INTRODUCTION TO XtremIO SNAPSHOTS

Abstract
This white paper introduces XtremIO snapshots, which refer to the captured state of data in volumes at a particular point in time to enable users to access that data when needed, including after the source volume has changed.

July 2014
Copyright © 2014 EMC Corporation. All Rights Reserved.

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

The information in this publication is provided "as is". EMC Corporation makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

For the most up-to-date listing of EMC product names, see EMC Corporation Trademarks on EMC.com.

Part Number H13035
# Table of Contents

**Executive Summary** ............................................................................................................. 4  
**Conventional Snapshots** ........................................................................................................ 5  
  Conventional Volume Management ......................................................................................... 5  
  Copy-on-Write Snapshots ......................................................................................................... 6  
  Redirect-on-Write Snapshots ................................................................................................... 8  
  Other Snapshot Technologies ................................................................................................... 11  
  Conventional Snapshots - Efficiency and Performance ......................................................... 11  
  Legacy Snapshots Use Cases ................................................................................................... 12  
**Introduction to XtremIO Snapshots** .................................................................................... 13  
  Creating Snapshots .............................................................................................................. 13  
  Capabilities ......................................................................................................................... 14  
**Comparison** ...................................................................................................................... 15  
**Architecture Advantage** ..................................................................................................... 16  
  Cross-consistent Snapshots of Multiple Volumes ................................................................. 20  
  Existence Bitmap .................................................................................................................. 21  
  Snapshot Deletion ................................................................................................................ 22  
  No Garbage Collection ......................................................................................................... 22  
  Volume Snapshot Group ....................................................................................................... 23  
  Remove Shadow Writes ....................................................................................................... 24  
**Testing and Development Use Cases** .............................................................................. 26  
  Backup ............................................................................................................................... 27  
  Logical Data Protection ........................................................................................................ 27  
  Offload Processing ................................................................................................................ 28  
  Provision Bulk VMs by Snapshopting VMFS ........................................................................ 29  
  Consolidation ...................................................................................................................... 29  
  Conclusion .......................................................................................................................... 30
Executive Summary

Snapshots are instantaneous copy images of volume data, capturing the data exactly as it appeared at the specific point in time when the snapshot was created. Snapshots enable users to save the volume data state, and then access the specific volume data at a later time, whenever needed, including after the source volume has changed.

XtremIO snapshots are implemented in a unique way that for the first time maintains space efficiency on writeable snapshots for both metadata and user data. In combination with XtremIO’s unique in-memory metadata architecture, XtremIO snapshots allow for large numbers of high performance, low latency, read/writeable snapshots.

XtremIO snapshots are efficient in metadata and physical space, can be created instantaneously, have no performance impact, and have the same data services as any other volume in the cluster (for example, thin provisioning and Inline Data Reduction).

XtremIO snapshots can be used in a variety of use cases, including:

- Near Continuous Data Protection (CDP) to protect against Logical corruption
- Backup
- Development and test
- Offload processing
- Bulk provisioning of VMs

XtremIO snapshots are easy to use, and appear and are managed as standard volumes in the cluster.
**Conventional Snapshots**

Conventional snapshots were invented primarily for the purpose of data protection in a space-efficient way. Taking a snapshot of a volume or a group of volumes creates a point-in-time copy of the original data set to which the user can roll back when needed.

**Conventional Volume Management**

In conventional block storage arrays, a logical volume is the range of logical addresses within that volume.

![Logical Volume Management Diagram](image)

**Figure 1: Conventional Volume Management**

The addresses are mapped to the physical data blocks on disk. The actual mapping procedure is the responsibility of the logical volume manager. This procedure can be straightforward or complex, depending on factors such as thin provisioning, tiering, deduplication, and other factors.
Copy-on-Write Snapshots

Legacy implementations of snapshots were based on a technology called "copy-on-first-write". In this scheme, metadata pertaining to where original data is stored is copied at the time of snapshot creation, and a new storage pool is defined and reserved in the cluster for the snapshot. No physical copy is produced yet. Every new write triggers a data movement operation between the production volume and the reserved snapshot pool.

