

# N series SnapMirror Async Guide

Disaster recovery with SnapMirror

Data replication with SnapMirror

Application development  
with SnapMirror



Alex Osuna  
Srinath Alapati  
Kyle Burrell  
Larry Touchette

# Redbooks





International Technical Support Organization

**N series SnapMirror Async Guide**

October 2011

**Note:** Before using this information and the product it supports, read the information in “Notices” on page vii.

**First Edition (October 2011)**

This edition applies to Data ONTAP 7.3.2 and above.

© Copyright International Business Machines Corporation 2011. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	vii
Trademarks .....	viii
<b>Preface</b> .....	ix
The team who wrote this book .....	ix
Now you can become a published author, too! .....	x
Comments welcome .....	x
Stay connected to IBM Redbooks .....	x
<b>Chapter 1. Introduction to SnapMirror</b> .....	1
1.1 Business applications .....	2
1.1.1 Disaster recovery .....	2
1.1.2 Remote data access .....	3
1.1.3 Application development, testing, and disaster recovery testing .....	3
1.1.4 Remote tape archiving .....	4
1.1.5 Load sharing .....	4
1.2 Benefits of SnapMirror .....	4
1.3 Summary of SnapMirror changes in Data ONTAP 7.3 .....	5
1.3.1 Increased concurrent transfers .....	5
1.3.2 Efficient use of multiprocessor systems .....	5
1.3.3 Efficient computation of changed blocks .....	5
<b>Chapter 2. Overview of SnapMirror</b> .....	7
2.1 The basics .....	8
2.2 Control files .....	8
2.2.1 snapmirror.conf .....	8
2.2.2 snapmirror.access .....	9
2.2.3 snapmirror.allow .....	9
2.3 Update process .....	9
2.4 Snapshot copy behavior in SnapMirror .....	10
2.5 Volume SnapMirror and qtree SnapMirror .....	12
2.6 SnapMirror volume replication .....	12
2.6.1 Characteristics of volume SnapMirror .....	12
2.6.2 Snapshot copy behavior and status in volume SnapMirror .....	13
2.7 SnapMirror qtree replication .....	14
2.7.1 Snapshot copy behavior and status in qtree SnapMirror .....	15
2.7.2 Key differences between volume and qtree SnapMirror .....	15
2.8 Support for volume types .....	17
2.9 Modes of SnapMirror .....	17
2.9.1 SnapMirror Async .....	17
2.9.2 SnapMirror Sync .....	18
2.9.3 SnapMirror Semi-Sync .....	20
2.9.4 Visibility interval .....	20
2.10 Configuration files .....	20
2.10.1 Access and security .....	20
2.10.2 Distribution .....	21
2.10.3 Configuration changes .....	22
2.11 Multipath support .....	22
2.11.1 Multiplexing mode .....	22

2.11.2	Failover mode	23
2.11.3	SnapMirror network compression	23
2.11.4	Network bandwidth versus RPO	23
2.11.5	What is SnapMirror network compression	24
2.11.6	Prerequisites for SnapMirror network compression	24
<b>Chapter 3.</b>	<b>Operational behaviors</b>	<b>25</b>
3.1	Active-Active configuration	26
3.2	Disk geometry	26
3.3	Cascading	26
3.3.1	Example of cascading	27
3.3.2	Snapshot copy propagation in a cascade configuration	28
3.4	Logging	30
3.5	Data ONTAP versions and resync	30
3.6	Data change rate	31
3.7	SnapMirror and LUNS	31
3.8	Space guarantees	32
3.8.1	Overcommitting aggregates on the source system	32
3.8.2	Overcommitting aggregates on the destination system	32
3.9	Update failures	33
3.10	Concurrent replication operations	34
3.11	NearStore personality	35
3.12	System-wide throttle	36
3.13	Dynamic throttle	36
3.14	Firewall configuration	37
3.14.1	SnapMirror Async	37
3.14.2	SnapMirror Sync and SnapMirror Semi-Sync	37
3.15	Network compression configuration and operation configuration	38
3.16	Enabling and disabling network compression	38
3.17	Reporting the compression ratio	39
3.18	Compression ratio and data sets	39
3.19	64-bit aggregates	40
3.20	SnapMirror over Fibre Channel	40
<b>Chapter 4.</b>	<b>Guidelines for SnapMirror</b>	<b>41</b>
4.1	Growing destination volume	42
4.2	SnapMirror window size, network latency, and compression	42
4.2.1	Window size changes in Data Ontap 7.3.2	42
4.2.2	Volume SnapMirror TCP window size limits	42
4.2.3	SnapMirror TCP window size and throughput	43
4.2.4	SnapMirror TCP window size and compression	43
4.3	Replication network configuration	44
4.4	Replication frequency and Snapshot schedules	44
4.5	Destination Qtree names	45
4.6	Many-to-one configuration	45
4.7	Upgrading to flexible volumes	45
4.8	Unicode	45
4.9	High file count environments and Qtree SnapMirror	45
4.10	Read performance on a FlexVol volume SnapMirror destination	46
4.11	Data ONTAP upgrade and revert considerations	46
4.12	SnapMirror network compression considerations	47
4.12.1	Compression versus decompression	47
4.12.2	Transfer times	47

4.12.3 Compression ratio . . . . .	48
<b>Chapter 5. Network-free seeding . . . . .</b>	<b>51</b>
5.1 SnapMirror to tape . . . . .	52
5.1.1 Overview . . . . .	52
5.1.2 Restrictions . . . . .	52
5.2 Logical Replication (LREP) . . . . .	53
5.2.1 Overview . . . . .	53
5.2.2 Downloading the LREP tool . . . . .	53
<b>Chapter 6. SnapMirror management . . . . .</b>	<b>55</b>
6.1 Protection Manager . . . . .	56
6.2 Concepts . . . . .	56
6.3 Data conformance and monitoring . . . . .	57
6.4 Disaster recovery . . . . .	57
<b>Chapter 7. Use of SnapMirror with other N series products . . . . .</b>	<b>59</b>
7.1 N series Manageability Suite . . . . .	60
7.1.1 Application Suite and Database Suite . . . . .	60
7.1.2 Server Suite . . . . .	61
7.2 FlexClone . . . . .	62
7.2.1 Volume SnapMirror, SnapDrive, and FlexClone . . . . .	62
7.2.2 Qtree SnapMirror and FlexClone . . . . .	64
7.2.3 Splitting and destroying a clone . . . . .	65
7.2.4 Invalid operations . . . . .	65
7.3 Setting up replication between FlexClone volumes . . . . .	66
7.4 SnapVault . . . . .	69
7.4.1 Differences between SnapMirror and SnapVault . . . . .	69
7.4.2 Two types of SnapVault deployments with SnapMirror . . . . .	70
7.5 SnapLock . . . . .	72
7.5.1 Replication restrictions . . . . .	72
7.5.2 End-to-end Snaplock Compliance . . . . .	73
7.5.3 Synchronous replication . . . . .	73
7.6 MultiStore . . . . .	73
7.6.1 Virtual storage controller DR . . . . .	73
7.6.2 Virtual storage controller migration . . . . .	74
7.7 MetroCluster . . . . .	75
7.8 FlexShare . . . . .	76
7.9 Deduplication for N series . . . . .	76
7.9.1 Volume SnapMirror . . . . .	76
7.9.2 Qtree SnapMirror . . . . .	77
7.9.3 Data ONTAP 7.3 and volume SnapMirror . . . . .	77
7.9.4 Failback . . . . .	78
7.9.5 Migration . . . . .	79
<b>Chapter 8. Tips for troubleshooting . . . . .</b>	<b>81</b>
<b>Chapter 9. Examples . . . . .</b>	<b>83</b>
9.1 Failover and failback with SnapMirror . . . . .	84
9.1.1 Planned failover (no disaster) . . . . .	84
9.2 Failover in the event of a real disaster . . . . .	85
9.2.1 Failover . . . . .	85
9.2.2 Replicating to the primary site . . . . .	86
9.2.3 Performing failback to the primary site . . . . .	86

9.2.4 Replicating to the DR site . . . . .	86
9.2.5 Housekeeping . . . . .	87
9.3 SnapLock and qtree SnapMirror resync . . . . .	87
9.3.1 Production failure . . . . .	87
9.3.2 DR testing . . . . .	87
9.4 Making the SnapVault destination writable . . . . .	88
9.5 Migrating SnapVault by using SnapMirror. . . . .	89
<b>Related publications . . . . .</b>	<b>91</b>
IBM Redbooks . . . . .	91
Other publications . . . . .	91
Online resources . . . . .	91
Help from IBM . . . . .	92
<b>Index . . . . .</b>	<b>93</b>



# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

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.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

# Trademarks


IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

IBM®

Redbooks®

Redpaper™

Redbooks (logo) ®

System Storage®

The following terms are trademarks of other companies:

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Snapshot, Network Appliance, FlexShare, FlexCache, WAFL, SnapVault, SnapRestore, SnapMover, SnapMirror, SnapManager, SnapLock, SnapDrive, NearStore, MultiStore, FlexVol, FlexClone, FilerView, DataFabric, Data ONTAP, NetApp, and the NetApp logo are trademarks or registered trademarks of NetApp, Inc. in the U.S. and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

# Preface

This IBM® Redbooks® publication presents an overview of implementing N series SnapMirror Async technology, with step-by-step configuration examples and guidelines to assist the reader in designing an optimal SnapMirror solution.

There are several approaches to increasing data availability in the face of hardware, software, or even site failures. Backups provide a way to recover lost data from an archival medium (tape or disk). Redundant hardware technologies also help mitigate the damage caused by hardware issues or failures. Mirroring provides a third mechanism to facilitate data availability and minimize downtime.

SnapMirror offers a fast and flexible enterprise solution for mirroring or replicating data over local area, wide area, and Fibre Channel (FC) networks. SnapMirror can be a key component in implementing enterprise data protection strategies.

If a disaster occurs at a source site, businesses can access mission-critical data from a replica on a remote N series storage system for uninterrupted operation.

## The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

**Alex Osuna** is a Project Leader at the International Technical Support Organization, Tucson Center. He writes extensively on all areas of data storage. During his 32 years in the IT industry Alex has held positions in maintenance, support, planning, sales, and publishing, focusing on storage. Alex holds certifications from IBM, Microsoft, Red Hat, and the Open Group.

**Srinath Alapati** is a Technical Marketing Engineer at Network Appliance Inc. Srinath joined NetApp in 2004 and has been a member of the Data Protection group for over two years. He has 10+ years of experience in IT, managing servers and storage infrastructure. Srinath has authored or coauthored multiple technical reports on SnapMirror, MetroCluster, VMware, and Exchange and speaks at various technical conferences. He is also a core team member involved in NetApp IT's disaster recovery implementation.

**Kyle Burrell** was an IT Specialist in Boulder, CO, USA. He was with IBM for 18 years. The last seven years were focused on storage.

**Larry Touchette** is a Technical Marketing Engineer in the NetApp Server Virtualization and Grid Infrastructure Business Unit. He moved into this role about 8 months ago. Prior to that he was on the NetApp Professional Services team for over 6 years, as both a PS Engineer and a PS Consultant. November marked the start of his 8th year at NetApp. Prior to NetApp he built a systems administration and engineering background in the automotive and IT tech industries.

Thanks to the following people for their contributions to this project:

Bill Wilson  
Development Engineering Technician Tucson Arizona

## Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author - all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- Send your comments in an email to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>



# Introduction to SnapMirror

This IBM Redbooks publication presents an overview of implementing SnapMirror Async technology, with step-by-step configuration examples and guidelines to assist the reader in designing an optimal SnapMirror solution.

In this chapter we discuss why SnapMirror is needed.

## 1.1 Business applications

There are several approaches to increasing data availability in the face of hardware, software, or even site failures. Backups provide a way to recover lost data from an archival medium (tape or disk). Redundant hardware technologies also help mitigate the damage caused by hardware issues or failures. Mirroring provides a third mechanism to facilitate data availability and minimize downtime. SnapMirror offers a fast and flexible enterprise solution for mirroring or replicating data over local area, wide area, and Fibre Channel (FC) networks. SnapMirror can be a key component in implementing enterprise data protection strategies. If a disaster occurs at a source site, businesses can access mission-critical data from a replica on a remote N series storage system for uninterrupted operation.

By providing a simple solution for replicating data across local, wide area, and FC networks, SnapMirror addresses the following critical business issues.

### 1.1.1 Disaster recovery

If critical data is replicated to a different physical location, a serious disaster does not necessarily mean extended periods of data unavailability. The client can access replicated data across the network until the damage caused by the disaster is repaired. Recovery might include recovery from corruption, natural disaster at the production site, accidental deletion, sabotage, and so on.

SnapMirror is often an integral part of disaster recovery plans. Data could be replicated to a destination system at a disaster recovery (DR) facility. Preferably, application servers would be replicated to this facility as well. If the DR facility needs to be made operational, applications can be switched over to the servers at the DR site and all application traffic can be directed to these servers for as long as necessary to recover the production site. When the source site is back online, SnapMirror can be used to transfer the data efficiently back to the production storage systems. After the production site takes over normal application operations again, SnapMirror transfers to the DR facility (see Figure 1-1) can resume without requiring a second complete data transfer.

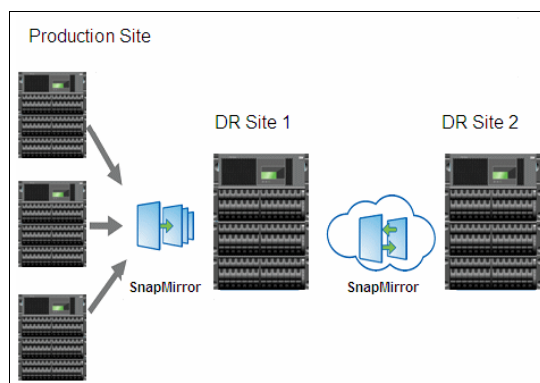


Figure 1-1 SnapMirror illustration

### 1.1.2 Remote data access

The data replication capability of SnapMirror allows the distribution of large amounts of data throughout the enterprise, enabling read-only access to data at DR and remote locations. Remote data access not only provides faster access to data by local clients, it also results in a more efficient and predictable use of expensive network and server resources. Storage administrators can replicate production data at a chosen time to minimize overall network utilization (see Figure 1-2).

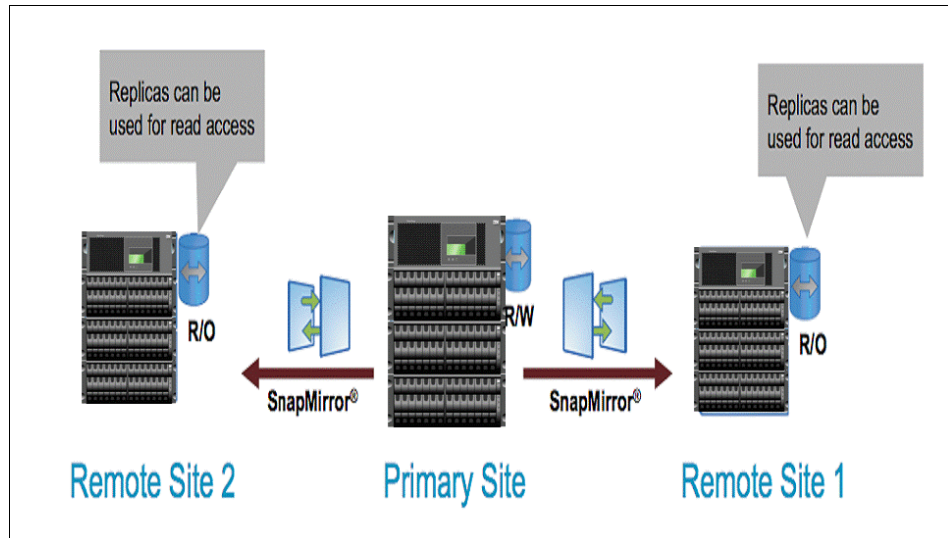


Figure 1-2 Remote read access to SnapMirror replicas

### 1.1.3 Application development, testing, and disaster recovery testing

Examples include test beds, database environments used for testing or simulating production environments, performance testing and monitoring, and development testing (see Figure 1-3).

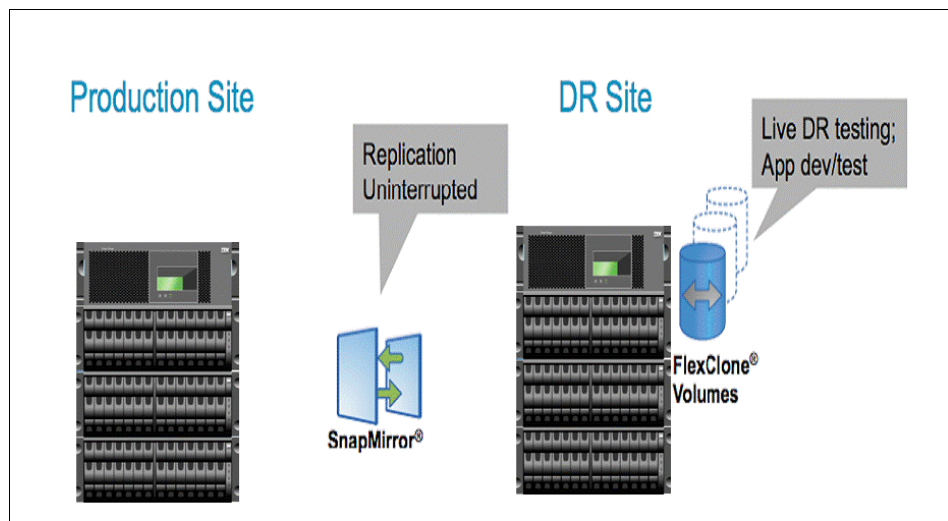


Figure 1-3 Application development/test and DR testing with SnapMirror and FlexClone

When FlexClone is used along with SnapMirror, the remote site can be used for live DR testing without interrupting production operations and replication.

### 1.1.4 Remote tape archiving

Some environments require off-site storage or off-site archiving. When a tape device is attached to an N series SnapMirror destination, data can be moved to tape periodically. SnapMirror can also be used for backup consolidation and for offloading tape backup overhead from production servers. This facilitates centralized backup operations, reducing backup administrative requirements at remote locations. It can also dramatically reduce overhead from stressful backup operations caused by small backup windows on production storage systems. Because backup operations are not occurring on the production systems, small backup windows are not as important (see Figure 1-4).

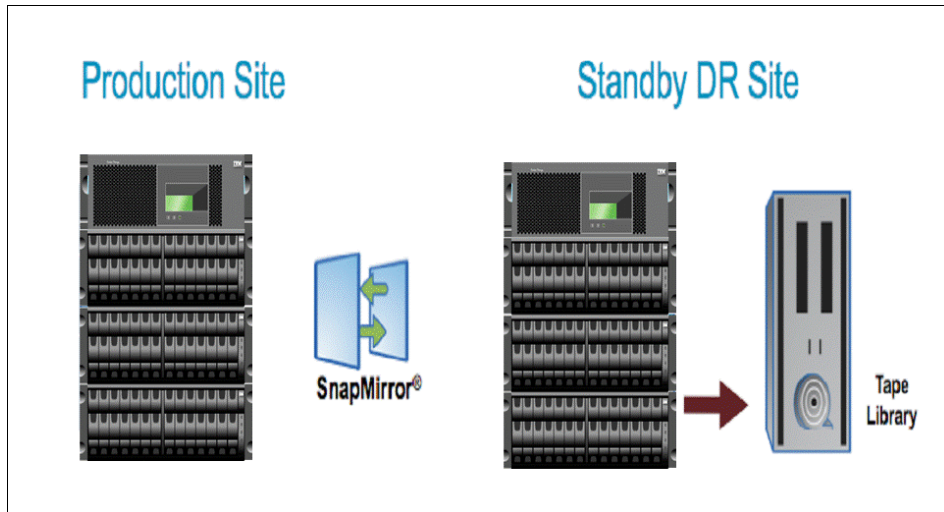


Figure 1-4 Tape Backup operations offloaded to the DR site

### 1.1.5 Load sharing

Load sharing is similar to the remote data access example described earlier in both implementation and benefit. The difference in a load sharing implementation lies in the distance between the source and target volumes of the replication as well as in the performance goals associated with the replication implementation. In load sharing, the goal is to minimize the contention for critical application or database server resources by moving all read-only activities off the critical “transaction” server to a “mirror” or read-only server. The benefit can be twofold:

- ▶ Optimize and partition network access to the data set.
- ▶ Reduce CPU contention on the source application server by providing read-only and reporting access to the mirrored data. FlexCache can also be used for the purpose of load sharing.

