache to serve the I/O requests from the CBRC-enabled virtual machine. The data stored in the cache is based on a hash signature and is not tied to a particular virtual machine. This provides an easier way to detect duplicate blocks across virtual machines that are served from the cache.

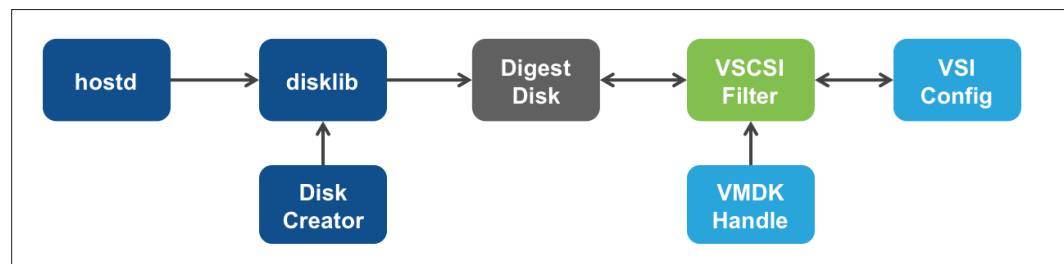


Figure 2: CBRC Filter Layout

### Offline Hashing

Offline hashing generates digests of the VMDK stored in the vSphere datastore. Each boot VMDK will have associated digests that represent hash values of the content blocks. This digest information contains associated digest headers, journaling, and digest hashes. The journaling is used for digest recovery and its hash values are used for online caching.

### Digest File

During offline hashing, the application creates a VMDK file for each boot VMDK. Each one will have an associated `-digest.vmdk` file representing the hash values of the block. This digest disk file contains the metadata and set of hashes. While performing offline hashing, the application examines the entire boot VMDK and generates hashes for each content block. The estimated size of the digest file is 0.5% of the logical disk size in the case of an SHA1 hash and 4K block size—when collision detection is disabled. The estimate is 1.2% of logical disk size in case of primary and secondary hash—when collision detection is enabled. (Collision detection is explained in the [Advanced Digest Settings](#) section of this document.) By default, the digest file for a 20GB Windows virtual machine will consume about 100MB of storage space.

When contents of the boot VMDK are modified, the corresponding hash value inside the digest is marked as invalid. As shown in Figure 3, this invalidation is performed by changing the corresponding hash bitmap value from 1 to 0.

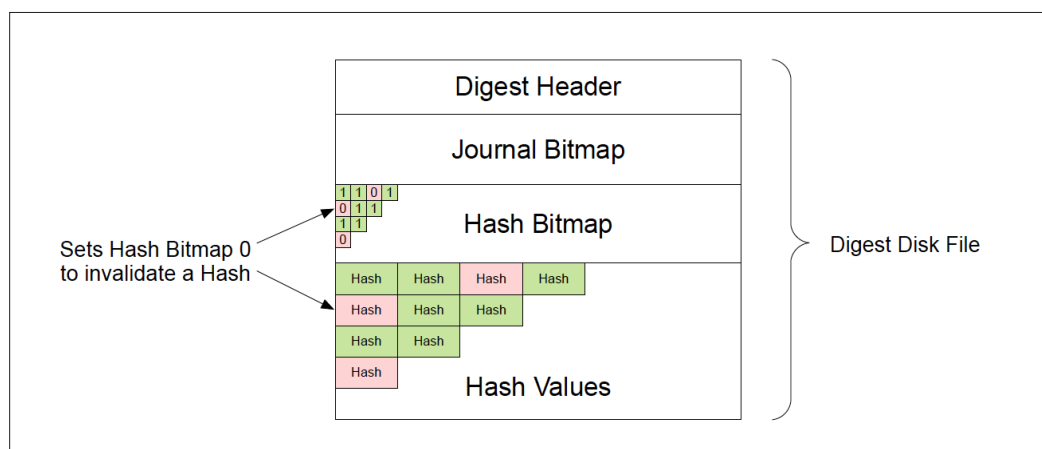


Figure 3: Digest Disk Layout

If the Virtual machine contains a snapshot, then a separate digest file is created for the snapshot VMDK. If a VMDK is cloned, the digest file is also copied. In case of linked clone virtual machines, the digest file of the base disk is shared between virtual machine instances in the same way that the base disk is shared. A separate digest file is created for each linked clone virtual machine, corresponding to each respective redo log (delta file). When changes to each linked clone virtual machine update to the redo log, the digest file corresponding to the redo log also gets updated, with hash invalidations and recomputed hashes for new content. In this case common blocks are shared across virtual machines from the single digest, and content changes are tracked individually through the delta disk digest file.

### Online Caching

When a virtual machine powers on, the metadata of the computed digest disk is loaded to reserved RAM space by VMware ESX® 5.0. This contains the valid block-hash mapping from the digest. When a virtual machine issues a read I/O request, it checks the digest's in-memory copy for a valid hash of the requested block. If the requested block is valid it is directly served from the RAM buffer cache, which eliminates read I/Os to back-end storage. If the requested block does not exist in the RAM buffer cache, then the back-end storage is accessed and the block is copied to the RAM buffer. The requested block is then served to the virtual machine. If the hash value of the requested block is marked as invalid, then content is served directly from the back-end storage layer.

### Digest Recompute

In case of a read request from a virtual machine, the cached content can be served from the RAM buffer by checking for its valid hash from the digest's in-memory copy. But any write I/O request must update the original disk, which invalidates specific block-hash references in the digest. These written blocks from the specific virtual machine cannot serve from the RAM buffer cache. During any such write IOPS, the in-memory copy of the digest is updated with the hash invalidation and passed down to the digest file in storage. At the end it is flushed to the digest disk file during virtual machine shutdown. This results in a certain percentage of invalid digest section increases on virtual machine usage. Because invalid hash blocks are not considered for caching, if the majority of the hashes are invalid there is not much to serve from the cache buffer. It is important for the digest file to be regenerated so that most of the hash-block references are valid. This is referred to as digest recompute or cache regenerate.

### Digest Journaling

When a write request is issued by a virtual machine, the following updates are required to invalidate the digest.

1. Update the digest in-memory copy with hash invalidations
2. Update the original disk with new data
3. Update the digest disk file with hash invalidation and new hash computation for new content

Update 2 happens immediately after update 1. The application updates the in-memory copy first, and then updates the new data block in the original disk. The third operation occurs separately, as the update needs to be performed from the in-memory digest to the stored digest disk file. The new hash needs to be generated, which takes additional computing power.