The I/O flow is as follows:

1. The cluster receives the new write.
2. The system reads the original data from the production volume.
3. The cluster writes the original data to the reserved snapshot pool.
4. The cluster overwrites production with the new data.

In this scheme, only the metadata pertaining to where original data is stored is copied at the time of the snapshot's creation. No physical copy of the data is performed. The volume manager then tracks the changing blocks on the original volume, while writes to the original volume are performed. The original data being written to is copied to the designated storage pool, set aside for the snapshot before the original data is overwritten (and hence the name "copy-on-write").

This means that every write has a penalty of an additional two I/O operations. One may imagine that, when using high performance media such as SSDs, the approach would be less problematic. This is not necessarily true, as the methodology still poses significant overhead in terms of data movement management. It impacts the latency for both write and read operations, limits the flexibility and performance for complex snapshot topologies (e.g. snapshots of snapshots), and it poses a negative impact on the SSD media endurance.
Read operations are performed in a similar way. Read I/Os to the production volume are always served from the production data reserved pool. Reads to the snapshot are served from the production pool for blocks that remain unchanged, and from the snapshot reserved pool for blocks that have changed.

**Disadvantages:**
- Metadata is copied when a snapshot is taken, which consumes time and capacity and makes it impossible to manage it in memory.
- The reserved snap pool must often be allocated up front, even if not fully utilized, and can run out of space.
- Performance is heavily penalized, especially for writes.
- Complex snapshot topologies experience severe performance degradation.

Figure 2: Copy-on-Write – Host Write
Redirect-on-Write Snapshots

In this scheme, only the metadata pertaining to where original data is stored is copied at the time of the snapshot’s creation. No physical copy of the data is performed. The volume manager tracks the changing blocks on the original volume while writes to the original volume are performed. However, when new data is written to the production volume, the data is written directly to the storage pool, and the volume manager updates the metadata of the production volume (redirected) to the new physical data location (and hence the name "redirect-on-write").
Introduction to XtremIO Snapshots

Disadvantages:

- Typically, metadata is copied when a snapshot is taken, consuming time and capacity.
- In some cases, additional operation is required to make snapshots read/writeable.
- Some implementations do not copy metadata when snapshots are taken, as they are read only. However, once the snapshot is to be accessed in read/write, metadata is copied, thus consuming time and memory.

**Figure 4: Redirect-on-Write - Host Write**
When a snapshot is created, the metadata of the production and that of the snapshot volumes within the volume manager, point to the same physical blocks. Once the production volume receives new writes (assuming that the LBAs of blocks B and D are overwritten), its metadata pointers update to the new block's location.

Reads are performed using the metadata pointers in the volume manager to determine where the physical block resides.

Disadvantage:
- As every snapshot consumes high amounts of metadata in memory, and the amount of memory in the array is limited, it is required to de-stage metadata to SSD, which hurts performance, even on an all-flash array.

Redirect-on-write snapshots are data efficient but not metadata efficient, as the methodology involves copying the entire metadata set of the original volume during the snapshot creation process.
Other Snapshot Technologies

Other technologies are available to make copies of volumes. Clones, typically taken from a static snapshot, provide a full copy of the data (since clones are typically made to physically separate hard drives or SSDs). Although clones provide the same performance as production volumes in the cluster, a clone’s creation time is typically very lengthy as data is copied to the clone.

Split mirrors are used to create clones with more efficiency from a dynamic source. Mirroring is established between the production volume and the clone (a synchronization process that remains on-going until the clone catches up with the production). Once achieved, administrators can split the clone to have an independent copy of the data.

Both technologies offer good performance. However, creation and refresh time is very long with either option. Both clones and split mirrors are inefficient in metadata and data capacity.