## 1.2 Benefits of SnapMirror

SnapMirror has many benefits for your protection; we list some here:

- ▶ Block-level updates reduce bandwidth and time requirements.
- ▶ Data consistency can be maintained at a DR site.
- ▶ A DR plan can be tested without affecting production and replication.



- ▶ A DR location can keep many Snapshot copies at once; data can be restored to a point in time before data corruption occurred.
- ▶ Data can be replicated between dissimilar N series storage systems.
- ▶ A standard IP or FC network can be used for replication.
- ▶ SnapMirror Async supports one-to-one, one-to-many, many-to-one, or many-to-many replication, referred to as cascading and multihop.
- ▶ Starting with Data ONTAP 7.3.2, volume SnapMirror offers native network compression functionality to reduce bandwidth costs.

## 1.3 Summary of SnapMirror changes in Data ONTAP 7.3

This section summarizes key changes in SnapMirror in Data ONTAP 7.3. For a complete list, see the release notes and the product documentation in the Data Protection Online Backup and Recovery Guide available on the IBM Storage support website:

<http://www.ibm.com/storage/support/nas>

### 1.3.1 Increased concurrent transfers

Data ONTAP 7.3 allows an increase in concurrent SnapMirror transfers for asynchronous modes. This applies to both volume and qtree SnapMirror. The increase depends on the platform. For example, N7800 allows 32 concurrent volume SnapMirror transfers in Data ONTAP 7.2 versus 150 concurrent volume SnapMirror transfers in Data ONTAP 7.3. Both of these numbers are without the NearStore option license.

In Data ONTAP 7.2, the concurrent transfers are halved when ATA disks are present in the system unless the NearStore option license is installed. For example, in Data ONTAP 7.2, N7800 allows 32 concurrent volume SnapMirror transfers with FC only drives and 16 concurrent volume SnapMirror transfers when ATA drives are present. Data ONTAP 7.3 removes this limitation. Therefore, an N7800 allows 150 volume SnapMirror concurrent streams regardless of the type of drive present in the system.

SnapMirror Sync and Semi-Sync do not have increased concurrent streams in Data ONTAP 7.3.

### 1.3.2 Efficient use of multiprocessor systems

SnapMirror Async in Data ONTAP 7.3 efficiently uses multiprocessor systems. This mitigates or eliminates domain bottleneck issues. SnapMirror Sync and SnapMirror Semi-Sync do not have these improvements in Data ONTAP 7.3

### 1.3.3 Efficient computation of changed blocks

A SnapMirror update involves computation of changed blocks prior to the transfer. This can take a long time for very large volumes. If the volume has minimal changes, the computation time can be longer than the transfer time.

SnapMirror in Data ONTAP 7.3 employs a more efficient comparison of the active map file to find the changed blocks. This results in significantly faster computation of changed blocks.





## Overview of SnapMirror

In this chapter we explain the basics of SnapMirror and the differences between volume SnapMirror and qtree SnapMirror. Also covered are Snapshot copies and how they are used with SnapMirror.

## 2.1 The basics

When mirroring asynchronously, SnapMirror replicates Snapshot copy images from a source volume or qtree to a partner destination volume or qtree, thus replicating source object data to destination objects at regular intervals. SnapMirror source volumes and qtrees are writable data objects whose data is to be replicated. The source volumes and qtrees are the objects that are normally visible, accessible, and writable by the storage system's clients.

The SnapMirror destination volumes and qtrees are read-only objects, usually located on a separate storage system, to which the source volumes and qtrees are replicated. Clients might want to use these read-only objects for auditing purposes before the objects are converted to writable objects. In addition, the read-only objects can be used for data verification. The more obvious use for the destination volumes and qtrees is to use them as true replicas for recovery from a disaster. In such a case, a disaster takes down the source volumes or qtrees, and the administrator uses SnapMirror commands to make the replicated data at the destination accessible and writable.

## 2.2 Control files

SnapMirror uses information in control files to maintain relationships and schedules. One of these control files, the `snapmirror.conf` file located on the destination system, allows scheduling to be maintained. This file, along with information entered by using the `snapmirror.access` option or the `snapmirror.allow` file is used to establish a relationship between a specified source volume, or qtree for replication, and the destination volume, or qtree where the mirror is kept.

### 2.2.1 `snapmirror.conf`

The `snapmirror.conf` file is located on the destination system. It contains the schedules of the SnapMirrors.

To access this file, on the command line, enter the following command:

```
rdfile /etc/snapmirror.conf
```

Figure 2-1 shows an example of the results of this command.

```
itsotuc4> rdfile /etc/snapmirror.conf
#Regenerated by registry Thu Feb 18 00:15:42 GMT 2010
9.11.218.114:itso_1 itsotuc4:itso_1 restart=always 0 1 * *
9.11.218.114:sangam itsotuc4:sangam - 0 2 * *
```

*Figure 2-1 Using `rdfile` to display `snapmirror.conf`*

**Tip:** The `snapmirror.conf` file is not required to establish relationships.

## 2.2.2 snapmirror.access

The snapmirror.access file is an option that can define which destination systems can access the source. To display the setting for snapmirror.access, on the command line, enter this command:

```
option snapmirror.access
```

Figure 2-2 shows an example of the output.

```
itsotuc1> options snapmirror.access
snapmirror.access          host=9.11.218.238,9.11.218.237
```

Figure 2-2 Displaying snapmirror.access setting

If snapmirror.access is set (see Figure 2-3) to *legacy* (the default), then the snapmirror.allow file defines which destinations can access the system.

```
itsotuc4*> options snapmirror.access
snapmirror.access          legacy
```

Figure 2-3 Displaying snapmirror.access default value

## 2.2.3 snapmirror.allow

The snapmirror.allow file (see Figure 2-4) is located on the source system. It defines which destination systems are allowed to copy from the source.

```
itsotuc1> rdfile /etc/snapmirror.allow
#Regenerated by registry Wed Aug 27 05:27:26 GMT 2008
9.11.202.150
itsotuc3
9.11.218.237
itsotuc4
9.11.218.238
```

Figure 2-4 Sample snapmirror.allow file

## 2.3 Update process

The SnapMirror update process performs the following tasks:

1. It creates a Snapshot copy of the data on the source volume.
2. It copies the data to the destination, a read-only volume or qtree on the same system or on a remote destination system.
3. It updates the destination file system to reflect incremental changes occurring to the source.

The result of this process is an online, read-only data set that is a point-in-time view of the data on the source at the time of the most recent update.

When using `snapmirror.conf`, the SnapMirror Snapshot copy creation and updates are controlled by a schedule that is local to the destination N series system. In a SAN environment, Snapshot copy creation involving logical unit numbers (LUNs) must be controlled by host systems. Scripts are set up to create Snapshot copies and to initiate the SnapMirror update to the remote storage system. For more information, see 7.1.2, “Server Suite” on page 61.

## 2.4 Snapshot copy behavior in SnapMirror

SnapMirror uses a Snapshot copy as a marker for a point in time for the replication process. A copy is kept on the source volume as the current point in time that both mirrors are in sync. When an update occurs, a new Snapshot copy is created and is compared against the previous Snapshot copy to determine the changes since the last update. SnapMirror marks the copies it needs to keep for a particular destination mirror in such a way that the `snap list` command (Example 2-1) displays the keyword **snapmirror** next to the necessary Snapshot copies. For more information, see 2.4, “Snapshot copy behavior in SnapMirror” on page 10 and 2.5, “Volume SnapMirror and qtree SnapMirror” on page 12.

*Example 2-1 Snapshot copies with one designated to be used for a SnapMirror*

```
itsotuc1> snap list
Volume root
working...
```

%/used	%/total	date	name
-----	-----	-----	-----
59% (59%)	2% ( 2%)	Aug 15 19:22	itsotuc4(0101181370)_vol4_dst.1 (snapmirror)
60% ( 7%)	2% ( 0%)	Apr 25 08:00	hourly.0
60% ( 0%)	2% ( 0%)	Apr 25 00:00	nightly.0
60% ( 0%)	2% ( 0%)	Apr 24 20:00	hourly.1
60% ( 0%)	2% ( 0%)	Apr 24 16:00	hourly.2
60% ( 0%)	2% ( 0%)	Apr 24 12:00	hourly.3
60% ( 0%)	2% ( 0%)	Apr 24 08:00	hourly.4
60% ( 0%)	2% ( 0%)	Apr 24 00:00	nightly.1
60% ( 0%)	2% ( 0%)	Apr 23 20:00	hourly.5

The **snapmirror destinations** command can be used to see which replica of a particular copy is marked as required at any time. On the source volume, SnapMirror creates the Snapshot copy for a particular destination and immediately marks it for that destination. At this point, both the previous copy and the new copy are marked for this destination. After a transfer is successfully completed, the mark for the previous copy is removed and deleted. Snapshot copies left for *cascade* mirrors from the destination also have the *snapmirror* tag in the `snap list` command output. (*Cascade mirrors* are a variation on the basic SnapMirror deployment, involving a writable source volume replicated to multiple read-only destinations, either one-to-one or one-to-many.)

Use the **snapmirror destinations -s** command to find out why a particular Snapshot copy is marked. This mark is kept as a reminder for SnapMirror to not delete a copy. This mark does not stop a user from deleting a copy marked for a destination that will no longer be a mirror; use the **snapmirror release** command to force a source to forget about a particular destination. This is a safe way to have SnapMirror remove its marks and clean up Snapshot copies that are no longer needed. Deleting a Snapshot copy that is marked as needed by SnapMirror is not advisable; you must do it with caution in order not to disallow a mirror from updating. While a transfer is in progress, SnapMirror uses the busy lock on a Snapshot copy.

This result can be seen in the **snap list** command output (see Figure 2-5). These locks do prevent users from deleting the Snapshot copy. The busy locks are removed when the transfer is complete.

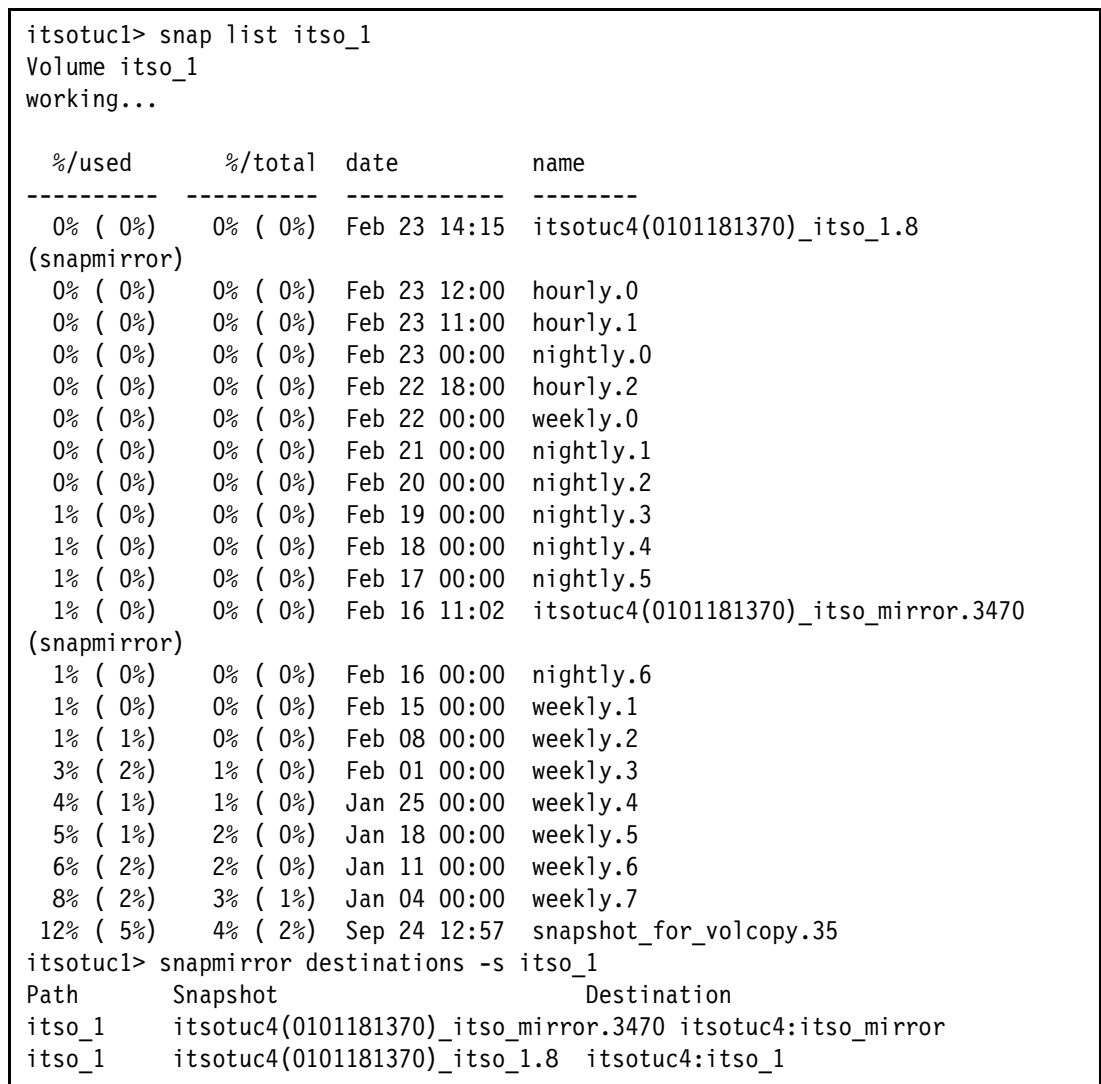


Figure 2-5 Identifying Snapshots used for SnapMirror

For volume replication, SnapMirror creates a Snapshot copy of the whole source volume that is copied to the destination volume. For qtree replication, SnapMirror creates Snapshot copies of one or more source volumes that contain qtrees identified for replication. This data is copied to a qtree on the destination volume and a Snapshot copy of that destination volume is created.

A volume SnapMirror Snapshot copy name has the following format:

dest\_name(sysid)\_name.number

Example: fasA(0050409813)\_vol1.6 (snapmirror)

- dest\_name** is the host name of the destination storage system.
- sysid** is the destination system ID number.
- name** is the name of the destination volume.

**number** is the number of successful transfers for the Snapshot copy, starting at 1. Data ONTAP increments this number for each transfer.

A qtree SnapMirror Snapshot copy name has the following format:

`dest_name(sysid)_name-src|dst.number`

Example: `fasA(0050409813)_vol1_qtree3-dst.15 (snapmirror)`

**dest\_name** is the host name of the destination storage system.

**sysid** is the destination system ID number.

**name** is the name of the destination volume or qtree path.

**src|dst** identifies the Snapshot copy location.

**number** is an arbitrary start point number for the Snapshot copy. Data ONTAP increments this number for each transfer.

In the output of the **snap list** command, Snapshot copies needed by SnapMirror are followed by the SnapMirror name in parentheses.

**Attention:** Deleting Snapshot copies marked **snapmirror** can cause SnapMirror updates to fail.

## 2.5 Volume SnapMirror and qtree SnapMirror

SnapMirror software provides the ability to replicate individual qtrees as well as whole volumes. The two types of replication are physical and logical. There are tradeoffs, including performance, manageability, configuration, and infrastructure resources. A comparison of the two is necessary to understand their implications.

## 2.6 SnapMirror volume replication

This section discusses considerations regarding volume SnapMirror.

### 2.6.1 Characteristics of volume SnapMirror

Volume SnapMirror has the following characteristics:

- ▶ SnapMirror volume replication can be synchronous or asynchronous.
- ▶ SnapMirror volume replication can occur only with volumes of the same type; that is, both volumes are traditional volumes or both are flexible volumes.
- ▶ SnapMirror volume replication copies a volume and all of its Snapshot copies to a destination volume.
- ▶ A destination volume that is set up for SnapMirror volume replication must first be set to restricted, read-only status.
- ▶ The destination volume (entire volume) is read-only unless it is made writable.
- ▶ SnapMirror volume replication is block-for-block replication; it transfers the file system verbatim. Therefore, earlier major releases of Data ONTAP cannot understand file system transfers from a later major release. Data ONTAP 7.2 and 7.3 and 8.0 are examples of the three different major release versions. Data ONTAP 7.3.3 and 7.3.5 are examples of the same major release but different minor releases (Table 2-1).



Table 2-1 Volume SnapMirror version restrictions

Volume SnapMirror source	Volume SnapMirror destination	Replication possible?
Data ONTAP 7.2	Data ONTAP 7.3	Yes
Data ONTAP 7.3	Data ONTAP 7.2	No
Data ONTAP 7.3.x	Data ONTAP 7.3.y	Yes
Data ONTAP 8.0.x	Data ONTAP 8.0.y	Yes

You can use volume-based SnapMirror to replicate data to a newer major release to assist in migrating to a newer Data ONTAP version. However, you cannot do this in the reverse direction.

Everything contained in a volume is replicated from one system or location to another, including metadata about the volume itself, such as language translation settings and other volume options stored as part of the volume, as well as all Snapshot copies of the volume.

With flexible volumes, volume replication can be as granular as traditional deployments of qtree-based replication. The entire volume is replicated and can be very large, but it can also be very small and used in the same way that a qtree is used in traditional deployments.

SnapMirror creates a Snapshot copy before performing the initial transfer. This copy is referred to as the baseline Snapshot copy. After performing an initial transfer of all data in the volume, volume SnapMirror sends to the destination only the blocks that have changed since the last successful replication. When SnapMirror performs an update transfer, it creates another new Snapshot copy and compares the changed blocks, which are sent as part of the update transfer.

## 2.6.2 Snapshot copy behavior and status in volume SnapMirror

Table 2-2 shows how Snapshot copies are replicated between the volume SnapMirror source and destination systems and also the state of the Snapshot copies on both source and destination systems. The example assumes that fas1 is the source storage system and vol1 is the source volume; and that fas2 is the destination storage system and vol2 is the destination volume.