Sometimes there are chances that the hash invalidations marked in the in-memory copy of the digest are not completely updated in the storage digest disk file. This occurs when the virtual machine powers off immediately, or if there is a host crash. In that case, the third update operation does not happen and the digest file does not update the hash invalidations and new hash computations. This leads to corrupted digest files.

To overcome this, the application uses a journaling header in the digest file. Before marking the hash as invalid (to the in-memory digest) whenever the write happens, the application tags the journal area that the particular block is updated in the original disk. Journaling receives the information regarding the updated blocks. Journal header hash invalidations can be updated in the digest file and a new hash can be computed for the updated block. This operation is called disk journaling.

When a virtual machine performs a graceful shutdown, the journal area is zeroed and valid hashes are written back to the digest disk file. Invalidated hashes from the digest memory copy are discarded. When the digest loads to memory next time, it first looks for the journal area for any pending update. In case of a clean shutdown, the journal changes to zero and all digest files are treated as updated. Otherwise it identifies the journal headers, which leads to a complete digest file update. This is called digest recovery.

The disk journaling operation is costly in terms of computing power and is temporarily disabled during virtual machine boot time. This delay is referred to as digest journal-boot interval. This interval is 10 minutes by default, allowing the virtual machine to complete the boot process and reach steady-state operation.

If a virtual machine with digest disk is powered on in a CBRC-disabled host, apart from losing the benefit of caching, this also results in updating the original VMDK, which leaves the digest file as it was. The journal header cannot track any changes being made to the original disk, and the digest file becomes invalid. When the virtual machine next powers on in a CBRC-enabled host, the digest file is marked as invalid and it recomputes the digest automatically when possible.

## CBRC Implementation

Figure 4 is a logical diagram that explains how storage read I/O is eliminated and boot storm is reduced using the CBRC implementation in vSphere 5.0. In the diagram there are four virtual machines cloned from a template. CBRC is enabled for these virtual machines. Figure 4 also describes Offline Hashing and Online Caching of the boot VMDK for one out of the four virtual machines. The explanation is based on the assumption that all virtual machines are cloned from the same master. In this case there will be many duplicate read I/O requests placed from each virtual machine for the same data content, especially during the boot process

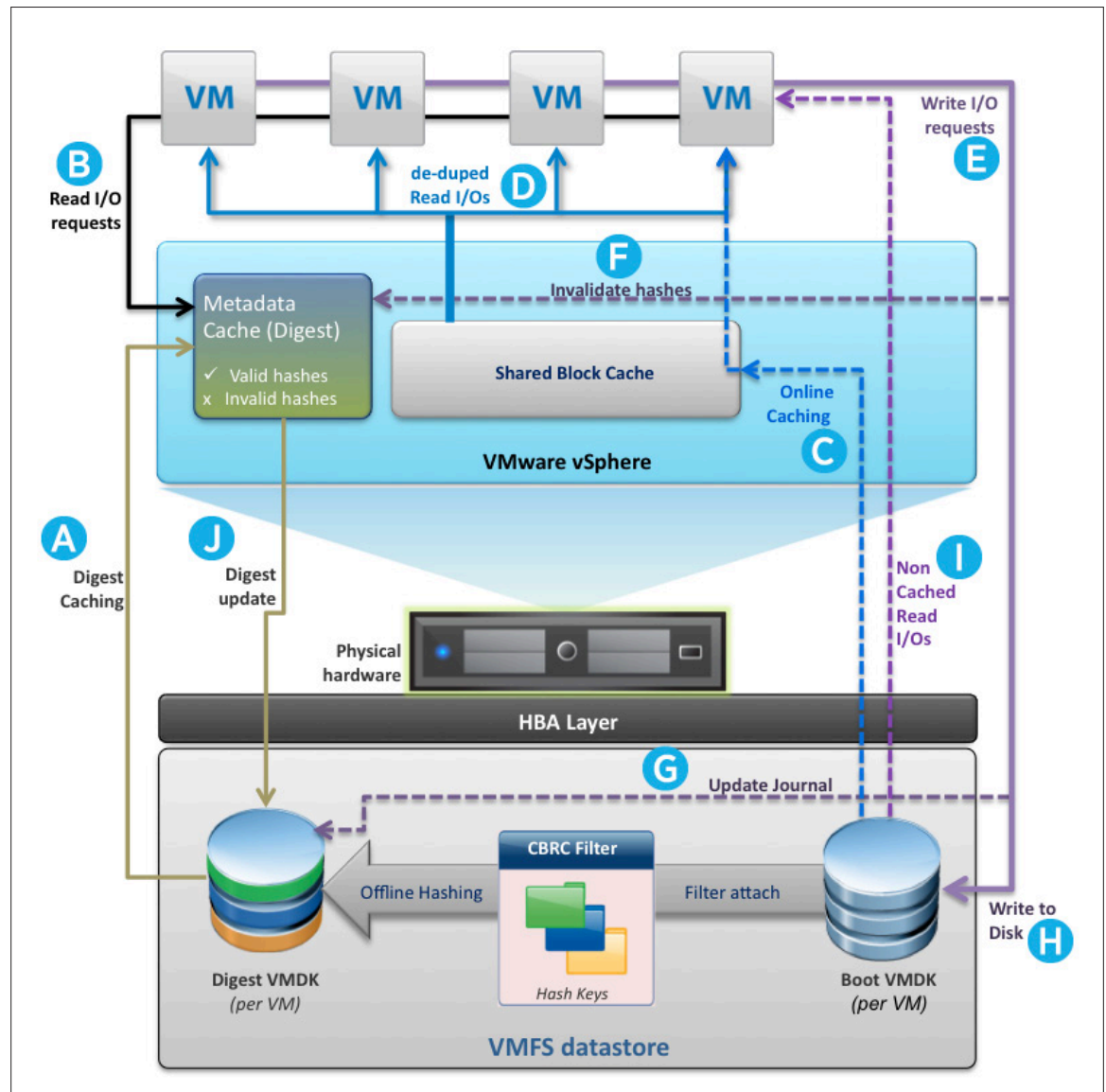


Figure 4: Storage I/O Storm Reduction Using CBRC Implementation in vSphere 5.0



**CBRC Work Flow**

Figure 4 identifies the key CBRC operations, which are marked with a letter. These operations are performed during the workflow scenarios below.