Conventional Snapshots - Efficiency and Performance

Writes to snapshots result in fragmentation – the same result holding true for either redirect-on-write or copy-on-write. Furthermore, when multiple snapshots are created, access to the original data, tracking of metadata changes for all snapshots, and fragmentation and reconciliation upon snapshot deletion will result in a heavy performance penalty.

Deleting a redirect-on-write snapshot involves scanning and processing the snapshot metadata, removing any corresponding data blocks belonging exclusively to the snapshot, and thus, potentially moving data from the snapshot reserve pool to the production volume pool. The time taken for this process to complete is proportionate to the original volume size, as opposed to being proportionate to the amount of changed blocks from those in the original volume (since the snapshot’s creation).

Copy-on-write snapshots are not efficient in writes to the original volume. Both copy-on-write and redirect-on-write snapshots have to deal with data fragmentation over time, as well as significant metadata changes. This means that once a snapshot is taken, the I/O performance on the original volume is often negatively impacted.

These performance issues cannot be mitigated in legacy dual controller architecture, let alone with active-passive architecture. The main reason for this is, per volume or snapshot, only one controller (or at best, two controllers) can be involved in the volume’s or snapshot’s management. Scalability is not possible, and a large number of snapshot and production volumes create a significant overhead on the controller, which impacts the performance on snapshots, production volumes or both.
Legacy Snapshots Use Cases

Historically, snapshots were created and used for short periods of time, typically to create a copy of the "live" production data for backup purposes. Snapshots enabled administrators to freeze the production application for a short period of time, take a snapshot, and then resume normal operations. These actions resulted in a static copy of the production data, typically accessed in read-only mode, which could then be backed up to an external backup device. The reasons why snapshots were only used for short periods of time included performance issues, capacity utilization considerations, and the limited number of supported snapshots.

Later on, following the development of redirect-on-write technologies, snapshot usage was extended for longer periods of time, mainly for testing and development processes. However, snapshot usage was limited because in most cases performance was impacted, either due to copy-on-write snapshot implementations having a high penalty on performance for the production environment, or redirect-on-write snapshots introducing increased read-latency, resulting from linked metadata data scanning in the array, or overhead on the controllers' CPUs and data fragmentation.

Introduction to XtremIO Snapshots

XtremIO snapshots are inherently writeable, but may be mounted as read-only to maintain immutability. It is possible to take snapshots from either the source or any snapshot of the source volume.

Snapshots can be used in a number of use cases, including:

- **Logical corruption protection**
  It is possible to create frequent snapshots (based on the desired RPO intervals) and use them to recover from any logical data corruption. A snapshot can be kept in the system for as long as it is needed. If a logical data corruption occurs, it is possible to use a snapshot of an earlier application state (prior to the logical data corruption occurrence) to recover the application to a known good point in time.

- **Backup**
  It is possible to create snapshots to be presented to a backup server/agent. This can be used in order to offload the backup process from the production server.

- **Development and test**
  It is possible to create snapshots of the production data, create multiple (space-efficient, high-performance) copies of the production system and present them for development and testing purposes.

- **Offload processing**
  It is possible to use snapshots as a means to offload the processing of data from the production server. For example, if there is a need to run a heavy process on the data (which can affect the production server’s performance), it is possible to use snapshots to create a recent copy of the production data and mount it on a different server. This process can then be run on the other server without consuming the production server’s resources.

**Creating Snapshots**

For detailed instructions on creating snapshots from volumes, from sets of volumes and from folders, refer to the *XtremIO Storage Array User Guide*. 
**Capabilities**

With XtremIO:

- Snapshots are created instantaneously and provide a workable copy of the production volume.
- Snapshots are read-write copies of source volumes.
- Snapshots appear and are managed in the cluster as regular volumes.
- Snapshots have the same data services as any volume in the system; the inline global deduplication and thin provisioning features are in constant operation.
- XtremIO Snapshots are both metadata and data efficient.
- Snapshotting requires no reserved space.
- The system supports consistent snapshots on multiple volumes.
- Snapshots can be on snapshots at any hierarchy or span.
- Deletion of a volume or any of its snapshots does not affect the child snapshot or parent snapshot/volume.
- The system provides predictable and consistent performance on production volumes and snapshots.
Comparison

Table 1 summarizes some of the key differences between the various snapshot technologies and XtremIO snapshots.