Table 2-2 Snapshot copies on source and destination for volume SnapMirror

Timeline	Snapshot copies on itsotuc1	Snapshot copies on itsotuc4
After volume initialization first	itsotuc4(0101181370)_itso2_sm.1 (snapmirror)	itsotuc4(0101181370)_itso2_sm.1
After first update of itso2	itsotuc4(0101181370)_itso2_sm.2 (snapmirror)	itsotuc4(0101181370)_itso2_sm.2 itsotuc4(0101181370)_itso2_sm.1
Create a Snapshot called demo on itsotuc1:itso2	demo itsotuc4(0101181370)_itso2_sm.2 (snapmirror)	itsotuc4(0101181370)_itso2_sm.2 itsotuc4(0101181370)_itso2_sm.1
Second update of itso2	itsotuc4(0101181370)_itso2_sm.3 (snapmirror) demo	itsotuc4(0101181370)_itso2_sm.3 demo itsotuc4(0101181370)_itso2_sm.2

Timeline	Snapshot copies on itsotuc1	Snapshot copies on itsotuc4
Delete the Snapshot called demo on itsotuc1:itso2	itsotuc4(0101181370)_itso2_sm.3 (snapmirror)	itsotuc4(0101181370)_itso2_sm.3 demo itsotuc4(0101181370)_itso2_sm.2
Third update of itso2	itsotuc4(0101181370)_itso2_sm.4 (snapmirror)	itsotuc4(0101181370)_itso2_sm.4 itsotuc4(0101181370)_itso2_sm.3

The snapmirror tag next to the Snapshot copy indicates a soft lock created by SnapMirror. Data ONTAP does not delete the Snapshot copies with soft locks but a user is able to delete these types of Snapshot copies. As just shown, in case of volume SnapMirror, if a Snapshot copy is deleted on the volume SnapMirror source, it is deleted on the destination at the next update. If a Snapshot copy is created on the volume SnapMirror source, it is created on the destination at the next update. This is not the case for qtree SnapMirror; see the next section.

## 2.7 SnapMirror qtree replication

SnapMirror qtree replication has the following characteristics:

- ▶ SnapMirror qtree replication is available only in asynchronous mode.
- ▶ SnapMirror qtree replication occurs between qtrees regardless of the type of volume (traditional or flexible) in which the qtree resides.
- ▶ A destination qtree is read-only, but the volume on which it is located must be online and writable.
- ▶ SnapMirror qtree replication is logical replication; all of the files and directories in the source file system are created in the destination file system. Therefore, replication can occur between any Data ONTAP releases.
- ▶ To replicate qtrees, qtree SnapMirror can either first create a Snapshot copy on the source volume that contains the qtree to be replicated or can also use an existing Snapshot copy on the source volume by specifying the `-s` flag.

**Attention:** Snapshot technology always operates on volumes, not on qtrees. This Snapshot copy contains a point-in-time copy of all of the data on the source volume, including both the data in the qtree to be replicated and also (presumably) other data that is not to be replicated.

- ▶ Qtree SnapMirror determines changed data by first looking through the inode file for inodes that have changed and changed inodes of the interesting qtree for changed data blocks. The SnapMirror software then transfers only the new or changed data blocks from this Snapshot copy that is associated with the designated qtree. On the destination volume, a new Snapshot copy is then created that contains a complete point-in-time copy of the entire destination volume, but that is associated specifically with the particular qtree that has been replicated.

**Important:** If the source file system contains a file type that cannot be represented on the destination file system, the replication fails. For example, Data ONTAP 7.0 supports files up to 16 TB in size, whereas earlier Data ONTAP versions support files up to 4 TB. If the source storage system is running Data ONTAP 7.0, the qtree that you want to replicate contains a file greater than 4 TB, and the destination storage system is running an earlier version of Data ONTAP, the replication fails.

## 2.7.1 Snapshot copy behavior and status in qtree SnapMirror

Table 2-3 shows how Snapshot copies are replicated between qtree SnapMirror source and destination systems and also the state of the Snapshot copies on both source and destination systems. The example assumes that fas1 is the source storage system and qt1 is the source qtree in volume vol1; and that fas2 is the destination storage system and qt2 is the destination qtree in volume vol2.

Table 2-3 Snapshot copies for source and destination for qtree SnapMirror

Timeline	Snapshot copies on fas1	Snapshot copies on fas2
After qtree initialization	fas2(0099909262)_vol2_qt2-src.0 (snapmirror)	fas2(0099909262)_vol2_qt2dst.2 (busy,snapmirror)
After first update of qt2	as2(0099909262)_vol2_qt2-src.1 (snapmirror)	fas2(0099909262)_vol2_qt2dst.4 (busy,snapmirror)
Create a Snapshot called demo on fas1:vol1	demo fas2(0099909262)_vol2_qt2-src.1 (snapmirror)	fas2(0099909262)_vol2_qt2dst.4 (busy,snapmirror)
Second update of vol2	fas2(0099909262)_vol2_qt2-src.2 (snapmirror) demo	fas2(0099909262)_vol2_qt2dst.6 (busy,snapmirror)
Delete the Snapshot called demo on fas1:vol1	fas2(0099909262)_vol2_qt2-src.2 (snapmirror)	fas2(0099909262)_vol2_qt2dst.6 (busy,snapmirror)
Third update of vol2	fas2(0099909262)_vol2_qt2-src.3 (snapmirror)	fas2(0099909262)_vol2_qt2dst.8 (busy,snapmirror)

The snapmirror tag next to the Snapshot copy indicates a soft lock created by SnapMirror. The busy,snapmirror tag indicates a hard lock created by SnapMirror. You cannot delete a Snapshot copy with a hard lock. As seen in Table 2-3, in case of qtree SnapMirror, the same Snapshot copies do not exist on both source and destination systems.

## 2.7.2 Key differences between volume and qtree SnapMirror

Here we list the differences between volume and qtree SnapMirror (note that these differences are not listed in order of importance).

**Tip:** Both volume and qtree SnapMirror can operate over Ethernet and Fiber Channel or a combination of both. For more information, see 2.11, “Multipath support” on page 22.

Table 4 Qtree SnapMirror versus volume SnapMirror

Qtree SnapMirror	Volume SnapMirror
Unaffected by disk size or disk checksum differences between the source and destination irrespective of type of volumes used (traditional or flexible).	Unaffected by disk size or disk checksum differences between the source and destination if flexible volumes are used. Affected by disk size or disk checksum differences between the source and destination if traditional volumes are used.
Destination volume must have free space available equal to approximately 105% of the data being replicated.	Destination volume must be equal or larger than the source volume.

Qtree SnapMirror	Volume SnapMirror
Sensitive to the number of files in a qtree due to the nature of the qtree replication process. The initial phase of scanning the inode file might be longer with a larger number (tens of millions) of files.	Not sensitive to the number of files in a volume.
Qtree SnapMirror destinations can be placed on the root volume of the destination storage system.	The root volume cannot be used as a destination for volume SnapMirror.
Replicates only one Snapshot copy of the source volume where the qtree resides (the copy created by the SnapMirror software at the time of the transfer) to the destination qtree. Therefore, qtree SnapMirror allows independent Snapshot copies on the source and destination.	Replicates all Snapshot copies on the source volume to the destination volume. Similarly, if a Snapshot copy is deleted on the source system, volume SnapMirror deletes the Snapshot copy at the next update. Therefore, volume SnapMirror is typically best for disaster recovery scenarios, because the same data exists on both source and destination. Note that the volume SnapMirror destination always keeps an extra SnapMirror Snapshot copy.
A qtree SnapMirror destination volume might contain replicated qtrees from multiple source volumes on one or more systems and might also contain qtrees or non-qtree data not managed by SnapMirror software.	A volume SnapMirror destination volume is always a replica of a single source volume.
Multiple relationships would have to be created to replicate all qtrees in a given volume by using qtree-based replication.	Volume-based replication can take care of this in one relationship (as long as the one volume contains all relevant qtrees).
For low-bandwidth wide area networks, qtree SnapMirror can be initialized using the LREP tool. See Chapter 5, "Network-free seeding" on page 51 for more information.	Volume SnapMirror can be initialized using a tape device (SnapMirror to Tape) by using the <b>snapmirror store</b> and <b>snapmirror retrieve</b> commands. See Chapter 5, "Network-free seeding" on page 51 for more information.
Qtree SnapMirror can only occur in a single hop. Cascading of mirrors (replicating from a qtree SnapMirror destination to another qtree SnapMirror source) is not supported.	Cascading of mirrors is supported for volume SnapMirror.
Qtree SnapMirror updates are not affected by backup operations. This allows a strategy called <i>continuous backup</i> , in which traditional backup windows are eliminated and tape library investments are fully used. SnapVault software, discussed later in this report, is optimized for continuous backup applications.	Volume SnapMirror updates can occur concurrently with a dump operation of the destination volume to tape by using the dump command or NDMP-based backup tools. However, if the volume SnapMirror update involves a deletion of the Snapshot copy that the dump operation is currently writing to tape, the SnapMirror update will be delayed until the dump operation is complete.
The latest Snapshot copy is used by qtree SnapMirror for future updates if the <b>-s</b> flag is not used.	Volume SnapMirror can use any common Snapshot copy for future updates.
Qtrees in source deduplicated volumes that are replicated with qtree SnapMirror are full size at the destination.	Source deduplicated volumes that are replicated with volume SnapMirror remain deduplicated at the destination.

Qtree SnapMirror	Volume SnapMirror
Even though the source volume is deduplicated, qtree SnapMirror will expand the data and send the entire data to the destination.	Deduplication savings also extend to the bandwidth savings because volume SnapMirror only transfers unique blocks.
Source and destination volumes can be independently deduplicated.	Destination volume is read-only and therefore destination volume cannot be independently deduplicated. If deduplication savings are desired on the destination volume, then the source volume must be deduplicated.
The files in the file system gain new identity (inode numbers, and so on) in the destination system. Therefore, file handles cannot be migrated to the destination system.	The files in the file system have the same identity on both source and destination system.
LUN clones can be created on the destination volume, but not in the destination qtree.	LUN clones cannot be created on the destination volume because the volume is read-only. However, LUN clones can be created on a FlexClone volume because the FlexClone volume is writable.

The decision as to which to use depends on individual site requirements. Volume SnapMirror and qtree SnapMirror can be freely mixed on both source and destination systems, although any individual destination volume can be a destination for only one or the other.

## 2.8 Support for volume types

Table 2-5 shows the support for SnapMirror replication between the two volume types.

*Table 2-5 Volume replication support*

Replication	Volume SnapMirror	Qtree SnapMirror
TradVol <--> TradVol	Yes <sup>a</sup>	Yes
TradVol <--> FlexVol	No	Yes
FlexVol <--> FlexVol	Yes <sup>a</sup>	Yes

a. Volume SnapMirror requires the destination system's Data ONTAP version to be the same as or higher than that of the source system.

## 2.9 Modes of SnapMirror

SnapMirror can be used in three different modes: SnapMirror Async, SnapMirror Sync, and SnapMirror Semi-Sync.

### 2.9.1 SnapMirror Async

SnapMirror Async can operate on both qtrees and volumes. In this mode, SnapMirror performs incremental, block-based replication as frequently as once per minute.

The first and most important step in this mode involves the creation of a one-time, baseline transfer of the entire data set. This is required before incremental updates can be performed. This operation proceeds as follows:

1. The source storage system creates a Snapshot copy (a read-only, point-in-time image of the file system). This copy is called the baseline copy.
2. All data blocks referenced by this Snapshot copy and any previous copies are transferred in case of volume SnapMirror and written to the destination file system. Qtree SnapMirror only copies the latest Snapshot copy.
3. After the initialization is complete, the source and destination file systems have at least one Snapshot copy in common.

After the initialization is complete, scheduled or manually triggered updates can occur. Each update transfers only the new and changed blocks from the source to the destination file system. This operation proceeds as follows:

1. The source storage system creates a Snapshot copy.
2. The new copy is compared to the baseline copy to determine which blocks have changed.
3. The changed blocks are sent to the destination and written to the file system.
4. After the update is complete, both file systems have the new Snapshot copy, which becomes the baseline copy for the next update.

Because asynchronous replication is periodic, SnapMirror Async is able to consolidate the changed blocks and conserve network bandwidth. There is minimal impact on write throughput and write latency.

## 2.9.2 SnapMirror Sync

Certain environments have very strict uptime requirements. All data that is written to one site must be mirrored to a remote site or system synchronously. SnapMirror Sync mode is a mode of replication that sends updates from the source to the destination as they occur, rather than according to a predetermined schedule. This helps make sure that data written on the source system is protected on the destination even if the entire source system fails. SnapMirror Semi-Sync mode, which minimizes data loss in a disaster while also minimizing the extent to which replication affects the performance of the source system, is also provided.

No additional license fees need to be paid to use this feature, although a free special license *snapmirror\_sync* must be installed; the only requirements are appropriate hardware, the correct version of Data ONTAP, and a SnapMirror license for each storage system. Unlike SnapMirror Async mode, which can replicate volumes or qtrees, SnapMirror Sync and Semi-Sync modes work only with volumes. SnapMirror Sync can have a significant performance impact and is not necessary or appropriate for all applications.

To check what has been licensed on your system, use the **license** command (see Figure 2-6):

```
license
```

```

itsotuc1> license
      a_sis XXXXXXX
      cifs site XXXXXXX
      cluster not licensed
      cluster_remote not licensed
      compression not licensed
      disk_sanitization site XXXXXXX
      fcp site XXXXXXX
      flex_clone XXXXXXX
      flex_scale not licensed
      flexcache_nfs not licensed
      gateway not licensed
      gateway_hitachi not licensed
      http not licensed
      iscsi XXXXXXX
      multistore site XXXXXXX
      nearstore_option XXXXXXX
      nfs site XXXXXXX
      operations_manager not licensed
      pamii not licensed
      protection_manager not licensed
      provisioning_manager not licensed
      smdomino not licensed
      smsql not licensed
      snapdrive_unix not licensed
      snapdrive_windows not licensed
      snaplock XXXXXXX
      snaplock_enterprise not licensed
      snapmanager_hyperv not licensed
      snapmanager_oracle not licensed
      snapmanager_sap not licensed
      snapmanager_sharepoint not licensed
      snapmanagerexchange not licensed
      snapmirror site XXXXXXX
      snapmirror_sync not licensed
      snapmover not licensed
      snaprestore site XXXXXXX
      snapvalidator not licensed

```

*Figure 2-6 Sample output of the license command*

The first step in synchronous replication is a one-time baseline transfer of the entire data set. After the baseline transfer is completed, SnapMirror will transition into synchronous mode with the help of NVLOG and CP forwarding.

After SnapMirror has transitioned into synchronous mode, the output of a SnapMirror status query shows that the relationship is “In-Sync.”

### 2.9.3 SnapMirror Semi-Sync

SnapMirror provides a semisynchronous mode, also called SnapMirror Semi-Sync. This mode differs from the synchronous mode in two key ways:

- ▶ User writes do not need to wait for the secondary or destination storage to acknowledge the write before continuing with the transaction. User writes are acknowledged immediately after they are committed to the primary or source system's memory.
- ▶ NVLOG forwarding is not used in semisynchronous mode. Therefore, SnapMirror Semi-Sync might offer faster application response times. This mode makes a reasonable compromise between performance and Recovery Point Objective (RPO) for many applications.

Before Data ONTAP 7.3, SnapMirror Semi-Sync was tunable, so that the destination system could be configured to lag behind the source system by a user-defined number of write operations or seconds. This was configurable by specifying a variable called *outstanding* in the SnapMirror configuration file. Starting in Data ONTAP 7.3, the outstanding parameter functionality is removed and there is a new mode called semi-sync. When using semi-sync mode, only the consistency points are synchronized. Therefore, this mode is also referred to as CP Sync mode.

Configuration of semi-synchronous mode is very similar to that of synchronous mode; simply replace *sync* with *semi-sync*, as in the following example:

```
fas1:vol1 fas2:vol1 - semi-sync
```

### 2.9.4 Visibility interval

The visibility interval specifies how often the source system takes a Snapshot copy in SnapMirror Sync and Semi-Sync modes. The default interval is 3 minutes. Because the same Snapshot copies exist on both source and destination system, this means that updated data in the file system is visible on the SnapMirror destination system in 3-minute increments. This generates more Snapshot creations and deletions, so if this value is small, a performance impact might be seen on the source volume. It is best to use the default value, unless a different value is necessary. This value is set with an option in the `/etc/snapmirror.conf` file and can be set on an individual volume basis.

## 2.10 Configuration files

The option `snapmirror.access` along with the control files `snapmirror.allow` and `snapmirror.conf` are used to configure SnapMirror.

### 2.10.1 Access and security

There are various security and access options and settings for the N series. We cover several of them in this section.

#### **snapmirror.access**

The `snapmirror.access` option specifies which SnapMirror destination storage systems can initiate transfers and which network interfaces they can use. This is the preferred method for controlling SnapMirror access on a SnapMirror source storage system.



On the source storage system console, use the **options snapmirror.access** command to specify the host names of storage systems that are allowed to copy data directly from the source storage system. For example:

```
options snapmirror.access host=fas2
```

The syntax for specifying which storage systems are allowed access to the server is the same for SNMP, telnet, and rsh.

**Tip:** If you set the snapmirror.access option to *legacy*, the snapmirror.allow file is used instead.

### **/etc/snapmirror.allow**

You can generate a snapmirror.allow file in the /etc directory on the source storage system. The /etc/snapmirror.allow file specifies the host names of storage systems that are allowed to copy data directly from the source storage system. For more information about the options command, see the product documentation available at the IBM Storage support website:

<http://www.ibm.com/storage/support>

### **/etc/snapmirror.conf**

This is the core configuration file for all SnapMirror operations. The /etc/snapmirror.conf file defines the relationship between the source and the destination, the schedule used by the destination to copy data, and the arguments that control SnapMirror when copying data. This file resides on the SnapMirror destination system.

## **2.10.2 Distribution**

You can create a single /etc/snapmirror.conf file for your site and copy it to all the storage systems that use SnapMirror. This file can contain entries pertaining to other storage systems. For example, the /etc/snapmirror.conf file on fas2 can contain an entry for copying a volume from fas3 to fas4. When fas2 reads the /etc/snapmirror.conf file, it ignores the entries for other storage systems. This relationship between fas3 and fas4 is considered invalid and therefore ignored. However, each time the file is read, a warning message is displayed on the system console for each line that is ignored.

There is no limit to the total number of entries in the /etc/snapmirror.conf file; however, there is a limit of 1024 *valid* relationships in the file. Entries beyond the entry limit for each storage system are ignored, and a warning message is displayed on the system console.

In an active-active configuration, the limit on the maximum number of entries applies to the storage system pair combination. If one controller in an active-active configuration fails, the limit stays at 1024 entries.

**Attention:** This limitation is different from the maximum number of simultaneous (or concurrent) replications you can have on a storage system. For that information, see 3.10, “Concurrent replication operations” on page 34 or the Data Protection Online Backup and Recovery Guide available on the IBM Storage support website:

<http://www.ibm.com/storage/support>

### 2.10.3 Configuration changes

If SnapMirror is enabled, changes to the `/etc/snapmirror.conf` file take effect within 2 minutes. If SnapMirror is not enabled, changes to the `/etc/snapmirror.conf` file take effect immediately after you enter the `snapmirror on` command to enable SnapMirror.

## 2.11 Multipath support

More than one physical path between a source and a destination system might be desired for a mirror relationship. SnapMirror Async (volume and qtree), SnapMirror Sync, and SnapMirror Semi-Sync support multiple paths for replication. Multipath support allows SnapMirror traffic to be load balanced between these paths and provides for failover (see Figure 2-7) in the event of a network outage.

Specifically, SnapMirror supports up to two paths for a particular relationship. Therefore, each replication relationship can be configured to use a distinct multipath connection. These multipath connections can be Ethernet, Fibre Channel, or a combination of the two. There are two modes of multipath operation.