***Initial Configuration (Operation: A)*****A.**

When CBRC is configured on a vSphere host and enabled for a virtual machine, the CBRC filter module is attached to the boot VMDK. The CBRC filter module creates a digest file for the boot VMDK by computing the hash values of every 4K block. When a virtual machine powers on in a CBRC-enabled host, the virtual machine kernel module receives information about the location of the digest file, and updates the VMDK metadata cache (called 'digest in-memory copy' or 'digest cache') based on the hash contents.

***Virtual Machine Issues Read Requests (Operations: B, C, and D)*****B.**

When the first virtual machine is powered on, it starts issuing read requests to the storage layer. Now, since CBRC is enabled, it checks the metadata cache (digest in-memory copy) for a valid hash of the requested block.

**C.**

If the hash is not found, the read request is sent to storage and the block is cached to RAM buffer (shared block cache).

**D.**

It is then served to the virtual machine buffer.

When the next virtual machine issues a read request for the same content, operation B is repeated, which finds a valid hash. If a valid hash is available, the corresponding block from the shared block cache (RAM buffer) is served to the virtual machine buffer. Operation C is not performed in this case and I/O to storage layer is eliminated.

***Virtual Machine Issues Write Requests (Operations: E, F, G, and H)*****E.**

When the virtual machine issues a write request, it first updates the digest memory copy (metadata cache) for that particular virtual machine.

**F.**

It marks the corresponding hash of the block as invalid.

**G.**

Then the journal header is updated for the block that is going to be modified.

**H.**

It immediately writes the block to the original disk.

***Non-Cached Read Requests (Operation: I)*****I.**

The virtual machine issues a read request for any such written content (explained in previous section through E, F, G, and H), that is served directly from storage in a conventional method.

***Digest Updates (Operation: J)*****J.**

The digest file is updated from the metadata cache in specific intervals. Finally, when the virtual machine shuts down, the complete valid digest is flushed to the disk and zeros the journal area.

***Recompute***

Over a period of time, when the digest reaches a stage where most of the contents are invalidated, a recompute is triggered. During recompute the digest disk content is regenerated, replacing the invalid content with the latest data. Like digest creation, recompute is also performed when the virtual machine is powered off.

### Read I/O Flow

When the virtual machine first powers on, the CBRC filter reads the per-VMDK digest file and updates a global hash table. This in-memory copy of the digest maintains the logical block number (LBN) to hash mapping of the VMDK, and also helps identify de-duplication blocks across VMDKs.

When a read is issued by the virtual machine, the hash value for the block is identified. If the data is available in the RAM cache, it then copies that data directly from the cache. If blocks are not present in the cache, it then issues a smallest possible I/O request to the back end that will fill the cache buffer. This buffer-fill data is copied to the virtual machine's data buffer.

Figure 5 explains the Read I/O flow.

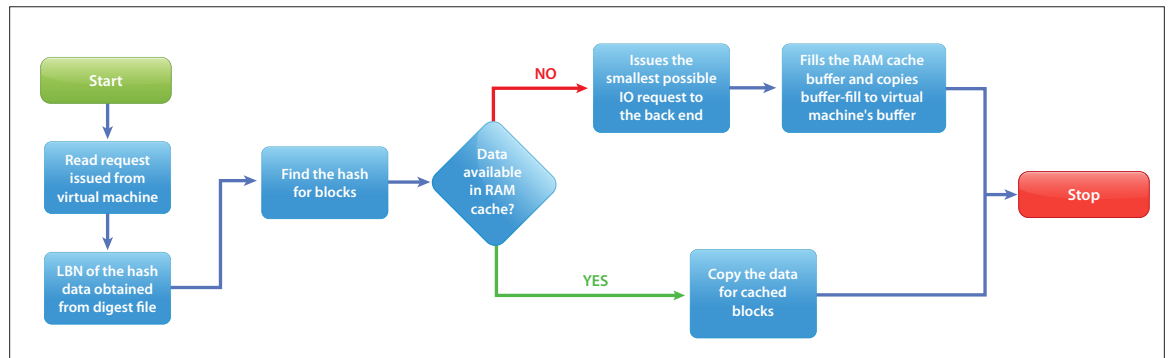


Figure 5: The Read-I/O Flow When CBRC Is Used

### Write I/O Flow

When a 'write request' is issued by the virtual machine, the digest metadata that includes the hash information for that block is updated as no longer valid. It then performs a 'Journal' of the write operation for digest recovery and sends the write request directly to the underlying storage back end.

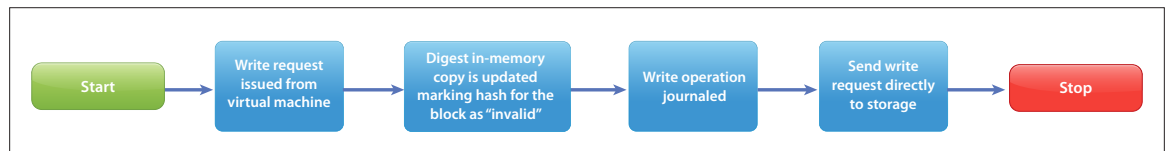


Figure 6: The Write-I/O Flow When CBRC Is Used

## CBRC Parameters in vSphere

In vSphere there are many parameters and settings required for CBRC. These are modified dynamically based on pre-defined policies and configurations through VMware View host caching settings. Direct modification of these values is not recommended. In this document these parameters are mentioned for a basic understating and reference purposes.

In vSphere these parameters are visible through VMware VI Client. Select the ESX host from the VMware vCenter™ Inventory and go to the configuration tab. In **Software Settings** select **Advanced Settings** and navigate to **CBRC**, as shown in Figure 7.