### Table 1: Snapshot Technologies vs. XtremIO Snapshots

<table>
<thead>
<tr>
<th></th>
<th>Copy-on-Write</th>
<th>Redirect-on-Write</th>
<th>Full Clones</th>
<th>XtremIO Snapshots</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Space-Efficient Data</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes + Inline Data Reduction</td>
</tr>
<tr>
<td><strong>Space-Efficient Metadata</strong></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Volumes' and Snapshots' Metadata</strong></td>
<td>Metadata served from disk and memory</td>
<td>Metadata served from disk and memory</td>
<td>Metadata served from disk and memory</td>
<td>Metadata always 100% in-memory</td>
</tr>
<tr>
<td><strong>Creation Time</strong></td>
<td>Instantaneous</td>
<td>Instantaneous</td>
<td>Long time</td>
<td>Instantaneous</td>
</tr>
<tr>
<td><strong>Performance Impact on Production</strong></td>
<td>High impact</td>
<td>Moderate impact</td>
<td>No impact after clone is completed</td>
<td>No impact</td>
</tr>
<tr>
<td><strong>Performance of Snapshots</strong></td>
<td>Degraded</td>
<td>Degraded</td>
<td>Same as production</td>
<td>Same as production</td>
</tr>
<tr>
<td><strong>Rapid Deletes</strong></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Delete Limitations</strong></td>
<td>Limited</td>
<td>Limited</td>
<td>n/a</td>
<td>Can delete any snapshots anywhere in the tree without affecting parents or children.</td>
</tr>
<tr>
<td><strong>Topology Limitations</strong></td>
<td>Limited</td>
<td>May support snap on snaps</td>
<td>No snap on snap support</td>
<td>Any topology</td>
</tr>
<tr>
<td><strong>Up-front Space Reservation</strong></td>
<td>Yes</td>
<td>May require space reservation</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>Data Services Limitations</strong></td>
<td>Yes</td>
<td>May limit data services</td>
<td>No</td>
<td>No – full data services available</td>
</tr>
</tbody>
</table>
Architecture Advantage

XtremIO snapshots provide:

- **Space efficient metadata**
  - Creation of a snapshot does not consume metadata.
  - Metadata is consumed only for new and globally unique data blocks.

- **No performance impact**
  - There is equal performance with snapshots and volumes.
  - No impact occurs when snapshots are created.
  - Read performance is equal on all snapshot hierarchy levels.

- **Inline-Data-Reduction-enhanced "redirect-on-unique-write" method**
  - Consumes space only for new unique data blocks.
  - There is no physical data movement on new writes.

- **Even distribution of system resources**
  - All Storage Controllers in the cluster are constantly engaged in managing the I/O data flow and metadata, regardless of the entity type.
  - More CPU power and memory remain available (a single controller, versus multiple controllers).
  - There is consistent even distribution of workload across all available resources.

- **Optimized for flash memory**
  - XtremIO snapshots are optimized for maximum flash endurance.
  - No data movements occur on snapshot creation or during writes.
  - Inline Data Reduction provides added value in capacity efficiency and flash endurance.
  - There are additional benefits with the flash media (in terms of performance).
  - The system provides efficient metadata and capacity utilization.
XtremIO's snapshot technology is implemented by leveraging the array's content-addressing capabilities along with in-memory metadata and the system's dual stage metadata (which results in Inline Data Reduction), optimized for SSD media with a unique metadata tree structure that directs I/Os to the right data timestamp. This allows efficient snapshotting that sustains high performance, while maximizing media endurance, both in terms of ability to create multiple snapshots and the amount of I/Os that a snapshot can support.