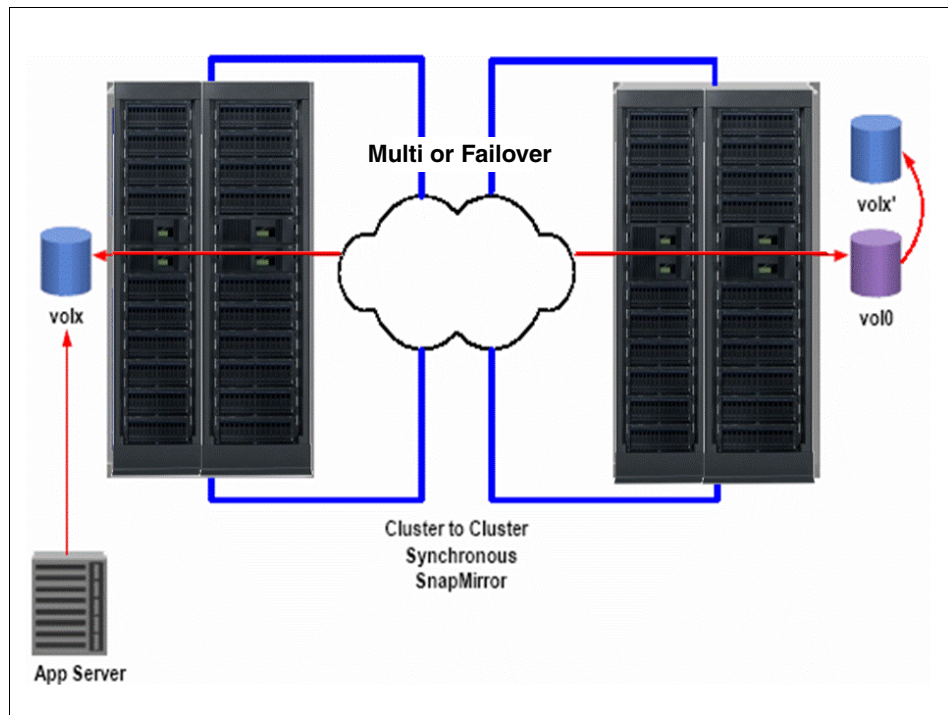


Figure 2-7 Multipath failover

### 2.11.1 Multiplexing mode

Both paths are used simultaneously, load-balancing transfers across the two. When a failure occurs, the load from both transfers moves to the remaining path.

Here is an example of an `/etc/snapmirror.conf` file:

```
pumpkin=multi(fas1_e0a, fas2_e0a)(fas1_e0b, fas2_e0b)
pumpkin:src_vol fas2:dst_vol - 0 * * *
```

## 2.11.2 Failover mode

One path is specified as the primary path in the configuration file. This path is the desired path. In case of failure, the second path is used.

Here is an example of an `/etc/snapmirror.conf` file:

```
pumpkin=failover(fas1_e0a, fas2_e0a)(fas1_e0b,fas2_e0b)
pumpkin:src_vol fas2:dst_vol - 0 * * *
```

**Tip:** It is best to use multipath to improve availability of the replication network.

## 2.11.3 SnapMirror network compression

With increasing network bandwidth costs coupled with data growth, clients are having to do more with less. As the amount of data to be protected increases, more network bandwidth is needed to maintain the recovery point objective (RPO) or the replication window. Otherwise, replication times increase as the amount of data sent over the network to the DR site increases. Differently put, if you do not want to or cannot increase the network bandwidth, you need to lower the replication frequency that is causing larger RPO values, increasing your exposure to larger data loss.

## 2.11.4 Network bandwidth versus RPO

The SnapMirror native network compression feature can cut down on the amount of data replicated over the network. It also offers you more flexibility and choices, as described next.

### Maintaining the same RPO level

**Problem:** Your data replication needs are growing. You need more bandwidth to maintain the same level of RPO.

**Solution:** By using network compression, it is possible to maintain the same RPO without purchasing additional network bandwidth.

### Improving your RPO without buying additional bandwidth

**Problem:** You are using all of your network bandwidth. However, you want to reduce your exposure to data loss—in other words, to improve your RPO.

**Solution:** By using network compression, you can improve your RPO without purchasing additional network bandwidth.

### Using the network bandwidth for other purposes

**Problem:** Your replication is consuming all of your bandwidth. You want to use the network bandwidth for other purposes, such as client access or applications without purchasing additional bandwidth.

**Solution:** By using network compression, it is possible to reduce the bandwidth consumed by SnapMirror without sacrificing RPO, thereby freeing up network bandwidth for other purposes.

### Speeding up the initial transfers

**Problem:** Initial SnapMirror transfers could be large and therefore could take a long time to complete under bandwidth constraints.

**Solution:** By using network compression, it is possible to speed up the initial SnapMirror transfers.

## 2.11.5 What is SnapMirror network compression

SnapMirror network compression enables data compression over the network for SnapMirror transfers. This is a native feature that is built into SnapMirror software. SnapMirror network compression is not the same as WAFL compression, and it does not compress data at rest. Figure 2-8 shows a very high level flow of SnapMirror network compression.

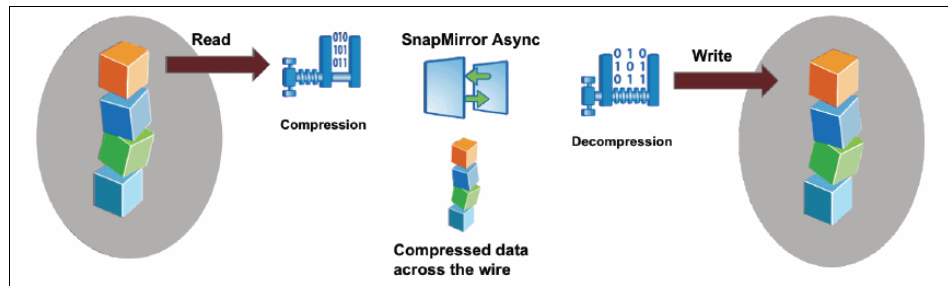


Figure 2-8 SnapMirror network compression functional diagram

On the source system, the data blocks that need to be sent to the destination system are handed off to the compression engine, which compresses the data blocks. The compression engine on the source system creates multiple threads depending on the number of processors available on the storage system. The multiple compression threads help to compress data in a parallel fashion. The compressed blocks are then sent over the network.

On the destination system, the compressed blocks are received over the network and are then decompressed. The destination compression engine also has multiple threads to decompress the data in a parallel fashion. The decompressed data is reordered and is saved to the disk on the appropriate volume.

In other words, when SnapMirror network compression is enabled, two additional steps are performed: Compression processing on the source system before data is sent over the network, and decompression processing on the destination system before the data is written to the disk.

SnapMirror network compression uses the standard gzip compression algorithm to compress the data blocks. The compression and the decompression engines are capable of using all free CPU cycles.

## 2.11.6 Prerequisites for SnapMirror network compression

SnapMirror network compression is supported only for the asynchronous mode of SnapMirror. The required version to run SnapMirror network compression is Data ONTAP 7.3.2 on both source and destination systems. All platforms that support Data ONTAP 7.3.2 also support SnapMirror network compression. With Data ONTAP 7.3.2, volume SnapMirror is supported without any special approvals. Qtree SnapMirror still requires IBM approval.

SnapMirror network compression is currently not supported for the synchronous and semi-synchronous modes of SnapMirror.

SnapMirror network compression requires the use of the `/etc/snapmirror.conf` file. Special configuration is required to enable compression. For more information, see 3.15, “Network compression configuration and operation configuration” on page 38.



## Operational behaviors

When evaluating your SnapMirror implementation, it is important to consider the following common SnapMirror behaviors and to understand when and why they might occur.

## 3.1 Active-Active configuration

The SnapMirror product complements active-active configuration technology by providing an additional level of recoverability. If a catastrophe disables access to an active-active pair of storage systems, one or more SnapMirror volumes can be immediately accessed in read-only mode while recovery takes place. If read-write access is required, the mirrored volume can be converted to a writable volume while the recovery takes place. If SnapMirror is actively updating data when a takeover or giveback operation is run, the update aborts leaving the destination volume in the state of the last completed update. After the takeover or giveback operation is completed, SnapMirror transfer continues as before from a restart checkpoint.

No specific additional steps are required for the implementation of SnapMirror in an active-active configuration environment. For more information about active-active configuration technology and takeover and giveback scenarios, see the Data ONTAP System Administrator's Guide, available on the IBM Storage support website at:

<http://www.ibm.com/storage/support>

## 3.2 Disk geometry

In the case of traditional volumes, volume SnapMirror performance was affected due to disk geometry. If the source disks were not the same size as the destination disks, problems occurred that resulted in data not being properly distributed across some spindles. For example, data cleanly striped across three 5-GB drives on the source that is replicated to a destination system with 15-GB disks would result in the data being laid out on one of the destination system spindles. Qtree SnapMirror does not have this performance issue.

Flexible volumes in Data ONTAP 7G eliminated the performance impact due to geometry mismatch for volume SnapMirror as well. Destination volumes no longer have to contain the same number of disks or the same size disks as the source volumes, allowing more efficient deployment of resources. With flexible volumes, SnapMirror is no longer bound by the physical limitations of copying physical disks block for block. The physical nature of volume SnapMirror has been virtualized.

The size of a flexible volume can also act as a hard quota for a group or project assigned to it. In each volume, user- and group-level quotas as well as qtrees can be used to obtain finer granularity of quota management.

## 3.3 Cascading

A variation on the basic SnapMirror deployment and function involves mirroring from established mirrors to more SnapMirror destinations. Figure 3-1 shows a sample cascade configuration with two hops.

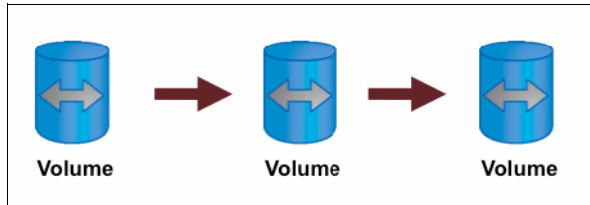


Figure 3-1 Cascading mirrors

The function of this deployment is to make a uniform set of data available on a read-only basis to users from various locations throughout a network, and to allow updating that data uniformly at regular intervals. However, cascading a SnapMirror replication from A to B to C and so on is allowed only with volume SnapMirror. Qtree SnapMirror does not support replication for more than one hop. During qtree SnapMirror replication, mapping takes place between source inodes and destination inodes.

### 3.3.1 Example of cascading

For example, suppose that the `/vol/vol1/qt7/user/email.txt` file has the inode number 456. When this qtree is transferred to the destination by using qtree SnapMirror (such as `vol1_rpl`), the `/vol/vol1_rpl/qt7_rpl/user/email.txt` file might have the inode number 5987432.

To be able to apply a modification on number 456 to number 5987432, qtree SnapMirror needs to keep a map of the inodes. Mapping the inodes is necessary because qtree SnapMirror is taking qtrees from different volumes and mirroring them into one common volume. Files from those qtrees might have the same inode number (because they come from different volumes or storage systems). Therefore, qtree SnapMirror reallocates the inodes, so that it does not have problems with conflicts in numbering. In addition, this inode mapping would cause problems because mapping the state can become confusing in a cascade, so this configuration is not allowed in a cascade configuration (see Table 3-1).

**Tip:** SnapMirror Sync and Semi-Sync cannot be cascaded. This means that you cannot configure multiple hops of SnapMirror Sync and Semi-Sync. However, SnapMirror Async (volume SnapMirror) can be cascaded from a SnapMirror Sync or Semi-Sync destination.

Table 3-1 Cascading support

Cascade configuration	Support
Sync/Semi-Sync → Volume SnapMirror	Yes
Sync/Semi-Sync → Sync/Semi-Sync	No
Sync/Semi-Sync → Qtree SnapMirror	No
Volume SnapMirror → Volume SnapMirror	Yes
Volume SnapMirror → Sync/Semi-Sync	No
Volume SnapMirror → Qtree SnapMirror	Yes
Qtree SnapMirror → Sync/Semi-Sync	No
Qtree SnapMirror → Volume SnapMirror	Yes
Qtree SnapMirror → Qtree SnapMirror	No

### 3.3.2 Snapshot copy propagation in a cascade configuration

This section demonstrates Snapshot copy propagation behavior in the following cascade configuration scenarios with examples.

#### Single hop volume SnapMirror

This configuration involves volume SnapMirror replication between two systems, filerA and filerB (see Table 3-2).

filerA:v2 → filerB:v2

Table 3-2 Snapshot copy propagation for single hop volume SnapMirror

Timeline	Snapshot copies on filerA	Snapshot copies on filerB
After volume initialization first	filerb(0101181370)_itso2_sm.1 (snapmirror)	filerb(0101181370)_itso2_sm.1
After first update of itso2	filerb(0101181370)_itso2_sm.2 (snapmirror)	filerb(0101181370)_itso2_sm.2 filerb(0101181370)_itso2_sm.1

**Tip:** SnapMirror creates a soft lock on the Snapshot copy of the source volume (snapmirror tag). The destination system carries an extra Snapshot copy.

#### Dual hop volume SnapMirror

This configuration involves volume SnapMirror replication among three systems, fas1, fas2, and fas3 (see Table 3-3).

fas1:v2 → fas2:v2 → fas3:v2

**Tip:** In the foregoing configuration, fas1:v2 to fas2:v2 and fas2:v2 to fas3:v2 transfers cannot occur at the same time.

Table 3-3 Snapshot copy propagation for dual hop volume SnapMirror

Timeline	Snapshot copies on fas1	Snapshot copies on fas2	Snapshot copies on fas3
1. After volume initialization on fas2	fas2(0099909262)_v2.1 (snapmirror)	fas2(0099909262)_v2.1	
2. Volume SnapMirror update on fas2 <sup>a</sup>	fas2(0099909262)_v2.2 (snapmirror)	fas2(0099909262)_v2.2 fas2(0099909262)_v2.1	
3. fas3:v2 initialization <sup>b</sup>	fas2(0099909262)_v2.2 (snapmirror)	fas2(0099909262)_v2.2 (snapmirror) fas2(0099909262)_v2.1	fas2(0099909262)_v2.2 fas2(0099909262)_v2.1
4. Volume SnapMirror update on fas2 <sup>c</sup>	fas2(0099909262)_v2.3 (snapmirror) fas2(0099909262)_v2.2 (snapmirror)	fas2(0099909262)_v2.3 fas2(0099909262)_v2.2 (snapmirror)	fas2(0099909262)_v2.2 fas2(0099909262)_v2.1
5. Volume SnapMirror update on fas3 <sup>d</sup>	fas2(0099909262)_v2.3 (snapmirror) fas2(0099909262)_v2.2 (snapmirror)	fas2(0099909262)_v2.3 fas2(0099909262)_v2.2 (snapmirror)	fas2(0099909262)_v2.3 <sup>e</sup> fas2(0099909262)_v2.2



- a. There is an extra Snapshot copy on fas2 (destination) after the first SnapMirror update (step 2).
- b. System fas3 also has the same number of Snapshot copies as fas2 after step 3 because there is a volume SnapMirror relationship between fas2 and fas3 systems
- c. A new soft lock exists on fas2:v2 after step 3 because fas2:v2 is now the volume SnapMirror source for fas3:v2.
- d. After step 4, the source system, fas1 contains two SnapMirror Snapshot copies. This is because the Snapshot copy fas2(0099909262)\_v2.2 is locked by fas2 system as it is required to continue to perform SnapMirror updates with fas3 system. This Snapshot copy on fas1 system is also used to perform direct SnapMirror updates with fas3 system in case fas2 system meets disaster
- e. After step 4, the source system, fas1 contains two SnapMirror Snapshot copies. This is because the Snapshot copy fas2(0099909262)\_v2.2 is locked by fas2 system as it is required to continue to perform SnapMirror updates with fas3 system. This Snapshot copy on fas1 system is also used to perform direct SnapMirror updates with fas3 system in case fas2 system meets disaster

### Single hop qtree SnapMirror

This configuration involves qtree SnapMirror replication between two systems, fas1 and fas2 (see Table 3-4).

fas1:vol1/qt1 → fas2:vol2/qt2

Table 3-4 Snapshot copy propagation for single hop qtree SnapMirror

Timeline	Snapshot copies on fas1	Snapshot copies on fas2
After qtree initialization and update on fas2	fas2(0099909262)_vol2_qt2-src.3 (snapmirror)	fas2(0099909262)_vol2_qt2-dst.8 (busy,snapmirror)

Be aware of the following Snapshot copy behaviors:

- ▶ Qtree SnapMirror Snapshot copy names are not identical. This is so, because the destination volume might contain other qtree or volume data besides qt2.
- ▶ The destination volume Snapshot copy has a hard lock created by SnapMirror (busy,snapmirror).
- ▶ The source volume Snapshot copy has a soft lock created by SnapMirror (snapmirror).

### Dual hop qtree SnapMirror and volume SnapMirror

This configuration involves qtree SnapMirror replication in the first hop and volume SnapMirror replication in the second hop (see Table 3-5).

fas1:vol1/qt1 → fas2:vol2/qt2; fas2:vol2 → fas3:vol3

Table 3-5 Snapshot copy propagation for dual hop qtree and volume SnapMirror

Timeline	Snapshot copies on fas1	Snapshot copies on fas2	Snapshot copies on fas3
After qtree initialization and update on fas2	fas2(0099909262)_vol2_qt2-src.3 (snapmirror)	fas2(0099909262)_vol2_qt2-dst.8 (busy,snapmirror)	
After fas3:vol3 initialization <sup>a</sup>	fas2(0099909262)_vol2_qt2-src.3 (snapmirror) fas3(0099909261)_vol3.1 fas2(0099909262)_vol2_qt2-dst.8	fas3(0099909261)_vol3.1 (snapmirror) fas2(0099909262)_vol2_qt2-dst.8 (busy,snapmirror)	fas3(0099909261)_vol3.1 fas2(0099909262)_vol2_qt2-dst.8

Timeline	Snapshot copies on fas1	Snapshot copies on fas2	Snapshot copies on fas3
After a qtree SnapMirror update on fas2 <sup>b</sup>	fas2(0099909262)_vol2_qt2-src.4 (snapmirror)  fas2(0099909262)_vol2_qt2-src.3 (snapmirror)	fas2(0099909262)_vol2_qt2dst.10 (busy,snapmirror)  fas3(0099909261)_vol3.1 (snapmirror)  fas2(0099909262)_vol2_qt2-dst.8 (snapmirror)	fas3(0099909261)_vol3.1  fas2(0099909262)_vol2_qt2-dst.8
After a volume SnapMirror update on fas3	fas2(0099909262)_vol2_qt2-src.4 (snapmirror)  fas2(0099909262)_vol2_qt2-src.3 (snapmirror)	fas3(0099909261)_vol3.2 (snapmirror)  fas2(0099909262)_vol2_qt2dst.10 (busy,snapmirror)  fas2(0099909262)_vol2_qt2-dst.8	fas3(0099909261)_vol3.2  fas2(0099909262)_vol2_qt2-dst.10  fas3(0099909261)_vol3.1 <sup>c</sup>  fas2(0099909262)_vol2_qt2-dst.8

- a. System fas3 also has the same number of Snapshot copies as fas2 after step 2 because there is a volume SnapMirror relationship between fas2 and fas3 systems.
- b. System fas1 retains the extra qtree SnapMirror Snapshot copy (fas2(0099909262)\_vol2\_qt2src.3) in step 3 because fas3 will be able to resynchronize with fas1 in the event fas2 meets with disaster.
- c. System fas3 has an extra volume SnapMirror Snapshot copy after the first volume SnapMirror update on the fas3 system.

## 3.4 Logging

The SnapMirror log file (located in /etc/logs/snapmirror) records the start and end of an update and other significant SnapMirror events. If a problem exists with updates, review the log file to see what happened since the last successful update. Because the log file is kept on the source and destination storage systems, quite often the source or the destination system might log the failure, and the other partner knows only that there was a failure. For this reason, review logs at both the source and the destination systems to get the most information about a failure. The log file contains the start and end times of each transfer, along with the amount of data transferred. It can be useful to look back and see the amount of data needed to make the update and the amount of time the updates take.