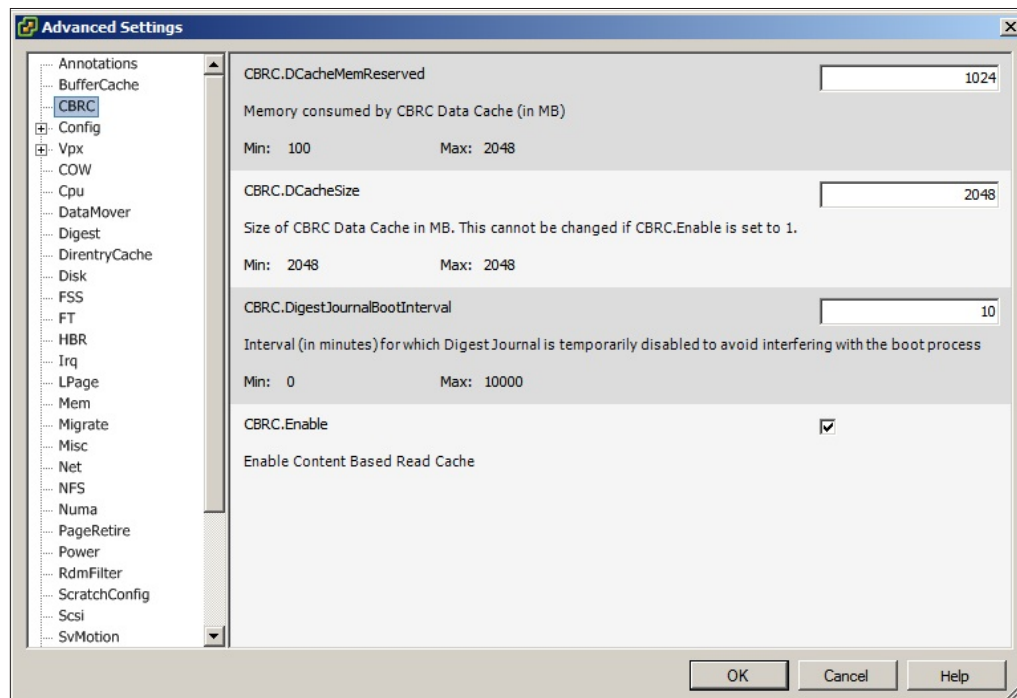


Figure 7: Per-Host CBRC Settings Through VI Client

**CBRC.DCacheMemReserved** – This option defines the amount of host memory in MB that is reserved for caching. This accepts a minimum of 100MB and maximum of 2048MB.

**CBRC.DCacheSize** – This represents the size of the data cache. This option is deprecated and not available for use.

**CBRC.DigestJournalBootInterval** – As stated before, to avoid boot-time impact, journaling needs to be restricted when the virtual machine boots up. To ensure that costly journaling is not performed when the virtual machine powers on, a default delay of 10 minutes is configured. This configuration is not exposed through VMware View.

**CBRC.Enable** – This option decides whether the vSphere host is enabled for serving digests from the reserved cache memory. Manual configuration of this option is not recommended. This is part of the host caching configuration in VMware View (this is explained in the [LDAP vCenter Attributes for Host Caching](#) section of this document).

### Advanced Digest Settings

VMware vSphere exposes advanced digest configuration information. On vSphere the parameters exposed are visible through VMware VI Client. Select the ESX host from the vCenter Inventory and go to the configuration tab. In **Software Settings** select **Advanced Settings** and navigate to **Digest**, as shown in Figure 8. These values are updated based on policies set through LDAP configuration in VMware View.

**Note:** Configuration is performed dynamically based on the internal policy set by the View Administrator. These options are given for reference purposes only and modification is not recommended.

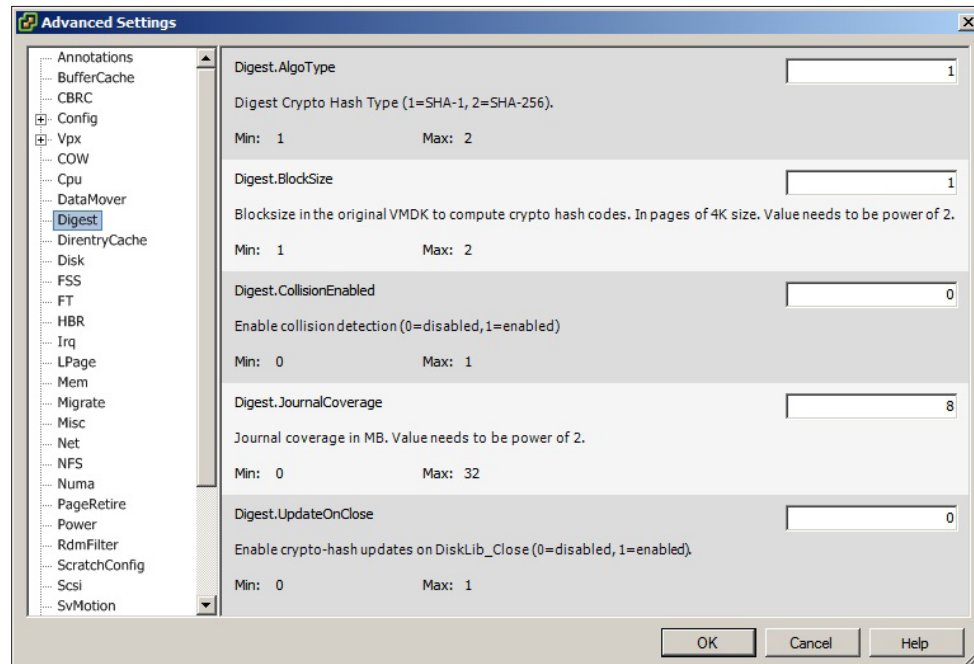


Figure 8: Per-Host Advanced Digest Settings Through VI Client

**Digest.AlgoType** – This defines the crypto algorithm type that is used for generating the digest hash. For SHA-1 the value is 1 and for SHA-256 the value is 2. This is a policy-based dynamic setting to detect and avoid hash collisions.

**Digest.BlockSize** – Digest metadata is created in a 4K block size by default. This setting defines the block size in original VMDK to compute crypto hash codes. The value needs to be set in the power of 2.

**Digest.CollisionEnabled** – The digest file contains a sequence of SHA1 hashes (one hash per 4K of VMDK block). This option provides a hash-collision detection (strict compare) operation if the SHA1 hashes are not deemed sufficient. If this option is enabled, a secondary hash is computed and stored in the digest file (SHA256). The two hashes will be used by the CBRC filter module to detect hash collisions. Blocks for which hash collisions are detected are not cached. The size of the digest file and the kernel memory footprint will increase if this option is enabled. By default, this is turned off.

[0=disabled, 1=enabled]

**Digest.JournalCoverage** – This defines the size of journal space in MB. In case of a host crash, the digests need to be reconstructed with minimum loss of invalid data block information. Journal space is used to store this information. This value needs to be in the power of 2, and the maximum is 32.

**Digest.UpdateOnClose** – This parameter decides when to update crypto-hashes. Any disklib write operation will update the metadata in the digest file with the updated blocks. If this option is enabled the digest hashes are recomputed after any disklib close operation. Turning this option on recomputes the digest transparently.