![Figure 6: Address-to-Content Mapping](image)

In the logical volume diagram shown in Figure 6, every written block address is mapped to a fingerprint. Therefore, this mapping is called Address-to-Content. Additionally, there is a separate metadata mapping from the content to the actual unique physical blocks that are written to the SSDs (thus forming the dual-stage metadata structure).

As every XtremIO volume is thinly provisioned, the addresses that have not been written remain empty and do not occupy metadata (or data) space. Therefore, the XtremIO thin provisioning is 100% space efficient.

When creating a snapshot, the system generates a pointer to the ancestor metadata of the actual volume in the cluster. Therefore, creating a snapshot is a very quick operation that does not have any impact on the cluster and does not consume any physical or logical capacity. Snapshot capacity consumption occurs only if a change requires writing a new block.

![Figure 7: XtremIO Snapshots](image)
When a snapshot is created, the existing metadata for the volume becomes an "ancestor" entity that is shared between the production volume and the snapshot. New empty containers are created for subsequent changes to the production volume and the snapshot volume. Thus, the act of creating a snapshot is instantaneous and involves no data or metadata copies.

When a new block is written to the ancestor, the system updates the metadata of the ancestor volume to reflect the new write and stores the block in the cluster, using the standard write flow process. As long as this block is shared between the snapshots and the ancestor volume, it is not deleted from the cluster following a write. This applies to both a write in a new location on the volume (a write on an unused LBA) and to a rewrite on an already written location.

The cluster manages both the snapshot's metadata and the ancestor's metadata via a tree structure. The snapshot and the ancestor volumes are represented as leaves in this structure, as shown in Figure 8.

The metadata is shared between all snapshots that remain unchanged (from the snapshot's original ancestor). The snapshot maintains unique metadata only for an LBA with a data block differing from that of its ancestor. This provides economical metadata management.

When a new snapshot is created, the cluster always creates two leaves (two descendant entities) from the snapshotted entity. One of the leaves represents the snapshot, and the other one becomes the source entity. The snapshotted entity is no longer used directly, but is kept in the cluster for metadata management purposes only.
Figure 9 illustrates a 16-block volume in the XtremIO system. The first row [marked as $A(t_0)/S(t_0)$] shows the volume at the time the first snapshot was taken ($t_0$). At $t_0$, the ancestor [$A(t_0)$] and the snapshot [$S(t_0)$] have the same data and metadata, because $S(t_0)$ is the read-only snapshot of $A(t_0)$ (containing the same data as its ancestor).

In Figure 9, before creating the snapshot at $S(t_1)$, two new blocks are written to $P$:
- H8 overwrites H2 at LBA 3.
- H2 is written to LBA D. But the data does not consume more physical capacity because it has the same fingerprint as the data stored in LBA 3 in $A(t_0)$ (H2).
$S_{(t1)}$ is a read/write snapshot. It contains two additional blocks (at LBA 2 and LBA 3) that are different from its ancestor snapshot.

Unlike traditional snapshots implementations, which reserve space for changed blocks and reserve an entire copy of the metadata for each snap, XtremIO does not require any reserved physical space for snapshots, and never has metadata bloat. Snapshots only use those resources when needed and they are consumed from the cluster's global resource pool. There is no pool management in XtremIO.

All the accessible entities in the snapshot tree representing the volume, including all snapshots originating from that volume, are managed by an entity called the Volume Snapshot Group (VSG).

An XtremIO snapshot only consumes metadata for new writes (unshared blocks), and specifically utilizes shared metadata from the snapshot’s ancestor entities. This allows the cluster to efficiently maintain large numbers of snapshots, using a very small storage overhead which is dynamic and proportional to the amount of changes in the entities.

For example, at time t2, LBAs 0, 3, 4, 6, 8, A, B, D and F are shared with the ancestor’s entities. Only LBA 5 (H6) is unique for this snapshot. Therefore, XtremIO consumes only one metadata unit. The remaining blocks are shared with the ancestors and use the ancestor data structure in order to compile the correct volume data and structure.