**Tip:** The time versus data sent might not be an accurate measure of achieved network throughput because the transfer is not constantly sending data.

## 3.5 Data ONTAP versions and resync

Qtree SnapMirror is not affected by Data ONTAP versions of source and destination systems. Volume SnapMirror requires the destination to be at the same or higher major release of Data ONTAP as that of the source. If the destination (the DR site) is running a higher major release version of Data ONTAP than that at the source (production site), the production site system will need to be upgraded if the newly written data at the DR site needs to be resynchronized to the production site (reversing the SnapMirror relationship).

SnapMirror resynchronization does not always require a full level 0 transfer. If both the source and destination have at least one Snapshot copy in common, SnapMirror computes and transfers only the changed blocks since the common Snapshot copy. In the absence of the common Snapshot copy, resynchronization requires a full transfer.

## 3.6 Data change rate

Using the **snap delta** command (Example 3-1), you can display the rate of change stored between two Snapshot copies as well as the rate of change between a Snapshot copy and the active file system. Data ONTAP displays the rates of change in two tables. The first table displays the rates of change between successive Snapshot copies. The second table displays a summary of the rate of change between the oldest Snapshot copy and the active file system.

*Example 3-1 Sample output from the snap delta command*

---

```
itsotuc4> snap delta
```

```
Volume vol0
working...
```

From Snapshot	To	KB changed	Time	Rate (KB/hour)
<hr/>				
hourly.0	Active File System	10460	0d 01:33	6707.516
hourly.1	hourly.0	13752	0d 03:59	3441.584
nightly.0	hourly.1	19132	0d 07:59	2392.746
hourly.2	nightly.0	13772	0d 04:00	3443.000
hourly.3	hourly.2	13800	0d 04:00	3439.252
hourly.4	hourly.3	13860	0d 03:59	3468.613
hourly.5	hourly.4	13792	0d 04:00	3448.000
nightly.1	hourly.5	21360	0d 07:59	2671.391

```
Summary...
```

From Snapshot	To	KB changed	Time	Rate (KB/hour)
<hr/>				
nightly.1	Active File System	119928	1d 13:33	3193.372

---

## 3.7 SnapMirror and LUNS

If the volumes or qtrees contain LUNs, the LUNs on the SnapMirror destination system are read-only, online, and unmapped starting in Data ONTAP 7.2. You can then map the LUNs and mount them read-only or use FlexClone to create a clone of the volume containing the LUNs and mount them read-write. This can be done without interrupting SnapMirror replication operations. Note that the use of FlexClone requires a license. In the case of qtree SnapMirror, LUNS can be cloned in the volume using the **lun clone** command, which is available with Data ONTAP software.

## 3.8 Space guarantees

When users require additional space, the administrator can increase the size of an aggregate volume by assigning additional disks to it. In a SnapMirror configuration, overcommitting the aggregate allows more efficient use of disk space on the destination. Only the data that is used on the SnapMirror source is used in the flexible volume on the SnapMirror destination. If that SnapMirror destination is broken, the disk usage is deducted from the overall aggregate. Unless mirrors are broken, you can have many source volumes of varying sizes all mapped to destination flexible volumes. Therefore, for a 1-TB SnapMirror source volume that is 75% full, the SnapMirror destination volume (or replica) needs 750 GB with the guarantee disabled and the full 1 TB with the guarantee enabled.

### 3.8.1 Overcommitting aggregates on the source system

To overcommit an aggregate volume, create flexible volumes with a guarantee of none or file so that the volume size is not limited by the aggregate size. The total size of the flexible volumes can be larger than the containing aggregate.

### 3.8.2 Overcommitting aggregates on the destination system

The disadvantage of overcommitting an aggregate is that SnapMirror updates will fail when the volume runs out of space. Another disadvantage is that not all volumes can be guaranteed if they all need to be made writable at once by breaking the SnapMirror relationship.

Prior to Data ONTAP 7.3, as long as the destination volume is a SnapMirror destination (replica), the guarantee is volume-disabled. Subsequently, when the destination is broken, the guarantee mode is the same as the volume mode. Table 3-6 summarizes the SnapMirror destination volume behavior in Data ONTAP 7.2.

*Table 3-6 SnapMirror destination volume guarantee behavior for Data ONTAP 7.2*

	Guarantee	Comments
Source volume	<b>volume</b>	Source volume is configured with guarantee.
Destination volume before SnapMirror initialization	<b>none</b>	Destination volume is configured with no guarantee. Destination volumes cannot have volume guarantees.
Destination volume when mirrored	<b>volume (disabled)</b>	Space guarantees are disabled on volume SnapMirror destination volumes.
Destination volume after SnapMirror break	<b>volume</b>	Destination volume inherits guarantee attribute of the source volume.

Starting in Data ONTAP 7.3, it is possible to set guarantees on the SnapMirror destination volume so that the SnapMirror updates never fail on that volume. The default behavior is that the volume guarantees are turned off. Table 3-7 summarizes the SnapMirror destination volume behavior in Data ONTAP 7.3 when the destination volume is configured with the volume guarantee.

Table 3-7 SnapMirror destination volume guarantee behavior for Data ONTAP 7.3

	Guarantee	Comments
Source volume	<b>volume</b>	Source volume is configured with guarantee.
Destination volume before SnapMirror initialization	<b>volume</b>	Destination volume is configured with volume guarantee.
Destination volume when mirrored	<b>volume</b>	Volume guarantee is preserved even for a SnapMirror destination volume.
Destination volume after SnapMirror break	<b>volume</b>	Volume guarantee is preserved on the destination volume.

Table 3-8 summarizes the SnapMirror destination volume behavior in Data ONTAP 7.3 when the destination volume is configured without guarantee.

Table 3-8 SnapMirror destination volume guarantee behavior for Data ONTAP 7.3

	Guarantee	Comments
Source volume	<b>volume</b>	Source volume is configured with guarantee.
Destination volume before SnapMirror initialization	<b>none</b>	Destination volume is configured without guarantee.
Destination volume when mirrored	<b>none</b>	The guarantee attribute is not inherited from the source volume.
Destination volume after SnapMirror break	<b>none</b>	The guarantee attribute is not inherited from the source volume.

## 3.9 Update failures

If a manually issued update fails for any reason, the user is informed. The update is not tried automatically because the user is in a position to reissue the command. If a scheduled transfer fails the `/etc/snapmirror`, and if the error is “retriable,” it is retried at the next minute. If it fails for non-retriable errors, such as user abort, or if the volume SnapMirror source denied the transfer for any reason, it is not retried at the next minute. Whether the error is retriable or non retriable, an update is always attempted whenever the schedule in `snapmirror.conf` specifies that the update should be performed.

If an update is in progress when another is scheduled to occur, SnapMirror starts another transfer as soon as the transfer is complete. However, if three updates pass while the current transfer is in progress, SnapMirror does only one more update; it does not go back and run updates that have been made obsolete by those scheduled later.

If a transfer fails and has to be retried, it is not generally started from the beginning. SnapMirror makes a restart checkpoint every 5 minutes during a transfer. If a restart checkpoint exists and if the baseline and incremental SnapMirror Snapshot copies exist, SnapMirror restarts the previous transfer where it left off. If not, SnapMirror creates a new Snapshot copy and starts a new transfer. Some of the conditions that prevent SnapMirror from restarting from a checkpoint are hostname changes, volume name changes, source volume size changes, and Data ONTAP version upgrades and downgrades.

Messages about failed snapmirror transfers can be found in the SnapMirror log file snapmirror (located in /etc/logs). Messages are also written to the /etc/logs/messages file, along with other system messages. Use the **rdfile** command to review these files.

Examples:

```
rdfile /etc/log/snapmirror
rdfile /etc/log/messages
```

## 3.10 Concurrent replication operations

A SnapMirror replication consists of two operations, one on the source side of the transfer and the other on the destination side. Therefore, if a storage system is the source of one replication and the destination of another replication, it uses two replication operations. Similarly, if a storage system is the source and the destination of the same replication, it uses two replication operations. Migrating from traditional volumes to FlexVol volumes with qtrees SnapMirror within the same controller is an example of the same storage system being both a source and a destination (see Table 3-9).

The number of concurrent replication operations in Data ONTAP 7.3 increased dramatically for asynchronous SnapMirror (both volume and qtrees). The increase depends on the platform. For example, N5600 supported 16 concurrent operations in Data ONTAP 7.2. Data ONTAP 7.3 supports up to 50 concurrent operations for volume SnapMirror.

Table 3-9 Concurrent replication operations

Storage System Model	Volume SnapMirror	Synchronous SnapMirror	Qtree SnapMirror	SnapVault	Open Systems SnapVault
5300	50	16	64	64	16
N5500	50	16	64	64	16
N5600	60	16	64	64	
N7600	100	24	96	96	34
N6040	50	16	64	64	16
N6060	50	16	64	64	16
N6070	50	16	64	64	16
N6210	50	16	64	64	16
N6240	50	16	64	64	16
N7600	100	24	96	96	34
N7700	100	24	96	96	34
N7800	150	32	128	128	32
N7900	150	32	128	128	32

For more information about the maximum number of concurrent replication operations that each storage system model can support, see the *Data Protection Online Backup and Recovery Guide* for the appropriate Data ONTAP release available on the IBM Storage support website:

<http://www.ibm.com/storage/support/nas>

A storage system might not reach the maximum number of concurrent replication operations for the following reasons:

- ▶ Storage system resources, such as CPU usage, memory, disk bandwidth, or network bandwidth, are taken away from SnapMirror or SnapVault operations.
- ▶ Each storage system in a high-availability (HA) configuration has the maximum number of concurrent replication operations. If a failover occurs, the surviving storage system cannot process more than the maximum number of concurrent replication operations specified for that storage system. These can be operations that were scheduled for the surviving storage system, the failed-over storage system, or both.

For example, each N5500 in a cluster can run a maximum of 16 concurrent replication operations. If one N5500 fails over to the other, it still has a maximum of 16 operations, which can be operations that were scheduled by the surviving N5500, the failed N5500, or both.

- ▶ Before Data ONTAP 7.3, concurrent operations were reduced to half when ATA drives were present in the storage system unless a NearStore option license was installed. Starting with Data ONTAP 7.3, this limitation due to ATA drives is removed.

## 3.11 NearStore personality

Starting with Data ONTAP 7.2, a software license option called the NearStore Personality option (nearstore\_option) has been introduced to use the N series storage systems as a secondary storage system. For specific Data ONTAP version requirements for this license, check the product manuals at this website:

<http://www.ibm.com/storage/support/nas/>

The goal of this license option is to provide increased concurrent streams when storage systems are used as destinations for SnapMirror/SnapVault transfers and to enable SnapVault for NetBackup. This license option should not be installed on primary storage systems where performance is paramount.

Before Data ONTAP 7.3, the concurrent operations were halved when ATA drives were added to the system. When the NearStore option license is installed, the concurrent operations are doubled again. An example follows.

**Limits:** Concurrent operation limits change with Data ONTAP versions and platforms; see the appropriate release documentation for that particular version. The concurrent operation limits are total for any given system and not cumulative. If the N5600 system allows 50 maximum concurrent replication operations for volume SnapMirror, or 16 maximum concurrent replication operations for SnapMirror Sync, the system will not allow both the 50 volume SnapMirror transfers and 16 SnapMirror Sync transfers at any given time.

## 3.12 System-wide throttle

Starting with Data ONTAP 7.2, there is a system-wide option to limit the total bandwidth used by all transfers at any time. This can be either the transmit bandwidth on the source or the receive bandwidth on the destination, or both. The per-transfer throttle from `snapmirror.conf` will still be applied. When both per-transfer and system-wide throttling are configured, throttling at system-wide is applied only if the combined bandwidth used by all the relationships goes above the system-wide throttling value.

System-wide throttling is enabled by using three new options for the **options** command:

`replication.throttle.enable`

This option enables global network throttling of SnapMirror and SnapVault transfers. The default value for this option is *off*.

`replication.throttle.incoming.max_kbs`

This option is set on the destination system. It specifies the maximum total bandwidth used by all the incoming SnapMirror and SnapVault transfers, specified in kilobytes/sec. The default value for this option is *unlimited*, which means that there is no limit on total bandwidth used. This option is valid only when the `replication.throttle.enable` option is *on*.

`replication.throttle.outgoing.max_kbs`

This option is set on the source system. It specifies the maximum total bandwidth used by all the outgoing SnapMirror and SnapVault transfers specified in kilobytes/sec. The default value for this option is *unlimited*, which means that there is no limit on total bandwidth used. This option is valid only when the `replication.throttle.enable` option is on (see Figure 3-2).

```
itsotuc4> options replication.throttle
replication.throttle.enable off
replication.throttle.incoming.max_kbs unlimited
replication.throttle.outgoing.max_kbs unlimited
```

Figure 3-2 Default throttle option settings

## 3.13 Dynamic throttle

Starting in Data ONTAP 7.1, an active SnapMirror relationship can be throttled to decrease the amount of bandwidth it uses. This dynamic relationship does not need to be stopped for the new throttle value to take effect.

Here is the syntax and an example of usage:

```
snapmirror throttle <n> <system>:<destination path>
```

Where `<n>` is the new throttle value in kilobytes per second.

An example is shown in Figure 3-3. The result is shown in Figure 3-4.



```

itsotuc4> snapmirror status
Snapmirror is on.
Source                Destination                State      Lag      Status
itsotuc1:itso_1      itsotuc4:itso_1          Snapmirrored  37:17:31
Transferring (1320 MB done)
itsotuc4>
itsotuc4> snapmirror throttle 1024 itsotuc4:itso_1
itsotuc4> Tue Feb 23 14:18:11 MST [replication.dst.throttleChange:notice]:
Throttle value changed to 1024 Kbytes/s for destination itsotuc4:itso_1

```

Figure 3-3 dynamically throttling a SnapMirror

```

src Tue Feb 23 14:15:29 MST itsotuc1:itso_1 itsotuc4:itso_1 Request
(9.11.218.238)
slk Tue Feb 23 14:15:30 MST state.softlock.itso_1.0007dc5c.118.itsotuc4:itso_1
Softlock_add (Transfer)
src Tue Feb 23 14:15:30 MST itsotuc1:itso_1 itsotuc4:itso_1 Start
src Tue Feb 23 14:18:09 MST itsotuc1:itso_1 itsotuc4:itso_1 Throttle (1024KB/s)

```

Figure 3-4 /etc/log/snapmirror after dynamic throttling

The new value is used only for a current transfer. The next scheduled transfer will use the throttle value specified in the `/etc/snapmirror.conf` file. This command can be used only when the transfer is active.

If the throttle value for the next scheduled transfer needs to be changed, then the value in the SnapMirror configuration file should be modified. The command can be run from either the source or the destination.

There is another way to change the throttle value for an active transfer: by changing the value in the `/etc/snapmirror.conf` file. This change takes effect in 2 minutes. The SnapMirror `throttle` command does not change the throttle value specified in `/etc/snapmirror.conf`.

## 3.14 Firewall configuration

SnapMirror uses the typical socket/bind/listen/accept sequence on a TCP socket.

### 3.14.1 SnapMirror Async

The SnapMirror source system listens on port 10566. A firewall configuration must allow requests to this port on the SnapMirror source system. When using a multipath connection, the destination system listens on port 10565.

### 3.14.2 SnapMirror Sync and SnapMirror Semi-Sync

SnapMirror requires additional TCP ports to be open. The source system listens on TCP ports 10566 and 10569. The destination system listens on TCP ports 10565, 10567, and 10568. Therefore, a range of TCP ports from 10565 to 10569 is best.

## 3.15 Network compression configuration and operation configuration

The SnapMirror configuration file (/etc/snapmirror.conf) is used to configure and enable SnapMirror network compression. SnapMirror network compression requires a connection name in the configuration file, as shown in the following examples. A connection name entry is mandatory for network compression to work. Merely adding the compression=enable option without the connection name entry does not enable SnapMirror network compression.

When using a single network path:

```
connection_name=multi(src_system,dst_system)
connection_name:src_vol dst_system:dst_vol compression=enable 0 * * *
```

In this example, connection\_name is the name of the specific connection between a SnapMirror source and destination system pair.

src\_system is the name of the SnapMirror source system.

dst\_system is the name of the SnapMirror destination system.

src\_vol is the path of the SnapMirror source volume.

dst\_vol is the path of the SnapMirror destination volume.

This is an example of an /etc/snapmirror.conf file:

```
pumpkin=multi(fas1,fas2)
pumpkin:src_vol fas2:dst_vol compression=enable 0 * * *
```

Here is another example when using multiple paths:

```
connection_name=multi(src_ip1,dst_ip1)(src_ip2,dst_ip2)
connection_name:src_vol dst_system:dst_vol compression=enable 0 * * *
```

This is an example of an /etc/snapmirror.conf file:

```
pumpkin=multi(fas1_e0a, fas2_e0a)(fas1_e0b,fas2_e0b)
pumpkin:src_vol fas2:dst_vol compression=enable 0 * * *
```

**Tip:** Entries on the connection name line can be either host names or IP addresses. The connection name itself cannot be the hostname.

## 3.16 Enabling and disabling network compression

SnapMirror network compression cannot be enabled and disabled for an active transfer. It can be enabled by adding the compression=enable option to the configuration file. It can be disabled by removing the compression=enable option from the configuration file. To enable compression for an existing transfer, you must first abort the transfer, update the configuration file, and then resume the transfer.

## 3.17 Reporting the compression ratio

The SnapMirror network compression ratio is reported in the log file (/etc/log/snapmirror) and is also displayed in the long (-l) listing of SnapMirror status. SnapMirror status displays the ratio only when SnapMirror is in transferring state. However, the log file will always contain the compression ratio information:

- From /etc/log/snapmirror:

```
dst Sat Jun 20 17:50:18 IST 20502_rep:oracleDB8 fas2050-nb02:oracleDB8 Start
dst Sun Jun 21 00:33:55 IST 20502_rep:oracleDB1 fas2050-nb02:oracleDB1 End
(52868900 KB, Compression 3.5 : 1)
```

- From the **snapmirror status** command:

```
fas2> snapmirror status -l dest
Snapmirror is on.
Source: fas1:src
Destination: fas2:dest
Status: Transferring
Progress: 24 KB
Compression Ratio: 3.5 : 1
State: Lag:
Mirror
Timestamp: Base
Snapshot: Current
Transfer Type: Initialize
Current Transfer Error:
Contents:
Last
Transfer Type: Initialize
Last Transfer Size: 132 KB
Last Transfer Duration: 00:00:04
Last Transfer From: fas1:src
```

## 3.18 Compression ratio and data sets

SnapMirror network compression ratios depend on the type of data set. Table 3-10 shows some of the compression ratios seen in the lab with real-world application data sets.

*Table 3-10 Sample SnapMirror network compression ratios and network savings*

Data set	Compression ratio	Percent network savings
Exchange DB	1.5:1	34%
Home Directory	2.7:1	60%
Oracle DB	3.5:1	70%

## 3.19 64-bit aggregates

Data ONTAP 8.0 7-Mode introduces a new aggregate type, 64-bit aggregates, with a larger maximum aggregate and flexible volume sizes. The maximum size for legacy aggregates, now called 32-bit aggregates, remains 16 TB. There are some restrictions on replicating between 32-bit and 64-bit aggregates using SnapMirror. Table 3-11 and Table 3-12 provide an overview of possible replication between 32-bit and 64-bit aggregates for volume and qtree SnapMirror. Volume SnapMirror replication is possible only between like type aggregates.