[0=disabled, 1=enabled]

# View Storage Accelerator

This paper has discussed how CBRC functions at the vSphere level and demonstrated its work flow. It further discusses configuring and managing View Storage Accelerator (host caching) in VMware View.

VMware View uniquely leverages CBRC in vSphere and provides host caching support to View desktops. It must be explicitly enabled and configured through View Administrator. There is no need for manual configuration of CBRC parameters in vSphere; instead the parameters are configured by the host caching options in the View Administrator console.

## Configuring Host Caching

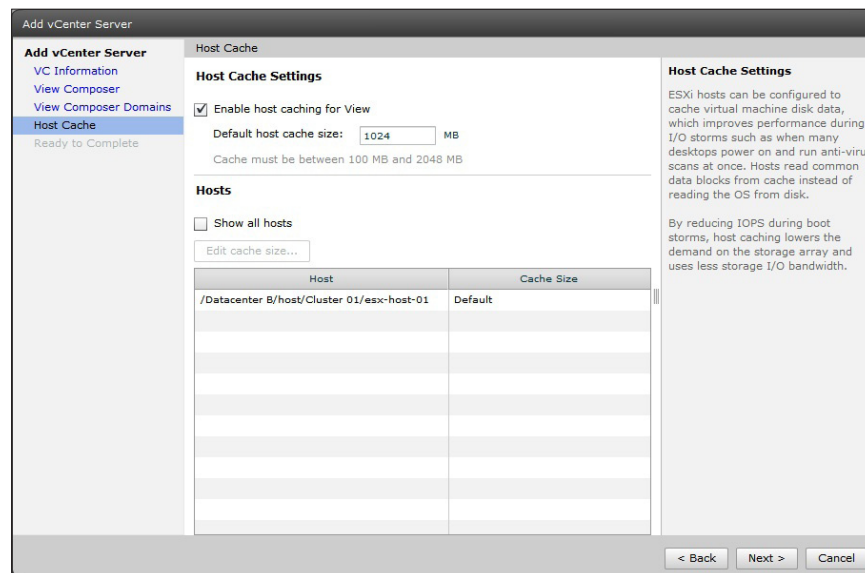
In VMware View 5.1, View Storage Accelerator is configured through Host Cache settings. Host Caching need to be enabled at two levels.

1. vCenter server settings of View Administrator
2. Advanced Storage settings for a desktop pool

The first configuration is part of a general policy about the use of host caching on hosts controlled by a vCenter. It is a quick way to disable host caching for all virtual machines in that vCenter. The second configuration enables the virtual machine to use host caching, when the vCenter is capable of it.

## vCenter Server Settings

Host caching is configured and applied on a per vCenter basis. As shown in Figure 9, navigate to the VMware View Administrator left panel inventory, **Servers**, where the vCenter Server lists all the vCenters added to the View Administrator. Edit the vCenter server and go to the **Host Cache** tab to enable host caching. These options are also available when adding a new vCenter server.



**Figure 9:** Enabling Host Caching for a vCenter Server in View Administrator

When host caching is enabled for a vCenter server through this wizard, all supported vSphere hosts (vSphere 5.0 or later) that are part of the vCenter server are identified as host caching-capable hosts.

All these hosts will be listed under Host Cache Settings. If one specific host's cache configuration needs to be changed, it can be overridden by setting a custom value. Select the specific host and click **Edit cache size...**, and select **Override default host cache size** and input a new cache size value.

The cache value can be from 100MB to 2048MB, as shown in Figure 10. Note that the host cache settings cannot be disabled for a specific host.

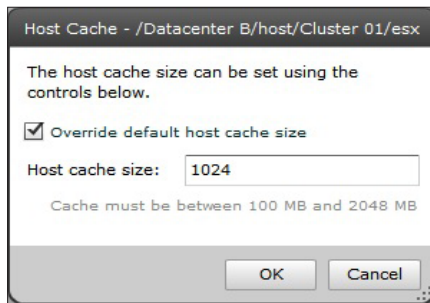


Figure 10: Overriding Default Host Cache Size

These configurations can also be changed for an added vCenter server, by editing vCenter server settings and navigating to the **Host Cache** tab.

**Note:** Though host caching parameters are configured through the View Administrator UI, configuration is not performed on a vSphere host until and unless a host caching-enabled virtual machine uses the host. These settings will reflect in vSphere CBRC settings only if the host is used by at least one virtual machine on a pool that is configured for host caching.

## Configuring Desktop Pool to Use Host Caching

When host caching is enabled at the vCenter level, then any desktop pool that is deployed in the vCenter (except a terminal server pool and physical machine pool) provides the **Use host caching** configuration. While the Pool Create wizard is working, this option can be enabled on the **Advanced Storage Options** page. Figure 11 shows how to enable **Use host caching** with the Automated Linked Clone Pool add wizard.

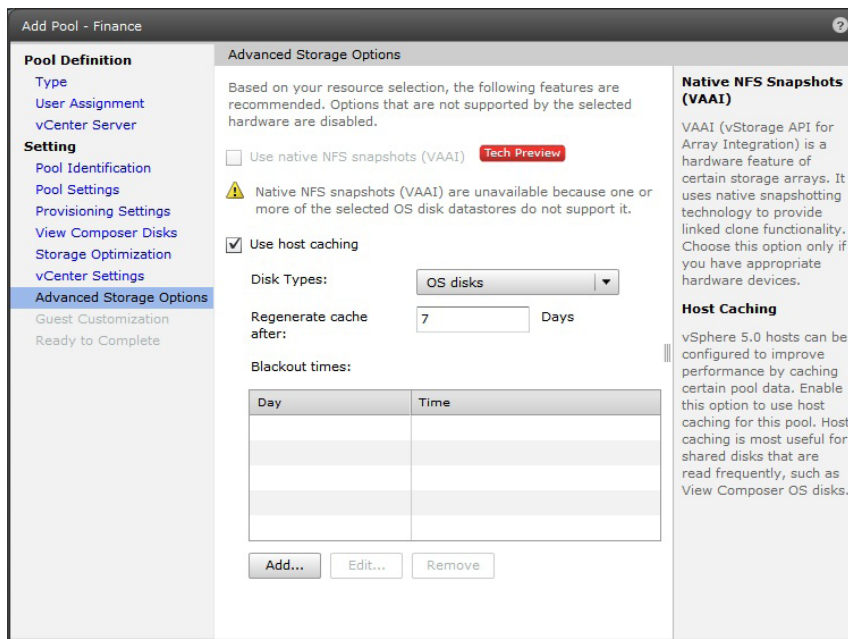


Figure 11: Configuring a New Automated Linked Clone Pool to Use Host Caching

The same configuration can be achieved by editing an existing desktop pool. However, host cache configuration changes of the pool are applied only when the virtual machine is powered off. If host caching is disabled on the vCenter server settings page, the Use host caching option will not be activated for the pool. Pool settings can also be edited to turn off host caching. Each pool setting is explained below. VMware recommends configuring host caching during the pool creation time.