**Cross-consistent Snapshots of Multiple Volumes**

The system supports the creation of snapshots on a set of volumes. All snapshots from the volumes in the set are cross-consistent. Snapshots on a set of volumes can be created manually by selecting a set of volumes for snapshotting or by placing volumes in a consistency group container folder and creating a snapshot of the consistency group from the folder.

In order to guarantee that the created snapshots are cross-consistent, the cluster quiesces the volumes within microseconds. No impact on system performance occurs, even when this operation is repeated in short intervals. Only the snapshotted source volumes are quiesced during the snapshot operation.

In order to guarantee consistency, the cluster holds any acknowledgments back to the hosts for any writes to the source volumes during the quiesce procedure, thus guaranteeing that no new writes are generated from the initiator during the quiescence. As a result, all snapshots are cross-consistent.
**Existence Bitmap**

The XtremIO Storage Array has an additional data structure, called the existence bitmap.

![Snapshot Tree Diagram](image)

- **External entity**
- **Ancestor entity (internal)**

**Figure 10: Snapshot Tree**

The XtremIO snapshot implementation enables taking a snapshot of a snapshot. In systems that allow such cascading snapshots, finding the location from which the data should be retrieved is a challenge that may impact the read performance. For each LBA in the snapshot, the data may be found either in that snapshot volume itself, in cases where that LBA was written to after the snapshot was taken, or in one of its ancestors.

The native algorithm for reading from a space-efficient snapshot is to check whether or not the data was written to the snapshot volume itself; if not, to check its parent, and so on. In some cases, if that LBA was not ever written, the original ancestor is reached, resulting in a lengthy search. This algorithm has a high performance penalty on reads. Furthermore, the performance is unpredictable as it depends on the length of the chain. Additionally, snapshots that are situated far from the root volume experience worse performance than snapshots close to the root.

The existence bitmap data structure and algorithm optimize read operations on space-efficient snapshots. The existence bitmap provides predictable, even performance for all snapshots, regardless of their distance from the root volume.
There is one structure per Volume Snapshot Group (VSG) containing bitmaps; one for each LBA in the original volume. The number of bits in each bitmap is equal to the maximal number of volumes per VSG. Whenever data is written to a volume, regardless of whether it is the original volume or a snapshot, the corresponding bit to that volume in the corresponding bitmap to the LBA is set.

The bitmap in Figure 10 maps, per LBA, where the LBA is written per snapshot. Each index in the bitmap represents a snapshot. For example, only $A_{t0}$ is written to LBA 0. Therefore, every read of LBA 0 for snapshots created after time $t1$ should be served by the metadata information stored at $A_{t0}$.

When reading data from a specific LBA, the cluster first reads the bitmap associated with this LBA (which typically fits in one cache line and is very efficient). It then finds, from the bitmap, which volume should be accessed in order to read the data. The cluster then goes directly to that volume to retrieve the data. As bitmap manipulation is an inexpensive operation in terms of performance, the depth of the snapshot in the snapshot chain affects neither the read nor the write performances.

### Snapshot Deletion

Snapshot deletions are lightweight and proportional only to the amount of changed blocks between the entities. The cluster uses its content-aware capabilities to handle snapshot deletions.

Each unique data block has a counter that indicates the number of instances of that block in the cluster. When a block is deleted, the counter value is decreased by one. Any block with a counter value of zero (meaning there is no logical block address [LBA] across all volumes or snapshots in the cluster referring to this block) is overwritten by XDP when new unique data enters the cluster.

Deleting a snapshot in the middle of the tree triggers a process which merges the metadata of the deleted entity's children with that of their grandparents. This process ensures that the tree structure is not fragmented.

### No Garbage Collection

With XtremIO, every block that needs to be deleted is immediately marked as "free". Therefore, there is no garbage collection in the SSD, and the cluster does not have to perform any scan process to locate and delete orphan blocks.

XtremIO’s snapshot implementation is entirely metadata driven and leverages the array’s Inline Data Reduction to ensure that data is never copied within the array. Thus many snapshots can be maintained.
**Volume Snapshot Group**

The Volume Snapshot Group is an entity that represents all external entities (entities that can be mapped) of a snapshot tree. All snapshots originating from a single ancestor share the same Volume Snapshot Group, which is created whenever a new volume is defined in the cluster.