*Table 3-11 Volume SnapMirror replication between 32-bit and 64-bit aggregates*

<b>Volume SnapMirror</b>	<b>Destination volume in 32 bit aggregate</b>	<b>Destination volume in 64 bit aggregate</b>
Source volume in 32-bit aggregate	Yes	No
Source volume in 64-bit aggregate	No	Yes

*Table 3-12 Qtree SnapMirror replication between 32-bit and 64-bit aggregates*

<b>Qtree SnapMirror</b>	<b>Destination qtree in 32 bit aggregate</b>	<b>Destination qtree in 64 bit aggregate</b>
Source qtree in 32-bit aggregate	Yes	Yes
Source qtree in 64-bit aggregate	Yes	Yes

## 3.20 SnapMirror over Fibre Channel

SnapMirror over Fibre Channel enables SnapMirror replication over a Fibre Channel SAN environment and includes all the features that are available with SnapMirror over Ethernet. For specific product requirements, see the Data Protection Online Backup and Recovery Guide at this website:

<http://www.ibm.com/storage/support/nas/>



## Guidelines for SnapMirror

The following best practices refer primarily to the asynchronous mode of SnapMirror.

## 4.1 Growing destination volume

For volume SnapMirror, the destination volume must be the same as or larger than the source volume. Volume SnapMirror updates fail if the destination volume is smaller than the source volume. During failover and giveback scenarios, the source and destination roles are reversed during these scenarios.

If the source volume size has been increased by autogrow (**vol autosize** command) or by manual process, the destination volume needs to be matched with the source volume. There are different ways to handle the size mismatch. The first way is to provision the destination volume larger than the potential size to which the source volume would ever grow.

In this scenario, the SnapMirror update matches the source volume size because the underlying volume has sufficient space. If the destination volume is not provisioned to have enough space, then first turn the `fs_size_fixed` option off on the destination volume and then resize the destination volume to the same size as the source volume by using the **vol size** command. The next SnapMirror update command will then match the destination volume size to be the same size as the source volume.

## 4.2 SnapMirror window size, network latency, and compression

The SnapMirror TCP window size is the amount of data that a source can send on a connection before it requires acknowledgment from the destination that the data was received. The default TCP window size for SnapMirror operations is 1,994,752 bytes (2 MB). Generally, it is better not to change the TCP window size, unless there are throughput issues related to bandwidth utilization. If the replication is occurring over a lossy network, TCP has automatic window sizing adjustments. On the other hand, if you have a network link that is very large with high latency, a larger window size might yield higher throughput.

### 4.2.1 Window size changes in Data Ontap 7.3.2

Starting in Data ONTAP 7.3.2, SnapMirror TCP window size has been increased for volume SnapMirror. See the following table for details on SnapMirror TCP window sizes for different Data ONTAP versions.

### 4.2.2 Volume SnapMirror TCP window size limits

This section describes the maximum SnapMirror TCP window size limits for volume SnapMirror for various Data ONTAP versions. Qtree SnapMirror has a window size limit of 2 MB regardless of the Data ONTAP version (see Table 4-1).

Table 4-1 Volume SnapMirror TCP window size

Data ONTAP version	Mode	Maximum Window Size
Data ONTAP 7.2	Single-path	2 MB
	Multipath	2 MB
Data ONTAP 7.3.0 <sup>a</sup>	Single-path	2 MB
	Multipath	2 MB

Data ONTAP version	Mode	Maximum Window Size
Data ONTAP 7.3.2	Single-path	7 MB
	Multipath	14 MB

a. Patches are available for Data ONTAP 7.3.0 and 7.3.1 versions to increase the window size limits from 2 MB to 7 MB (single-path) and 14 MB (multipath).

Multipath configuration increases the maximum window size up to 14 MB. Multipath configuration is discussed in 2.10, “Configuration files” on page 20.

### 4.2.3 SnapMirror TCP window size and throughput

Maximum possible throughput is defined by the following equations:

$$\text{Maximum Throughput} = \text{Window Size} / \text{Round Trip Time (RTT)}$$

or

$$\text{Window Size} = \text{Max. Theoretical Throughput} \times \text{Round Trip Time}$$

These equations show that as the round trip time increases, the maximum possible throughput decreases *unless* SnapMirror TCP window size is increased.

This section explores this equation with two different networks:

- ▶ Link 1: OC-3 link (155 Mbps) with 100 ms RTT
- ▶ Link 2: OC-12 link (622 Mbps) with 100 ms RTT

The required window size to fully utilize the Link 1 = 155 Mbps X 100 ms = 1,937,500 bytes. Because the default window size is greater than this value, no changes need to be made in the configuration file.

The required window size to fully use Link 2 = 622 Mbps X 100 ms = 7,775,000 bytes.

Because the required window size (7,775,000 bytes) is greater than the default window size (2 MB), and if you need SnapMirror to fully use the link, you must configure SnapMirror with the larger window size as shown in the following example.

Example of /etc/ snapmirror.conf file:

```
pumpkin=multi (192.168.0.1, 192.168.0.2) ( 192.168.1.1,192.168.1.2)
pumpkin:sourcevol orange:destvol wsize=7775000 0 * * *
```

Use the above formula to determine whether a larger than 2 MB window size is required. It is best that the TCP window size be changed only if a window size larger than 2 MB is required.

### 4.2.4 SnapMirror TCP window size and compression

This section discusses the effects of network compression on SnapMirror TCP window size. SnapMirror has a built-in flow control mechanism at the application layer to determine how much data to pass on to the next layer. The network compression occurs between the application layer and the network layer. Because the flow control mechanism is at the application layer, the flow control is based on uncompressed data. It is unaware that compression is occurring at the next layer. Therefore, the previous formula needs to be modified when compression is enabled:

$$\text{Window Size} = \text{Max. Theoretical Throughput} \times \text{Round Trip Time} \times \text{Compression Ratio}$$

Assume that the data set has a compression ratio of 2:1. Using the 155 Mbps link with 100 ms Round Trip Time (RTT) example, calculate the required window size to fully use the pipe:

Required window size = 155 Mbps X 100ms X 2 = 3,875,000 bytes

Here is an example of the `/etc/ snapmirror.conf` file:

```
pumpkin=multi (192.168.0.1, 192.168.0.2) ( 192.168.1.1,192.168.1.2) pumpkin:sourcevol  
orange:destvol compression=enable,wsiz=7775000 0 * * *
```

Finally, note that synchronous replication in general is not feasible over large distances such as wide area networks, which typically have large round-trip time and packet loss, resulting in performance impact on the primary workload. Therefore, window size does not become an issue in synchronous replication scenarios.

## 4.3 Replication network configuration

When possible, use a private network between source and destination for replication purposes. This isolates replication traffic from the client traffic. Otherwise, these two types of traffic compete for bandwidth. Sometimes, SnapMirror might need to be throttled for two primary reasons: Decrease WAN bandwidth use by SnapMirror, and decrease the storage system's resource consumption by SnapMirror. To figure out the amount of bandwidth required, follow these simple steps:

1. Find out the peak data change rate (using the **snap delta** command).
2. Find out the RPO (or SnapMirror update interval).

After you have this data, you can now calculate the theoretical minimum required bandwidth. An example follows.

Assume you have a data set of 1 TB with a daily change rate of 5% (or 50 GB per day or 2 GB/hour). Assume your RPO requirement is 1 hour. This means that you will need to complete the SnapMirror transfer of 2 GB in one hour or approximately 4.7 Mbps. This is the theoretical minimum bandwidth required to complete the SnapMirror transfer. The actual throughput depends on the storage system utilization, round trip latency of the network link, and the network pipe utilization by other applications.

Also keep in mind that the data change rate is not uniform throughout the day even though the above example assumes same data change rate throughout the 24-hour period. Use the peak data change rate in your scenario to calculate the minimum bandwidth requirement.

## 4.4 Replication frequency and Snapshot schedules

Although it is possible to do replication updates every minute, it is not desirable. The first step in a SnapMirror update involves computation of changed blocks. This can be a CPU-intensive process. The storage system can spend a lot of time in computing changed blocks when SnapMirror updates are set up to run every minute on multiple volumes. This in turn could affect the primary workloads.

The other issue is that the entire SnapMirror update process must finish within the minute for the all the volumes before the next update starts again. There might not be sufficient time to calculate the block changes and transfer the data within this short period. Therefore, asynchronous SnapMirror updates at every minute are not desirable.



If the RPO requirements are very low (< 3 minutes or so), consider the use of SnapMirror Semi-Sync.

For optimal performance, make sure that the SnapMirror updates and Snapshot schedules (`snap sched`) do not occur at the same time.

## 4.5 Destination Qtree names

Destination qtree names cannot contain wildcards and special characters. There is also a 64-character limit on the length of names. A qtree name must be present and appended to the full volume path. The full qtree path must also be preceded with `/vol` before the volume name that the qtree resides in. For non-qtree data, “-” is specified in place of the qtree name. This replicates all the data in the volume that is not in a qtree to a qtree. This can have impact on mount points because the data paths have changed.

## 4.6 Many-to-one configuration

When multiple systems are replicating data to a single system, make sure that the destination storage system can service the combined write throughput requirements of all source systems.

## 4.7 Upgrading to flexible volumes

For volume SnapMirror, the source and destination volumes must be “like” volumes. That is, both source and destination must be either traditional or flexible volumes. In keeping with this requirement, the source and destination volumes need to be upgraded simultaneously. Furthermore, because the destination is a read-only volume, for migration purposes, the destination volume must be writable, so that a container file can be created. Therefore, the SnapMirror relationship must be broken prior to starting the migration process. After starting the migration process on the source and destination volumes, the SnapMirror relationship can be resumed.

## 4.8 Unicode

Directories on all SnapMirror source volumes that support CIFS clients must be in Unicode format before being replicated to a destination; otherwise, the directories in the read-only destination volume will not be in Unicode format and attempts through CIFS to access directories and open files on the destination volume might receive “access denied” errors. This is true in qtree SnapMirror deployments only.

## 4.9 High file count environments and Qtree SnapMirror

A high file count (HFC) environment is defined as any single volume containing millions of files. The `filestats` command helps identify HFC environments. The command requires CPU time, so it should be run during low I/O periods. When using qtree SnapMirror in an HFC environment, follow these guidelines:

- ▶ Avoid HFC with numerous qtrees. Each qtree triggers an additional scan of changed inodes. It is best that users stay under two qtrees in HFC environments.
- ▶ Avoid HFC and large directories with applications that generate lots of activity in each of these directories.
- ▶ Avoid HFC and many small directories with applications that generate lots of activity in each of these directories.

## 4.10 Read performance on a FlexVol volume SnapMirror destination

When a volume SnapMirror update finishes in a FlexVol configuration, the destination storage system launches a process to recreate the fast-path metadata so that it is consistent with the FlexVol and aggregate layout on the destination storage system. During metadata creation, read performance on a flexible volume that is a volume SnapMirror destination might be significantly worse than that experienced from a flexible volume that is not a volume SnapMirror destination. This does not affect qtree SnapMirror, flexible volume SnapMirror source, or volume SnapMirror for traditional volumes.

When this process finishes, reads to the SnapMirror destination use the normal fast-path, and read performance on the destination flexible volume returns to normal.

This read performance issue might affect two classic scenarios: 1) Users who need immediate access to the volume SnapMirror FlexVol destination soon after an update is completed, and 2) Users who perform tape backups from the destination volume soon after an update is completed. Slow performance is seen until the process completes recreation of the fast-path metadata.

**Guideline:** To minimize the impact due to read performance, use Data ONTAP 7.2.4 or later.

## 4.11 Data ONTAP upgrade and revert considerations

When upgrading and reverting between major Data ONTAP versions, exercise caution when deleting Snapshot copies that exist in the older Data ONTAP versions. Consider the following scenario: A system is upgraded from Data ONTAP 7.2 to Data ONTAP 7.3. All the Snapshot copies that currently exist were created after the upgrade.

The same is true on the destination because volume SnapMirror maintains the same set of Snapshot copies on source and destination. If the system needs to be reverted back to Data ONTAP 7.2, the relationship must be initialized. This is necessary because one of the steps during the revert process is to delete the Snapshot copies created in Data ONTAP 7.3.

**Guideline:** This situation can be avoided by manually creating a Snapshot copy on the older Data ONTAP version before the upgrade. When you are sure that there is no need to revert, you can delete this manually created Snapshot copy.

## 4.12 SnapMirror network compression considerations

Compression is processor-intensive and therefore each N series platform has limitations on compression throughput. Table 4-2 shows approximate compression throughput limits. At these limits, almost all of the N series system processing cores are being used by compression.

*Table 4-2 Compression throughput limits*

Storage platform	Maximum best bandwidth for compression
N3300, N3600, N5200	60 Mb/sec
N3400, N5500, N5300, N6040, N7600, and N7700	120 Mb/sec
N6060, N5600, N6070, N7800	240 Mb/sec
N7900	500 Mb/sec

Keep in mind these per-system compression throughput limits when deploying SnapMirror network compression in medium or high bandwidth environments. If the available bandwidth for replication exceeds the compression throughput limits in a single-system environment, the benefits of compression might not be realized.

### 4.12.1 Compression versus decompression

SnapMirror network compression and decompression processing result in some processor overhead. In general, processor overhead due to decompression is roughly 40% less than the processor overhead due to compression.

### 4.12.2 Transfer times

If the goals are to keep the SnapMirror transfer time constant and to lower the network bandwidth utilization, the data set with the highest compression ratio has the lowest processor overhead and uses the lowest bandwidth. This is because the highest compression ratio results in the lowest amount of work to be done to meet the fixed amount of transfer window. In this case, the processor overhead can be as low as 10% due to compression.

For example, assume a data set that yields 2:1 compression and that without network compression the SnapMirror update takes one hour using a bandwidth of 100 Mb/sec. With network compression enabled only 50 Mb/sec bandwidth is needed to achieve the same transfer time. Because over-the-wire throughput is lower, processor utilization due to network processing is decreased, compensating for the increased processor utilization by compression.

If the goal is to lower the SnapMirror transfer time by making the entire bandwidth available, the data set with the highest compression might result in the highest amount of processor overhead and finish the transfer in the lowest amount of time. For example, consider the same data set as above with 2:1 compression in which an update without compression takes one hour using a bandwidth of 100 Mb/sec. With network compression enabled, the transfer completes in 30 minutes.

Because the work is completed faster by using the entire bandwidth, network processing overhead is higher, and the SnapMirror processing must also be completed in half the time. If processor utilization is too high, you can use SnapMirror throttling (either per transfer or global) to adjust the throughput so that processor utilization does not go too high. Figure 4-1 summarizes the effects of compression on total transfer time for our three data sets.

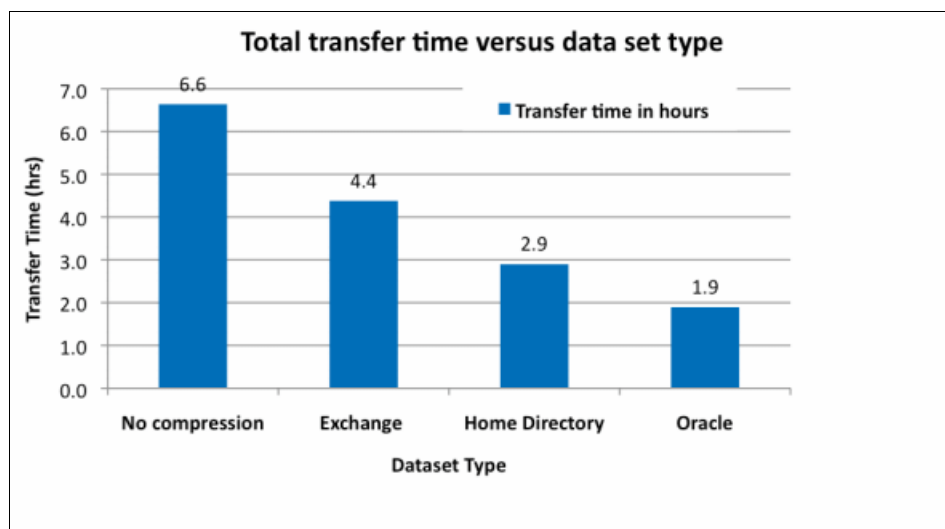


Figure 4-1 Transfer times for various data sets

### 4.12.3 Compression ratio

Another factor that affects processor overhead is the compression ratio achievable for a data set. If a data set yields a high compression ratio, the total amount of effort to compress and send it over the wire is less compared to a data set that is hard to compress. In contrast, if you do not throttle bandwidth for a transfer, a data set with a high compression ratio could require a lot of compression work to fill the pipe, raising processor utilization.

For example, to fill a 100 Mb/sec pipe with Oracle data that compresses at 3.5:1, your storage system would have to compress data at a rate of 350 Mb/sec; while filling the same pipe with Exchange data that compresses at 1.5:1 would only require a compression or decompression rate of 150 Mb/sec. If the goals are to keep the SnapMirror transfer time constant and to lower the network bandwidth utilization, the data set with the highest compression ratio has the lowest processor overhead and uses the lowest bandwidth.

Figure 4-2 shows the interdependence between data transfer time, network bandwidth consumption, and processor overhead. Decreasing transfer time necessarily consumes more network bandwidth and system processor resources. The processor overhead in the reduced transfer time case depends on the bandwidth and the N series platform.

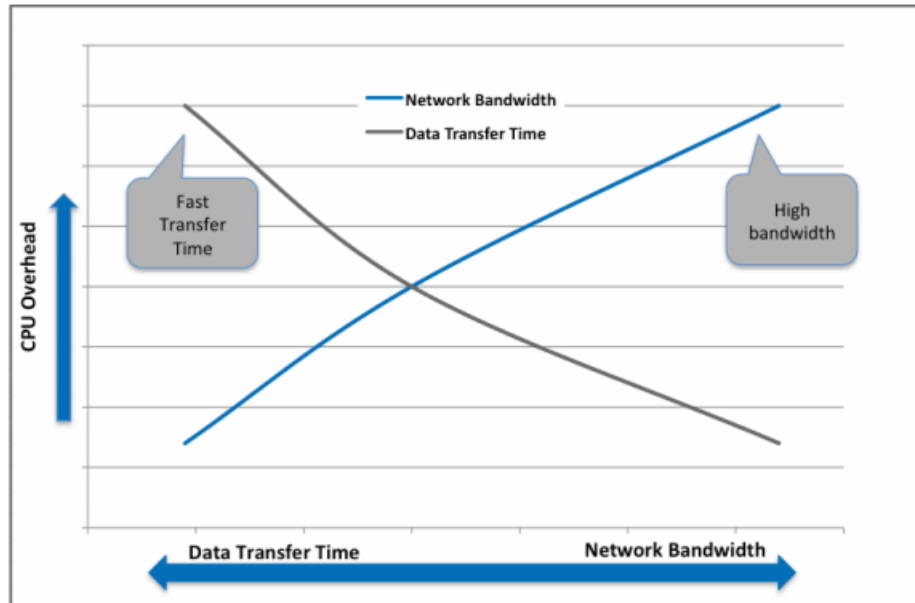


Figure 4-2 Relationship between CPU overhead, transfer time, and network bandwidth





## Network-free seeding

Network-free seeding is defined as initial data transfer between SnapMirror source and destination without the use of a network. This is extremely useful in limited network bandwidth scenarios such as wide area networks. Because the initial SnapMirror transfer involves the entire data transfer in a given volume or qtree, this can take a very long time over a small network pipe. SnapMirror supports network-free seeding with the use of two technologies: SnapMirror to tape (SM2T) and Logical Replication (LREP).

## 5.1 SnapMirror to tape

SnapMirror to tape (SM2T) gives users the ability to perform the initial full-volume transfer of a SnapMirror volume by using tapes.

### 5.1.1 Overview

The user can transfer the contents of a source volume to tapes. When the tapes have been created, remote volumes can be “seeded” with the source volume data by shipping these tapes to the remote locations and restoring their contents to the target volumes. After the initial data has been transferred to the target volumes from tapes, subsequent incremental transfers are performed over the network.