### Use Host Caching

The host caching option enables all the virtual machines in the pool to use host caching configured for the vCenter. If virtual machines are powered on, these settings will not apply immediately and they must wait for the virtual machines to be in the powered-off state. When this configuration is applied to a pool of virtual machines, in the back end at vSphere the CBRC module is attached to the VMDK of the pool-virtual machine. A digest disk is created with metadata and hashes.

When the virtual machine next powers on, the digest is loaded to the metadata cache. If the virtual machine issues a read request, it first checks the metadata cache for a valid block and if found the data block is served directly from the vSphere CBRC cache. If multiple virtual machines issue read requests for similar data content, they are each served from this shared per-host CBRC cache.

### Disk Types

This defines the VMDK disk type of the pool virtual machine that will be used for host caching. This setting is valid only for a linked clone pool with user data redirection. By default only the boot VMDK of the pool virtual machine is selected for digest creation and online caching. This is referred to as 'OS disks' in pool settings. In case of an automated linked clone pool with user data redirection, each pool virtual machine will have an additional persistent disk to store user data. Drop down the **Disk Types** option to choose **OS and persistent disks** so that both the User data disk VMDK and boot VMDK are enabled for host caching.

### Regenerate Cache

Regenerate Cache defines the scheduled interval to perform digest recompute. Write activity to the VMDK can occur during a pool-virtual machine's lifetime, which will result in a large number of invalid hashes in the digest. Data blocks corresponding to these invalid hashes will not be cached. For a continuous benefit, it is important to reduce the number of invalid hashes in the digest by performing digest recompute periodically. The default is 7 days. The Regenerate Cache (digest recompute) operation is less intensive than a create operation because the hashes are computed only from the blocks that have been changed since the digest creation or last regeneration.

### Blackout Times

Blackout Times are those parts of a recurring schedule when cache regeneration should not be performed. This option allows the administrator to plan recompute to run only during non-business hours and off-peak times. As shown in Figure 12, the cache regeneration is not performed from 9 a.m. to 6 p.m. on all weekdays.

The administrator is also allowed to add a list of multiple blackout times or days, and can further modify or remove these list entries. This can be done through the blackout timetable in the host caching wizard, as shown in the table in Figure 10. Administrators can use the **Add**, **Edit** and **Remove** options below this table.

Figure 12: Setting Restriction Period for Cache Regeneration



## Virtual Machine Host Caching Status

The View Administrator Console displays virtual machine host caching status. Navigate to **Inventory-Desktops** and double click a desktop virtual machine for a summary page. Select the vCenter settings tab. Here **General Box Host Caching** represents host caching status. Each status level is explained below.

- **Off** – The host caching status is shown as off if it is not configured for host caching, or if it should not be configured to use host caching.
- **Current** – If host caching is enabled and valid digests are cached, the status shows current. This indicates that the virtual machine is configured to use host caching, and digests are up to date.
- **Out of date** – Because virtual machine lifecycle digests can become invalid over a period of time, the status might display out of date. This means that while the virtual machine is configured to use host caching, the digest needs to be regenerated.
- **Error** – This is displayed only when something goes wrong with virtual machine host caching or a CBRC operation. The Error status message is cleared by the next successful operation.

## Advanced Host Caching Configuration

This section explains all host caching parameters in View Server LDAP. Some of these, marked as optional, can be fine-tuned by the administrator. **Caution:** VMware strongly recommends that users consult with VMware Support before making changes to these parameters.

### LDAP vCenter Attributes for Host Caching

From the View Connection Server OS console, as shown in Figure 13, browse **LDAP** through **ADSI Edit**. Navigate to **OU=Properties – OU=VirtualCenter**. vCenter entries are listed here. Open **Properties**.