The VSG information can be viewed in the GUI's Volume Properties pane.

![Volume Properties Pane in GUI](image)

*Figure 11: Volume Properties Pane in GUI*
The VSG information can also be viewed under the VSG Index, using the following CLI command:

```
show-volumes or show-volume
```

```
<table>
<thead>
<tr>
<th>Volume-Name</th>
<th>Index Vol-Size LB-Size VSG-Space-In-Use Offset Ancestor-Name Index VSG-Index Cluster-Name Index Parent-Folder</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCHIVE-0:Master</td>
<td>22 500G 512 0 0 ARCHIVE-0 1 1 vhex-shrink33 1 /Oracle_Master</td>
</tr>
<tr>
<td>ARCHIVE-0</td>
<td>1 500G 512 0 0 1 vhex-shrink33 1 /Oracle</td>
</tr>
<tr>
<td>ARCHIVE-1</td>
<td>2 500G 512 0 0 2 vhex-shrink33 1 /Oracle</td>
</tr>
<tr>
<td>ARCHIVE-2</td>
<td>3 500G 512 0 0 3 vhex-shrink33 1 /Oracle</td>
</tr>
</tbody>
</table>
```

To list information on all existing volume snapshot groups in the cluster, use the following CLI command:

```
show-volume-snapshot-groups
```

```
<table>
<thead>
<tr>
<th>Name Index Num-of-Vols Vol-Rise Thin-Provisioning-Ratio Logical-Space-In-Use Num-Of-Active-Merges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 100% 0.000 0.000 0 0</td>
</tr>
<tr>
<td>2 2 100% 0.000 0 0</td>
</tr>
<tr>
<td>3 2 100% 0.000 0 0</td>
</tr>
<tr>
<td>4 2 100% 0.000 0 0</td>
</tr>
<tr>
<td>5 3 2.19% 0.000 0 0</td>
</tr>
<tr>
<td>6 3 2.19% 0.000 0 0</td>
</tr>
</tbody>
</table>
```

**Remove Shadow Writes**

Writes to the same LBA on a volume and its snapshots frees the LBA data in the entities' shared ancestor. This improves the array utilization, both in terms of metadata consumption and physical capacity consumption.

Once a snapshot is created, the result is a simple structure with three entities:

1. The ancestor entity
2. A new snapshot entity
3. A new production entity

In the following example, a volume is completely full with unique, non-deduplicated data, and all LBAs contain data.

Observe the impact on managing the data and metadata under the following scenario, where the snapshot is written on LBA 4 and after a while LBA 4 on the production is over-written.
Initially, when the snapshot is created, both the production and the snapshot contain no unique data. All reads are served, using the metadata and data of the ancestor entity.

Assume that the later writes to LBA 4 were written both for the production and the snapshot. The ancestor LBA 4 is now shadowed by the updates to LBA 4 on the snapshot and the production volume. Therefore, the data in the ancestor is no longer needed and can be deleted from the cluster. The cluster frees the metadata related to LBA 4 and decreases the reference count of the fingerprint stored on LBA 4. If that fingerprint is not used elsewhere in the cluster, the fingerprint and its corresponding content are deleted from the cluster, freeing both physical and logical capacity.

Removal of shadow writes occurs asynchronously without impacting the cluster performance.

Figure 12: Removing Shadow Writes
Testing and Development Use Cases

XtremIO snapshots can be leveraged to provide Test and Development copies of production data. Multiple master copies can be created, and each copy can be processed (such as an anonymization process) to be prepared as a golden image for development or testing purposes. Multiple copies can then be created from each master copy and presented to various development teams. Provisioning more copies is an easy and instantaneous process. Further snapshots of the provisioned copies can also be taken.

XtremIO snapshot efficiency enables copies to be created based on demand for maximum business efficiency, rather than based on storage capacity or performance limitations.