This feature can significantly reduce the data transfer over the WAN and the amount of time required to initialize a SnapMirror target volume. SnapMirror to tape does not support incremental transfers and is not intended for backups or data archiving. Symantec NetBackup 4.5 and later can also be used to control the SnapMirror to tape operations.

### 5.1.2 Restrictions

Only a baseline transfer to tape is currently supported, not incremental. There is no way to get just the changed blocks since the last backup, as you can with the **snapmirror update** command. SM2T also transfers all Snapshot copies in the volume and the active file system. There is no way to select a single Snapshot copy where you know that an application or file system was consistent (unlike with an NDMP- based application).

SM2T works only with volumes and is based on volume SnapMirror. Therefore, backup and restore works only between volumes of the same type (both flexible or both traditional volumes). Because SM2T is based on volume SnapMirror, SM2T has the same version restrictions as volume SnapMirror.

**Tip:** When using SM2T, using the **-g** option allows you to specify the destination geometry at the time of writing the tape with **snapmirror store**. This is to mitigate any performance issues with disk geometry mismatch, and it only applies to traditional volumes.



## 5.2 Logical Replication (LREP)

LREP is a logical replication tool that is useful for qtree SnapMirror or SnapVault initial transfers (also commonly referred to as seeding baselines).

### 5.2.1 Overview

Like SnapMirror to tape, LREP is used to perform the initial transfer (the baseline) to portable media. The portable media is shipped to the remote site and then the data can be transferred to the destination system. No network bandwidth is used, only a manual process of moving the media. After the data is on the destination system, modify the SnapMirror relationship to reflect the actual source and destination relationship.

While SnapMirror to tape runs on Data ONTAP, LREP is available for open systems clients such as Windows, UNIX, and Linux. LREP can also use a disk drive or even a FAS system as the “portable” media.

Two utilities are required for the entire LREP process: lrep\_writer is used at the location of the destination system and lrep\_reader is used at the source system.

You can download the latest LREP tool and user guide from this website:

<http://www.ibm.com/storage/support/nas>

### 5.2.2 Downloading the LREP tool

The binary files for the LREP tool are also packaged with the Open Systems SnapVault software. You can access the binary files from the same location where you uncompressed the Open Systems SnapVault software package on the primary storage system. LREP 2.0 is packaged with Open Systems SnapVault 2.5.

You can download Open Systems SnapVault following the instructions for Open System SnapVault Publication Matrix available on the IBM NAS support website:

<http://www.ibm.com/storage/support/nas>





## SnapMirror management

The most common methods to manage SnapMirror are CLI and FilerView. These methods work very well in small environments with a handful of systems. In large environments, they become tedious and cumbersome.

## 6.1 Protection Manager

Protection Manager provides policy-based management and automated data protection configurations. Automation and policy-based management approaches reduce the possibility of user errors. Protection Manager also provides a holistic view of the N series disk-based data protection status. Protection Manager runs within the N series Management Console alongside Performance Advisor and Provisioning Manager. Protection Manager can detect, manage, and monitor SnapMirror, SnapVault, and Open Systems SnapVault relationships. Currently, Protection Manager cannot manage SnapMirror Sync and SnapMirror Semi-Sync.

## 6.2 Concepts

Protection Manager uses three fundamental concepts: policies, resource pools, and data sets.

A *policy* is a rule that describes how to protect data. Protection Manager helps define policies in a graphical and intuitive manner. The policy can be applied to a volume, a LUN, or a user-defined group called a *data set*. An example of a data set is a group of LUNs that support the same application. The policies define the protection levels of the data sets to a group of resources called *resource pools* (see Figure 6-1).

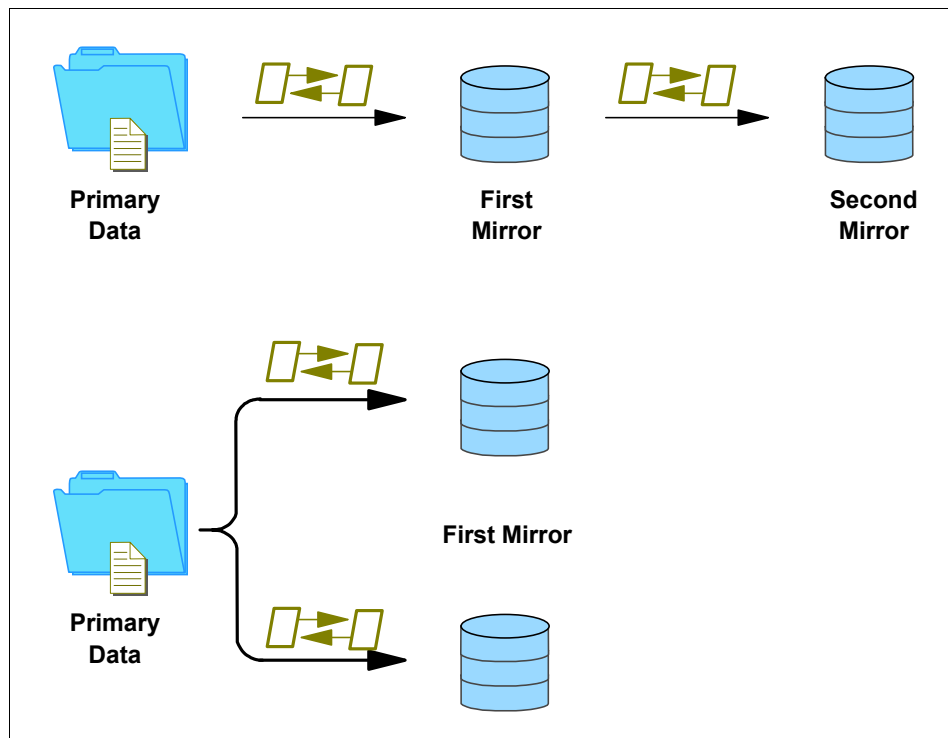


Figure 6-1 Examples of policies

**Tip:** Mirrors defined in Protection Manager use volume SnapMirror, and backup definition uses qtree SnapMirror.

Protection Manager also allows setting up throttle schedules to limit network bandwidth used by the mirror and backup operations. After the schedule is created, it can be applied to one or multiple policies.

### 6.3 Data conformance and monitoring

Protection Manager provides a conformance monitor that regularly checks for data set conformance to a policy. Data sets are marked as either in conformance or out of conformance. When the conformance change is detected, Protection Manager can attempt to perform corrective steps to bring the data set back into conformance or notify the administrator of the conformance changes.

Protection Manager also allows the administrator to monitor data protection status and to set alerts when certain conditions are met. These conditions can be failed data protection tasks or when SnapMirror lag times exceed certain thresholds.

### 6.4 Disaster recovery

Protection Manager 3.7 provides a new DR management feature that allows the user to apply a DR-capable policy to a data set. DR-capable data sets have attributes such as failover readiness and capabilities such as automated failover, and they export the DR volumes to the DR clients. DR failover readiness is shown on the dashboard with the status (see Figure 6-2 and Figure 6-3).

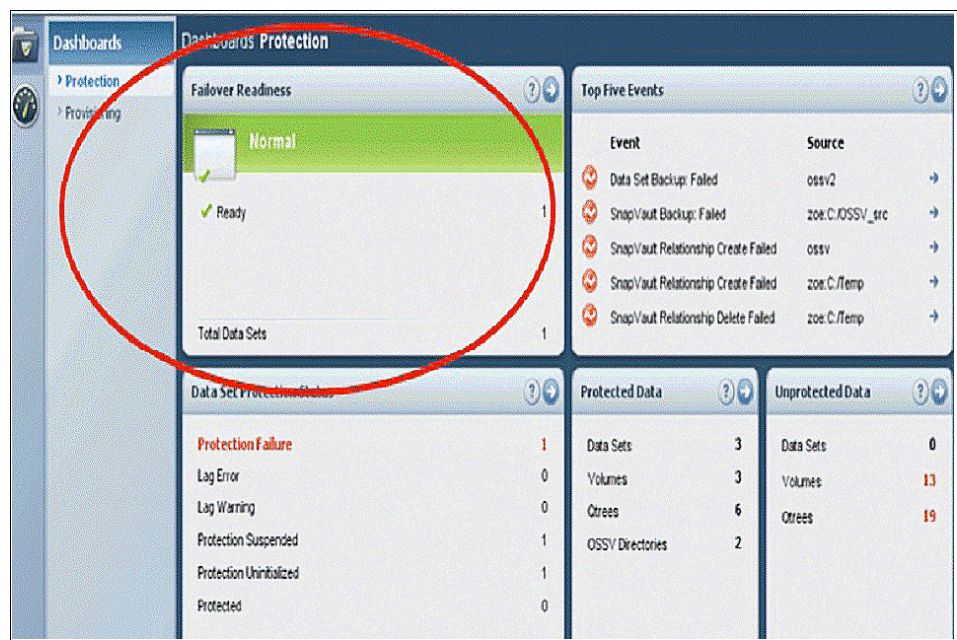


Figure 6-2 Failover readiness dashboard

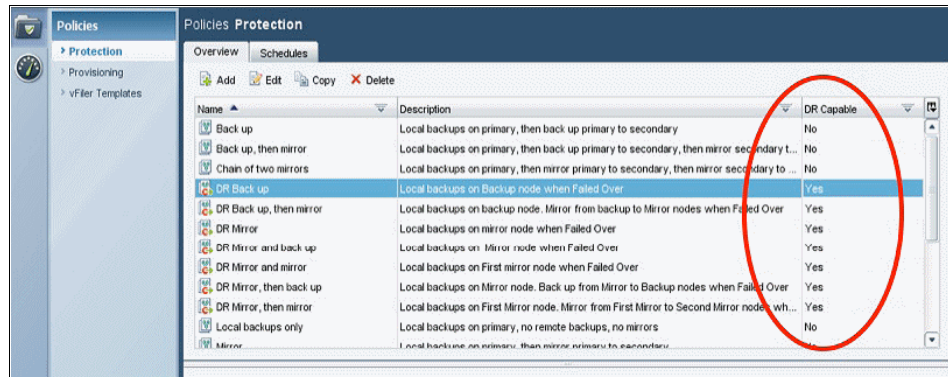


Figure 6-3 Disaster-Recovery capable attribute of data sets

There is also a new disaster recovery tab in Protection Manager 3.7. Only DR-capable data sets are listed under this tab. Four important action buttons are available. The Failover button makes the DR volumes writable by breaking the SnapMirror relationships. The Test button verifies that the failover scripts work properly without doing a failover. The Update button performs a SnapMirror update. The **Cancel** button cancels a failover task. Protection Manager 3.7 does not provide an automated failback. This can be done using CLI.

For more information about Protection Manager, see *IBM System Storage N series Provisioning Manager and Protection Manager Administration Guide for Use with DataFabric Manager 4.0*, GC26-7889-06.



## Use of SnapMirror with other N series products

This chapter covers the possible uses of SnapMirror with other N series products.

## 7.1 N series Manageability Suite

The N series Manageability Suite (see Figure 7-1) consists of four layers designed to integrate with your current environment.

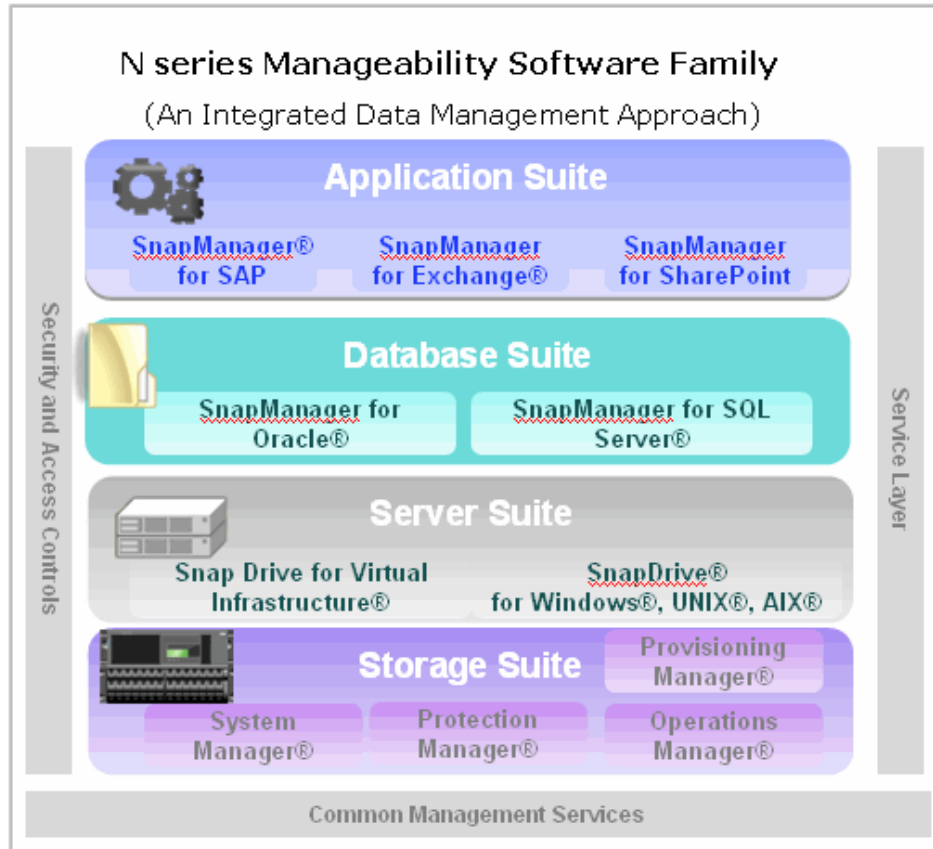


Figure 7-1 N series Manageability Suite

### 7.1.1 Application Suite and Database Suite

The N series Manageability Application Suite consists of SnapManager for Exchange, SnapManager for SharePoint Server, and SnapManager for SAP. The N series Manageability Database Suite consists of SnapManager for Oracle and SnapManager for SQL Server. The SnapManager Suite integrates with key applications, thereby reducing the risks associated with backups, and providing more flexibility in streamlining IT operations.

The SnapManager Suite can be used to take consistent backups by leveraging the application integration. With the use of N series Snapshot technology, SnapMirror can be used to extend the protection by replicating these consistent backups to a different storage system located either within the data center or to another data center located on the campus or at a remote DR site.

Within the SnapManager Suite, SnapManager for Exchange offers the most comprehensive integration with SnapMirror, providing the ability to automatically replicate the consistent Snapshot copies by using SnapMirror. SnapManager for Exchange 5.0 also provides automated failover for Exchange environments.



## 7.1.2 Server Suite

The N series Manageability Server Suite includes SnapManager for Virtual Infrastructure and SnapDrive (for Windows, UNIX, and Linux).

SnapManager for Virtual Infrastructure provides storage and virtual infrastructure administrators with an automated solution for data protection and recovery of virtual machines in a VMware ESX environment. This is achieved by integrating N series Snapshot, SnapRestore, and SnapMirror for automated backup and recovery of data stores.

SnapDrive automates storage provisioning tasks and simplifies the process of taking error-free, host-consistent data Snapshot copies. SnapDrive provides a server-aware alternative to maintaining manual host connections to underlying N series storage systems. It reduces the risk of data disruption and increases storage management flexibility, delivering higher productivity and utilization.

SnapDrive for Windows and SnapDrive for UNIX can be used for managing (creating, deleting, and renaming) Snapshot copies on the source volume of SnapMirror. Any changes to Snapshot copies on the source system are immediately made visible on the destination system.

In general, SnapDrive is well integrated with volume SnapMirror. For example, SnapDrive for Windows can create a rolling Snapshot copy and then perform a volume SnapMirror update. SnapDrive for UNIX cannot perform any SnapMirror operations. Also, SnapDrive does not have integration with qtree SnapMirror, nor does it support qtree-level SnapMirror operations.

For more information, see the *IBM System Storage N series SnapDrive 6.0 for Windows Installation and Administration Guide*, GC26-7880-03, available on the IBM Storage support website:

<http://www.ibm.com/storage/support/nas>

## 7.2 FlexClone

Starting with Data ONTAP 7G, storage administrators have access to a powerful new feature that allows them to instantly create clones of a flexible volume. A FlexClone volume is a writable point-in-time image of a flexible volume or another FlexClone volume. These volumes take only a few seconds to create and do not cause interruption to the parent flexible volume. FlexClone volumes use space very efficiently, leveraging the Data ONTAP architecture to store only data that changes between the parent and clone volumes. FlexClone offers substantial space savings for work environments that require multiple copies of the same data, such as source trees, chip simulations, and weather simulations, without causing any performance bottlenecks.

FlexClone also makes it possible to create a writable volume from a read-only SnapMirror destination without interrupting the SnapMirror replication process and the production operations. A FlexClone volume can also be split from its parent to create a new standalone volume. Cloning is available only with flexible volumes, not with traditional volumes. Cloning does not require any special hardware. However, the storage system must have the *flex\_clone* license installed (see Figure 7-2).

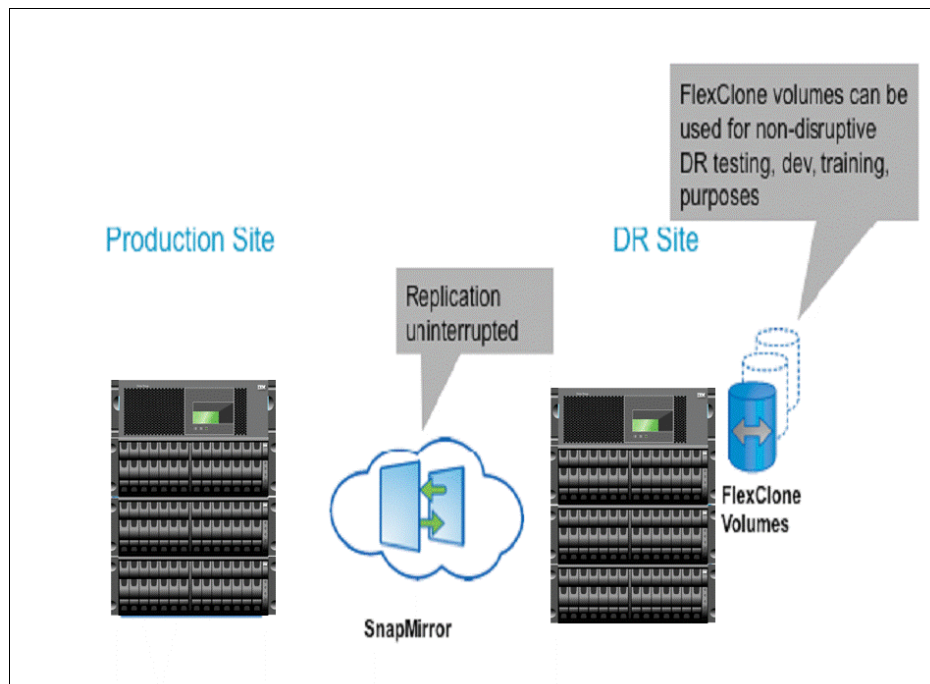


Figure 7-2 Non-disruptive DR testing, dev, training with SnapMirror and FlexClone

### 7.2.1 Volume SnapMirror, SnapDrive, and FlexClone

When a clone is created on a volume SnapMirror destination volume, Data ONTAP locks the Snapshot copy that the clone is based on. This Snapshot copy cannot be deleted. This is done to protect the clone. Figure 7-3 shows a Snapshot marked busy. The resulting error message is also shown after attempting to delete a locked Snapshot copy.

```

itsotuc4> snap list itso2_sm
Volume itso2_sm
working...

  %/used    %/total    date          name
-----
  0% ( 0%)  0% ( 0%)  Mar 04 10:07
itsotuc4(0101181370)_itso2_sm.6 (busy,snapmirror,vclone)
  0% ( 0%)  0% ( 0%)  Mar 04 09:43
itsotuc4(0101181370)_itso2_sm.5
  0% ( 0%)  0% ( 0%)  Mar 01 00:00  weekly.0
  0% ( 0%)  0% ( 0%)  Feb 28 00:00  nightly.0
  0% ( 0%)  0% ( 0%)  Feb 27 00:00  nightly.1
itsotuc4> snap delete itso2_sm itsotuc4(0101181370)_itso2_sm.6
Snapshot operation not allowed on a read-only volume.