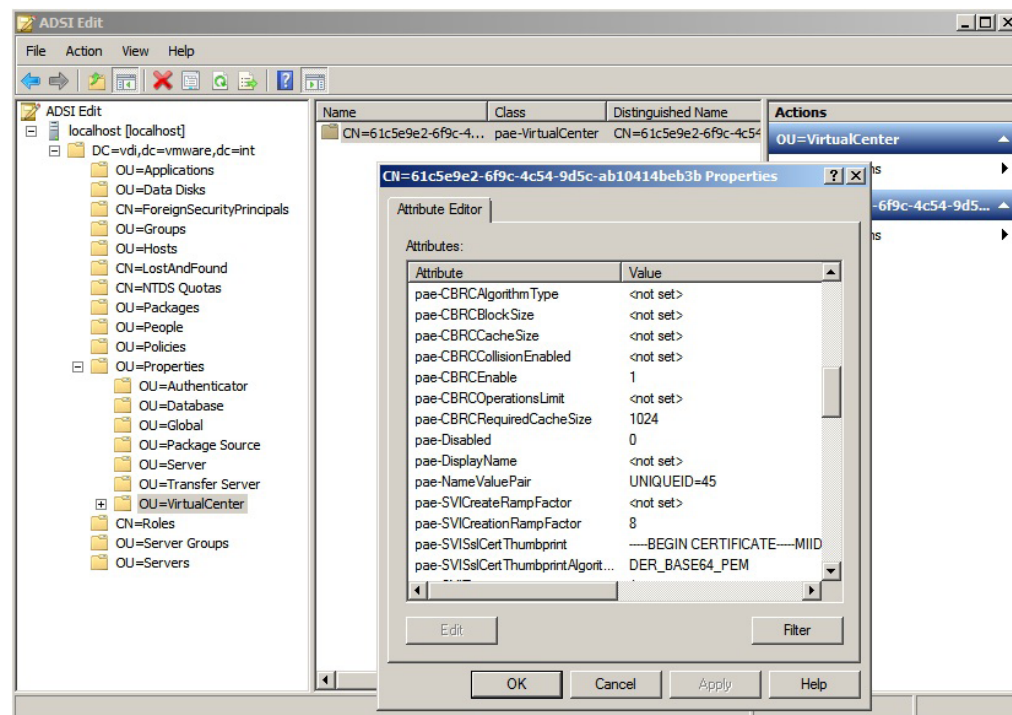


Figure 13: LDAP CBRC Settings for vCenter



- **pae-CBRCAAlgorithmType** – This sets the algorithm for creating digest hashes. Options are 1 for SHA-1 and 2 for SHA-256. The default is 1 (SHA-1). This configuration is optional. This controls the vSphere advanced parameter named Digest.AlgoType. **Note:** Changing this setting to 2 will have a CPU overhead.
- **pae-CBRCCBlockSize** – This sets the size for the cache blocks to compute from VMDK that are configured to use host caching. Settings are represented in the number for values of 4K pages. This controls the Digest.BlockSize vSphere advanced digest parameter. Options are 1, 2, 4, 8, or 16. The default is 1.
- **pae-CBRCCacheSize** – This is configured with a fixed value of 2048MB in relation to the vSphere CBRC.DCacheSize setting.
- **pae-CBRCCollisionEnabled** – Enable or disable collision detection. Options are TRUE or FALSE. Default is FALSE. This sets the Digest.CollisionEnabled parameter in vSphere advanced digest settings.
- **pae-CBRCEnable** – Enable or disable CBRC on vSphere hosts on vCenter. The default is TRUE. This enables the parameter CBRC.Enable in vSphere configuration.
- **pae-CBRCOperationsLimit** – This defines the maximum number of concurrent CBRC operations. There are two operations related to CBRC that are issued to vCenter from the View Connection Server. These are (1) Configure CBRC for virtual machine; and (2) Regenerate Digest for virtual machine. These operations are governed and controlled by this limit. The default is 50.
- **pae-CBRCRequiredCacheSize** – This represents the memory buffer size required for online caching. The value sets the configuration of host cache size in View Administrator UI. The value accepted is sized in MB, with a minimum of 100MB and maximum of 2048MB. The default is 1024MB. This will drive the CBRC.DCacheMemReserved setting on the vSphere CBRC setting.

#### LDAP Attributes for Desktop Pool Host Caching Configuration

From the View Connection Server OS console, browse **LDAP** through **ADSI Edit**. Navigate to **OU=Server Groups**. Entries for All configured desktop pool are listed here. Navigate to its properties, as shown in Figure 14.

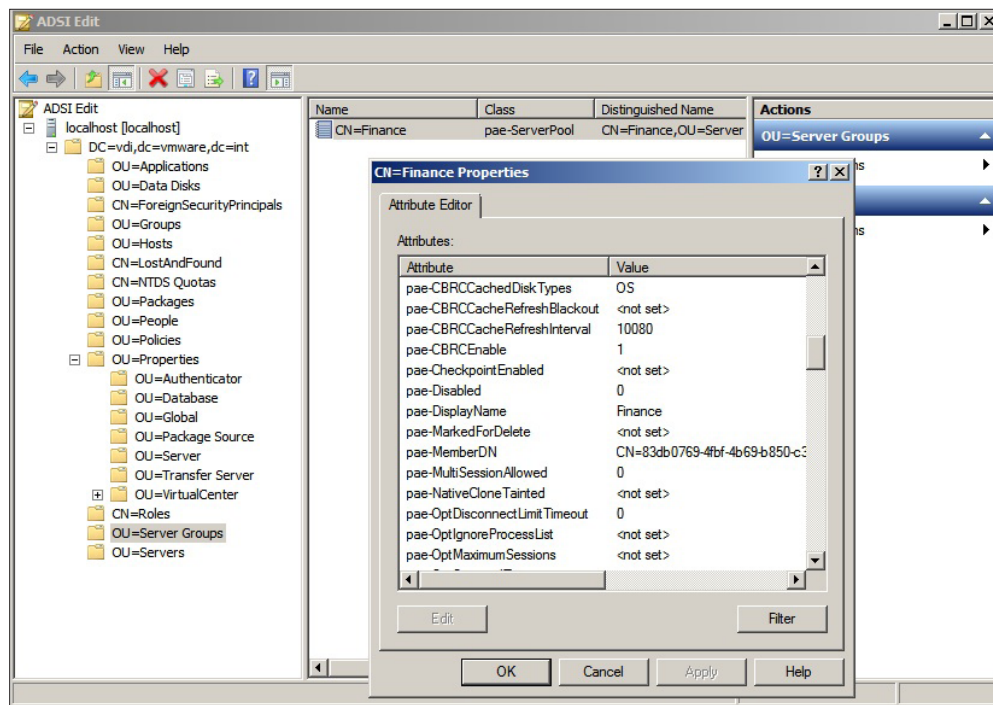


Figure 14: LDAP Host Caching Settings for Desktop Pool

**Note:** Figure 14 highlights the LDAP attributes in View Connection Server against the host caching configuration for a View desktop pool. All values are updated based on desktop pool settings modified through View Administrator, and do not need manual configuration. These settings are explained here for reference purposes only.

- **pae-CBRCCachedDiskTypes** – This represents which disk type is chosen for caching. Values are OS or UDD. OS means OS disks are used for host caching, and UDD means both OS disks and persistent disks are used for host caching. These are applicable when pools with User data re-direction are present. The default is OS.
- **pae-CBRCCacheRefreshBlackout** – Based on the blackout days configuration, this parameter sets two values. The first value is the blackout start time in **HH:mm** (or **u:HH:mm** if a specific day of the week is selected). The second value is the blackout period in minutes.
- **pae-CBRCCacheRefreshInterval** – This value is updated from the “Regenerate cache after” configuration (in number of days) at pool advanced storage options. This parameter accepts a value in integers representing minutes. The default is 10080 (1 week).
- **pae-CBRCEnable** – This represents whether the pool is configured to use host caching. In **Pool Settings Advanced Storage Options**, if **Use host caching** is checked this value will be set to 1. Also in pool summary page, in the vCenter information box, host caching will show as enabled.

#### Virtual Machine Host Caching LDAP Attributes

From the View Connection Server OS console, browse **LDAP** through **ADSI Edit**. Navigate to **OU=Servers**. Entries of desktop virtual machines will be listed here. Right-click the specific virtual machine entry and go to its properties to see all the attributes, as illustrated in Figure 15.