Key Benefits:

- Space-efficient snapshots.
  - Resources are only appropriated for new writes.
  - No physical or logical capacity is reserved.
- Dozens of Test and Development copies can be created, and creating a high performance sandbox for every engineer is feasible.
- Quick refresh of Test or Development environments.
- Deduplication, compression, and thin provisioning are always enabled.
- Snapshot of a snapshot is supported.
Backup

XtremIO snapshots can be used for backup of production data into an external backup device. The backup can be taken from the snapshots instead of the production copy, thus offloading the process of reading the data to an external backup server.

Key Benefits:

• Space efficient snapshots.
  ▪ No physical or logical capacity is reserved or used in snapshots.
  ▪ Efficient logical memory consumption.
• Immediate creation and mapping of snapshots.
• No resources are appropriated from production for backup.
  ▪ Free SAN BW/IOPS resources.
  ▪ Free CPU resources.
  ▪ Free network resources.

Logical Data Protection

It is possible to create a snapshot of the production volumes to protect against logical data corruption. Snapshots can be created over a short interval to provide fine Recover Point Objective (RPO) in a cyclic manner (a superior RPO to a backup).

For example, it is possible to create 48 snapshots every 30 minutes and delete the latest snapshot. This provides an RPO of 30 minutes for the last day of production changes.

Key Benefits:

• Space-efficient snapshots
  ▪ No physical capacity is reserved.
• Immediate creation of snapshots
• Fast restore from data corruption
• Fine Recover Point Objective
  ▪ Snapshot can be seconds apart.
Offload Processing

XtremIO snapshots can be used to:

- Offload processing to an external server.
- Extract, Transform and Load (ETL) processes to load data into a data warehouse.
- Achieve near real-time analytics for business intelligence (BI) reports.
- Consolidate online transaction processing (OLTP) and real-time on a single platform.

Key Benefits:

- Immediate updated copy of production without the need to make a brute force SAN copy
- Consolidating different workloads into a single platform
- Accurate real-time BI analytics based on the most updated copy of production data
- Space-efficient snapshots
  - No physical or logical capacity is reserved or used in snap.
  - Deduplication and thin provisioning are always on.
- Offloading processing from the production server
  - Free SAN BW/IOPS resources.
  - Free CPU resources.
  - FREE Network resources.
- High performance on copies of production
Provision Bulk VMs by Snapshotting VMFS

Taking a snapshot on a VMFS volume containing many virtual machines has proven to be the fastest method of cloning VMs. This method is much faster than any alternatives, such as the VAAI XCOPY method.

The cloned VMs on the snapshot should be configured and registered with vSphere (requiring sys-prepping the VMs and setting the OS name).

This can enable fast deployment of bulk VMs for VDI or VSI purposes.

**Key Benefits:**
- Fast provisioning of large scale VMs
- Space-efficient cloning of VMs

Consolidation

The XtremIO snapshot feature enables consolidation of production data with testing and development and real-time analytics. Having different workloads on the XtremIO array does not require any special considerations, as long as the overall aggregated performance of the workload is within the system performance specifications.

XtremIO can serve all copies of the data at the same low latency and high IOPS performance. Furthermore, as test data repeats itself, the testing or development copies benefit from the deduplication capabilities of XtremIO, leading to further savings.

Creation of copies is an easy process, and refresh cycle time is minimal.
Conclusion

The XtremIO snapshot feature offers a large number of high performance, low latency, read/writeable snapshots.

Snapshots are efficient in metadata and physical space, can be created instantaneously, have no performance impact, and have the same data services as any other volume in the cluster (such as thin provisioning and Inline Data Reduction).

XtremIO snapshots are easy to use and manage, and leverage a sophisticated metadata management engine that provides superior support for flash media, enabling high performance snapshotting.

XtremIO snapshots can be used for Development and Test, backups, protection against logical corruption, offload processing (real-time analytics, SAP landscape, and more) and consolidation of testing and development with production.