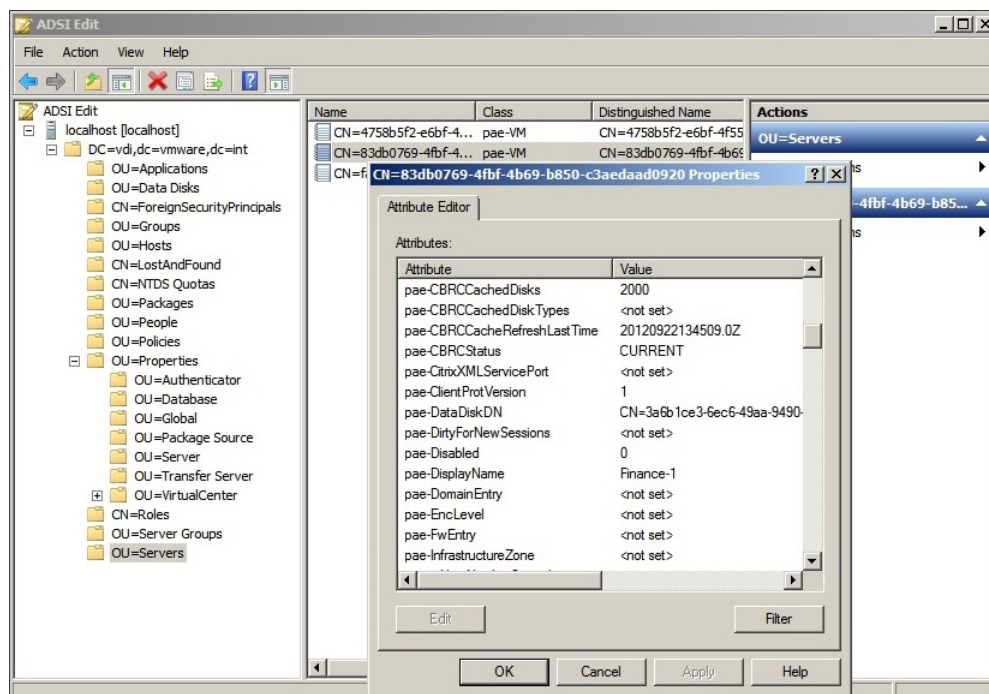


Figure 15: LDAP Host Caching Settings for Desktop Pool

**Note:** Figure 15 lists parameters that are updated dynamically based on the virtual machine's host caching operations. These settings should not be modified manually. It is discussed here for reference purposes only.

- **pae-CBRCCachedDisks** – This attribute contains multiple string values based on the disk(s) that are configured for caching. Values are in the form of numbers that are an internal representation of the disk type used.
- **pae-CBRCCachedDiskTypes** – This attribute contains multiple string values. This highlights the disk types that have currently been configured for host caching.
- **pae-CBRCCacheRefreshLastTime** – This attribute contains a value in the form of LDAP Generalized Time format. This value represents the time of last successful configuration or regeneration of the digest. The value will be blank in case there is no cache regeneration. This is in LDAP UTC/GMT time, for example 20120922134509.0Z (2012 September 22, 13 Hours 45 Minute 09 Seconds). This is displayed in the local time format with respect to local time zone and daylight savings adjustments. A value of 19000101000000.0Z means cache regeneration has never been performed.
- **pae-CBRCStatus** – This string value represents the current virtual machine's CBRC status. Values can be any one of OFF, CURRENT, OUTOFDATE, or ERROR. This is reflected in the virtual machine host caching status of View Administrator.

### Useful Tips

- While planning for host caching-enabled View desktops, the administrator performs guest OS optimization for maximum performance. Refer to [VMware documentation](#) for the agent optimization guide.
- In case of linked clone desktops, consider enabling host caching during pool creation rather than modifying pools after they have been created. This will make digest creation and host caching configuration for clones seamless, prior to completing final virtual machine settings like checkpoint reference.

See item 1 in the [Limitations](#) list. To enable caching on existing linked clone virtual machines, recompose them to a new base image (which might be the same content), so the base disk replica has caching enabled from the outset.

- Ensure the vSphere resource pool / cluster does not contain host(s) that do not support CBRC. If one or more hosts in a cluster are not capable of host caching, the pools provisioned on the cluster will not benefit from host caching.
- If virtual machines in the pools are already powered on prior to host caching configuration, bring down the virtual machines manually (through VI client or View Administrator) to apply the changes.
- In case of an 'ESX Memory Overcommit' scenario, VMware recommends that administrators run 10 percent fewer virtual machines with host caching, as the CBRC also requires memory for caching. If not reduced by 10 percent, overall performance will be impacted.
- The digest re-computation might cause a momentary increase in the number of reads to the back-end storage layer. To avoid conflicts, the administrator can black out peak business hours and plan the cache regenerate operation for off-peak hours.

## Conclusion

The objective of this white paper is to provide information on using the Content-Based Read Cache (CBRC) feature in VMware vSphere 5.0 and show how it is integrated with host caching and related features of VMware View 5.1.

As mentioned in the Introduction, we have first shown how the CBRC back end works and then detailed host caching in VMware View to help users gain a thorough understanding and in-depth knowledge of these features. Though CBRC implementation and its design aspects are discussed through the internal configuration of vSphere, it is not advisable to make any modifications directly on the vSphere host. Any modification to the optional LDAP configuration is not recommended without advice from a VMware Support Professional. Intentionally, not all advanced and in-depth configurations are listed in this paper.

## References

For more information about VMware View 5.1, please visit the [product pages](#).

Additional online documentation can be found through these links:

- [Product Documentation](#)
- [VMware View 5.1 Documentation Center](#)
- [VMware End-User Computing Blog: Optimizing Storage With View Storage Accelerator](#)
- [VMware KB TV: How to enable the storage accelerator in VMware View 5.1](#)

## About the Author

Sivaprasad Govindankutty is a Staff Engineer in the End User Computing Group at VMware. As part of the R&D team, he is an expert in virtual desktop management features and vSphere integration for VMware View.

Follow Sivaprasad's blog at <http://communities.vmware.com/people/skg/blog>.

## Acknowledgements

VMware would like to acknowledge the following individuals for their contributions to this paper and help with content review:

R&D Engineering – Frank Taylor, Rupesh Bajaj

Product Marketing – Matt Lesak, Fred Schimscheimer, Cyndie Zikmund