```

Figure 7-3 SnapMirror marked for use by FlexClone

## Considerations for volume SnapMirror

There is also a soft lock on the corresponding Snapshot copy on the SnapMirror source system. Data ONTAP will not delete this Snapshot copy; however, the user can delete it on the SnapMirror source volume. If the user deletes the Snapshot copy on the source volume, the next SnapMirror update fails because it attempts and fails to delete the corresponding Snapshot copy on the destination volume. All SnapMirror updates to the destination volume continue to fail until the clone is destroyed or split. Use caution when deleting Snapshot copies when SnapMirror and FlexClone are involved.

Also, if a FlexClone volume is created from a Snapshot copy in the destination volume that is not the most recent copy, and so has locked down the Snapshot copy, if that Snapshot copy no longer exists on the source volume, every update attempts to delete the copy on the destination. In this case, all SnapMirror updates to the destination volume will fail until the clone is destroyed or split. This does not occur if the clone is created from the most recent Snapshot copy in the SnapMirror destination, because that copy still exists in the source volume.

SnapDrive for Windows creates rolling Snapshot copies on the source volume. When SnapDrive creates a new rolling Snapshot copy, it deletes the old rolling Snapshot copy. Therefore if a FlexClone volume is created on the SnapMirror destination using the SnapDrive rolling Snapshot copy, the next SnapDrive update will delete the corresponding Snapshot copy on the source volume and SnapMirror updates will fail from that point onwards. There are two ways around this issue. See the best practices below.

## Best practices for volume SnapMirror

These are some best practices for volume SnapMirror:

- ▶ Create FlexClone volumes on the destination from manually created Snapshot copies instead of scheduled copies.
- ▶ If you want to create FlexClone volumes on the destination from SnapDrive-created rolling Snapshot copies, do one of the following actions:
  - Perform a manual SnapMirror update following the creation of the FlexClone volume. This update process will propagate a soft lock on the corresponding Snapshot copy on the source system. The next SnapDrive update then creates another rolling Snapshot copy without deleting the Snapshot copy which has a soft lock associated with it.

- Rename the rolling Snapshot copy created by SnapDrive before creating the FlexClone volume. This step will help SnapDrive not delete this renamed Snapshot copy.
- ▶ Do not create FlexClone volumes on the destination using the volume SnapMirror Snapshot copies. If you have to create FlexClone volumes from volume SnapMirror Snapshot copies, use the latest SnapMirror Snapshot copy.
- ▶ Do not delete the Snapshot copies on the source if a FlexClone volume exists on the destination using the corresponding Snapshot copy.

## 7.2.2 Qtree SnapMirror and FlexClone

A few key behaviors are different when creating FlexClone volumes on a qtree SnapMirror destination system.

### Considerations for qtree SnapMirror

Qtree SnapMirror does not maintain the same Snapshot copies of the volume on the source and destination systems. Because of this characteristic, a FlexClone volume created from a Snapshot copy on the qtree SnapMirror destination does not cause a lock on that Snapshot copy on the source volume. Deleting that Snapshot copy on the source volume has no impact on the replication or the destination volume.

Therefore, the advantage of qtree SnapMirror is that a FlexClone volume can live for a long time on the SnapMirror destination system without space implications on the source system.

If a Snapshot copy is not specified when creating a FlexClone volume on the qtree SnapMirror destination volume, the **vol clone** command creates a new Snapshot copy on that volume.

If a FlexClone volume is created using the qtree SnapMirror baseline Snapshot copy, the qtree in the FlexClone volume will be writable.

If a FlexClone volume is created on the qtree SnapMirror destination volume without specifying a backing Snapshot copy for the clone creation, a separate SnapMirror relationship appears in the **snapmirror status** command output. Following is an example that demonstrates this behavior. In the example, there is a qtree SnapMirror relationship between `itsotuc1:/vol/vol1/qt1` and `itsotuc2:/vol/vol4/qt1`.

This is the **snapmirror status** output for the relationship:

```
itsotuc1:/vol/vol1/qt1 itsotuc2:/vol/vol4/qt1 Snapmirrored 2689:49:43
Transferring (294 MB done)
```

A FlexClone volume called **c2** is created from the parent volume `vol4`:

```
itsotuc2> vol clone create c2 -b vol4
```

A new relationship for the FlexClone volume appears in the **snapmirror status** command output:

```
itsotuc1:/vol/vol1/qt1 itsotuc2:/vol/c2/qt1 Snapmirrored 2689:51:31 Idle
itsotuc1:/vol/vol1/qt1 itsotuc2:/vol/vol4/qt1 Snapmirrored 2689:51:31 Transferring (1232
MB done)
```

Notice that the new FlexClone SnapMirror relationship does not impact the qtree SnapMirror relationship belonging to the parent volume.

The qtree in the FlexClone volume **c2** is read-only, whereas the FlexClone volume itself is writable.

In order to make the qtree in the FlexClone volume writable, quiesce and break operations must be performed on the FlexClone volume relationship. The original qtree SnapMirror relationship remains unaffected.

```
itsotuc2> snapmirror quiesce /vol/c2/qt1
itsotuc2> snapmirror break /vol/c2/qt1
snapmirror break: Destination /vol/c2/qt1 is now writable.
```

### Best practices for qtree SnapMirror

These are a few best practices for qtree SnapMirror:

- ▶ SnapMirror updates require a common SnapMirror Snapshot copy. Therefore, do not delete SnapMirror Snapshot copies on either the source or the destination system.
- ▶ FlexClone volumes are backed by the Snapshot copy from which they are created. The backing Snapshot copy is hard-locked (with a busy tag) and therefore cannot be deleted. After the FlexClone volume is destroyed, the lock is removed as well.

## 7.2.3 Splitting and destroying a clone

Splitting a FlexClone volume from its parent removes the connection between the clone and its parent. The administrator can split a FlexClone volume in a SnapMirror environment without affecting the SnapMirror transfer, because it becomes an independent entity after the FlexClone volume is split. In fact, it is a good idea to split clones that have been used for an extended period of time to avoid any impact on SnapMirror, especially if the source Snapshot copy could be deleted. Be aware that when the FlexClone volume is split, all existing Snapshot copies of the FlexClone volume are deleted.

If a FlexClone volume is not required any more, it can be directly destroyed without splitting it.

## 7.2.4 Invalid operations

FlexClone data resides in the parent Snapshot copy, so operations that would destroy the parent volume are not allowed. The following operations are *not* allowed:

- ▶ Destroying a parent volume (but it can be taken offline)
- ▶ Destroying a parent and clone shared Snapshot copy
- ▶ Using **vol copy** over a parent volume
- ▶ Using **snapmirror initialize** over a parent volume
- ▶ Using **snap restore** with a parent volume
- ▶ Using **snapmirror resync** with a parent volume before Data ONTAP 7.3. SnapMirror **resync** is possible to a parent volume in Data ONTAP 7.3, as long as the resync procedure does not delete the clone Snapshot copy.

**Attention:** A FlexClone volume cannot be used as a SnapMirror destination.

## 7.3 Setting up replication between FlexClone volumes

The previous section, 7.2, “FlexClone” on page 62, discusses use of FlexClone volumes on the SnapMirror destination system for DR testing and development. Some environments make use of FlexClone volumes on the source system to provide space-efficient copies of data for virtual machine clones for Virtual Desktop Infrastructure (VDI), data warehousing, local development and testing, and so on.

Figure 7-4 shows the use of FlexClone volumes on production system for test and development.

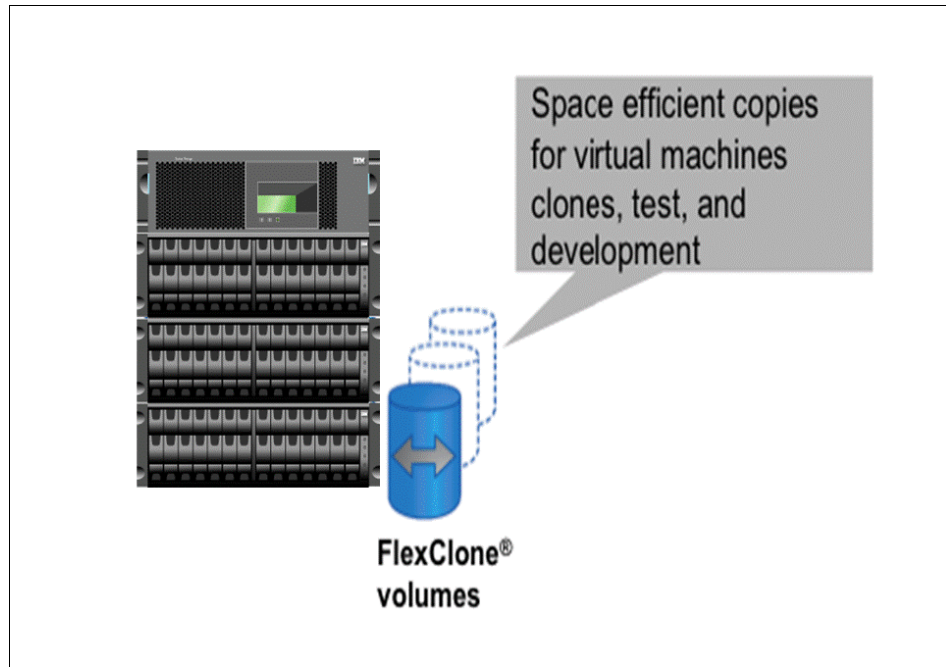


Figure 7-4 FlexClones on production system for test

In many instances, the space-efficient copies contain critical data on the production system that warrants replication for disaster recovery purposes. Before Data ONTAP 8.0.1 7-Mode (see Figure 7-5), when a FlexClone volume is replicated using volume SnapMirror, the FlexClone volume on the destination system requires additional capacity that is equal to the size of the parent volume.

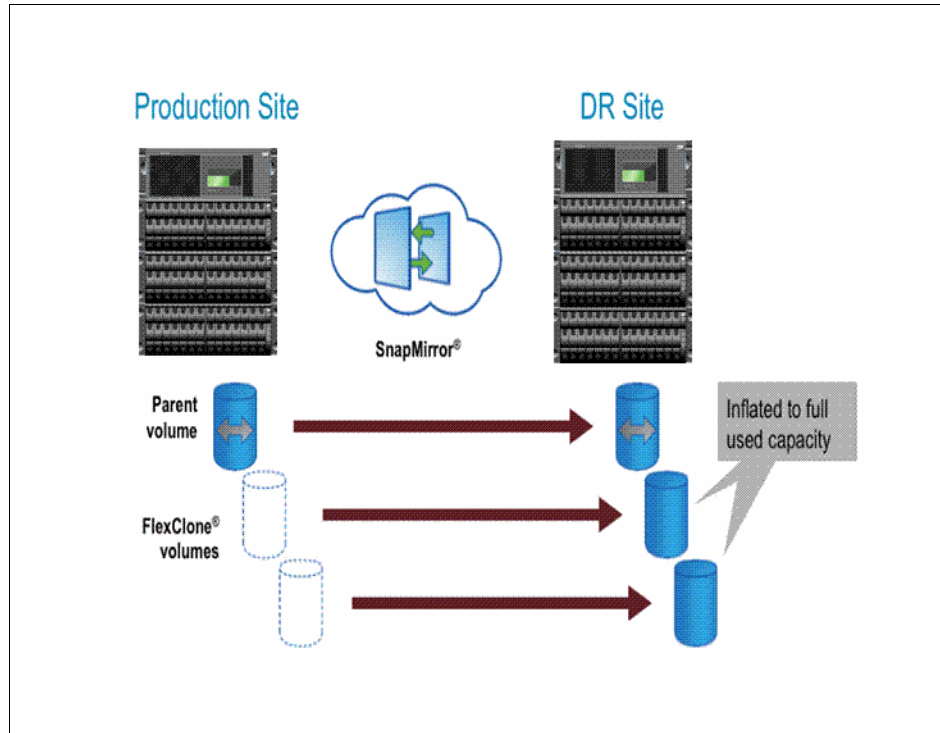


Figure 7-5 Space efficiency behavior when replicating FlexClone volumes before Data ONTAP 8.0.1 7-Mode

Starting with Data ONTAP 8.0.1 7-Mode, FlexClone volumes can be replicated using volume SnapMirror without the need for additional capacity on the destination system as long as the parent of the FlexClone volume is also replicated. The diagram in Figure 7-6 illustrates the space efficient attributes being propagated on the SnapMirror destination system.

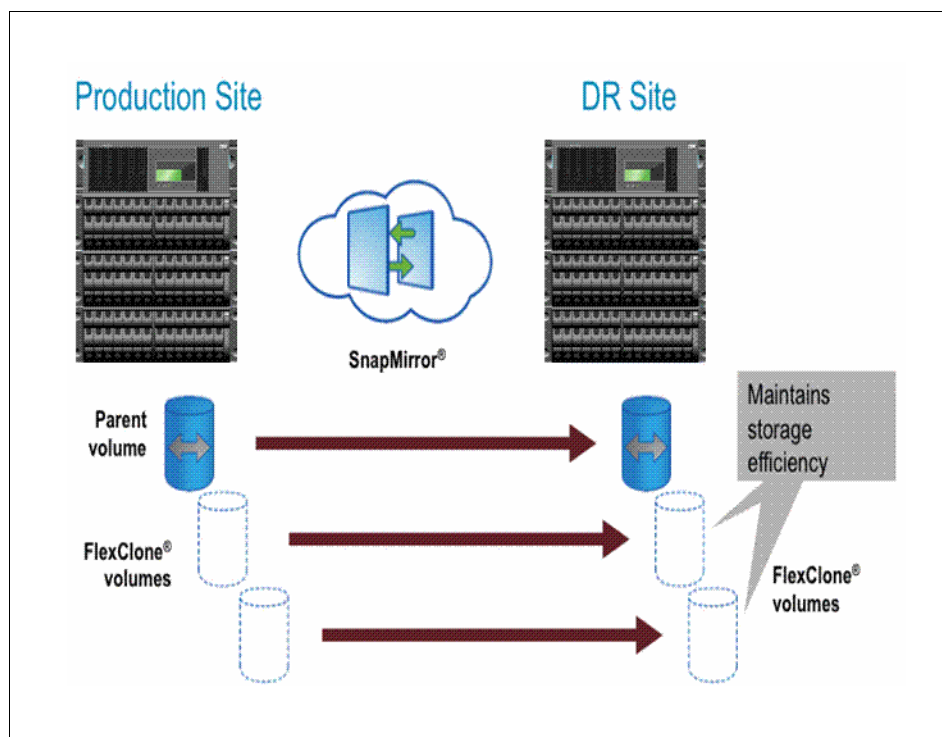


Figure 7-6 Space efficiency behavior when replicating FlexClone volumes starting Data ONTAP 8.0.1 7-Mode

In the following procedure, the steps described are required to ensure successful FlexClone volume replication while maintaining space efficiency. The procedure assumes that a parent volume is already replicated to the destination using volume SnapMirror and that FlexClone volumes already exist at the source only.

The source system fas1 has a parent volume vol1 and a clone volume cl\_vol1, which is based on the Snapshot copy snap1. The source volume vol1 is replicated to a destination volume vol2 on the system fas2:

1. Ensure that the base Snapshot copy of the FlexClone volume *snap1* exists at the destination. This can be done by performing a SnapMirror update operation on fas2 vol2.

```
fas2> snap list vol2
Volume vol2
working...
%/used %/total date name
-----
0% ( 0%) 0% ( 0%) Aug 27 14:25 fas2(0151745371)_vol2.6
0% ( 0%) 0% ( 0%) Aug 27 14:21 snap1
0% ( 0%) 0% ( 0%) Aug 27 14:20 fas2(0151745371)_vol2.5
```

2. Create a FlexClone volume cl\_vol2 on the destination system fas2 using the same base:

```
Snapshot copy snap1.
fas2> vol clone create cl_vol2 -b vol2 snap1
```



3. Establish a new SnapMirror relationship between the two FlexClone volumes using the SnapMirror resync command. The SnapMirror resync process establishes the SnapMirror relationship between the FlexClone volumes using snap1 as the SnapMirror baseline Snapshot copy. The process also updates the destination FlexClone volume with any changed or new blocks contained within the source FlexClone volume since the creation of snap1.

```
fas2> snapmirror resync -S fas1:cl_vol1 cl_vol2
```

4. Verify the FlexClone SnapMirror relationship with the SnapMirror status command.

```
fas2 > snapmirror status
Snapmirror is on.
Source Destination State Lag Status
fas1:cl_vol1 fas2:cl_vol2 Snapmirrored 00:00:18 Idle
fas1:vol1 fas2:vol2 Snapmirrored 00:02:41 Idle
```

A SnapMirror relationship for a FlexClone volume does not require initialization. After the initial resync as outlined in step 3, it is managed in the same manner as any other Flexible volume. The FlexClone volume SnapMirror relationship can be broken, updated, resynchronized in either direction (failover and failback) just like a SnapMirror relationship for a Flexible volume. The relationship can be imported into Protection Manager for management.

## 7.4 SnapVault

Whereas SnapMirror is typically used for disaster recovery purposes, SnapVault is used for long-term backup of production data or disaster recovery data.

### 7.4.1 Differences between SnapMirror and SnapVault

Table 7-1 gives a high-level view of the differences between SnapMirror and SnapVault.

Table 7-1 SnapVault and SnapMirror comparison

Feature	Volume SnapMirror	Qtree SnapMirror	SnapVault
Replication type	Physical	Logical	Logical
Replication network	FC or IP	FC or IP	IP only
Multiple paths for replication	Yes	Yes	No
Data ONTAP version sensitive?	Yes	No	No
RPO (How much data can I afford to lose?)	1 minute <sup>a</sup>	1 minute <sup>b</sup>	1 hour
Failover capability	Yes	Yes	Yes, when combined with SnapMirror
Snapshot retention for backup use	No	Possible but tedious	Yes
Snapshot coalescing	N/A	No	Yes
Failback resync	Yes	Yes	No

Feature	Volume SnapMirror	Qtrees SnapMirror	SnapVault
Deduplication	Destination inherits deduplication savings; network savings as well.	Destination does not inherit deduplication savings.	SnapVault and deduplication are integrated; destination does not inherit deduplication savings.

- a. Although 1-minute updates are possible, they are not desirable. Use SnapMirror Semi-Sync for low RPO (< 3 minutes).
- b. Although 1-minute updates are possible, they are not desirable. SnapMirror Semi-Sync cannot be used on standalone qtrees.

## 7.4.2 Two types of SnapVault deployments with SnapMirror

This section describes the deployment capabilities of SnapVault and SnapMirror

### DR protection for long-term backup data

In the configuration shown in Figure 7-7, data on various production systems requires long-term backup protection. SnapVault is used to achieve this requirement. In case of disaster, the long-term backups are replicated to a DR site by using volume SnapMirror.

Because volume SnapMirror copies all Snapshot copies, all the long-term backups at the SnapVault destination are available at the DR site. The data at the DR site can be used to restore or access the desired data in the event of partial data loss or an entire SnapVault system failure at the production site.

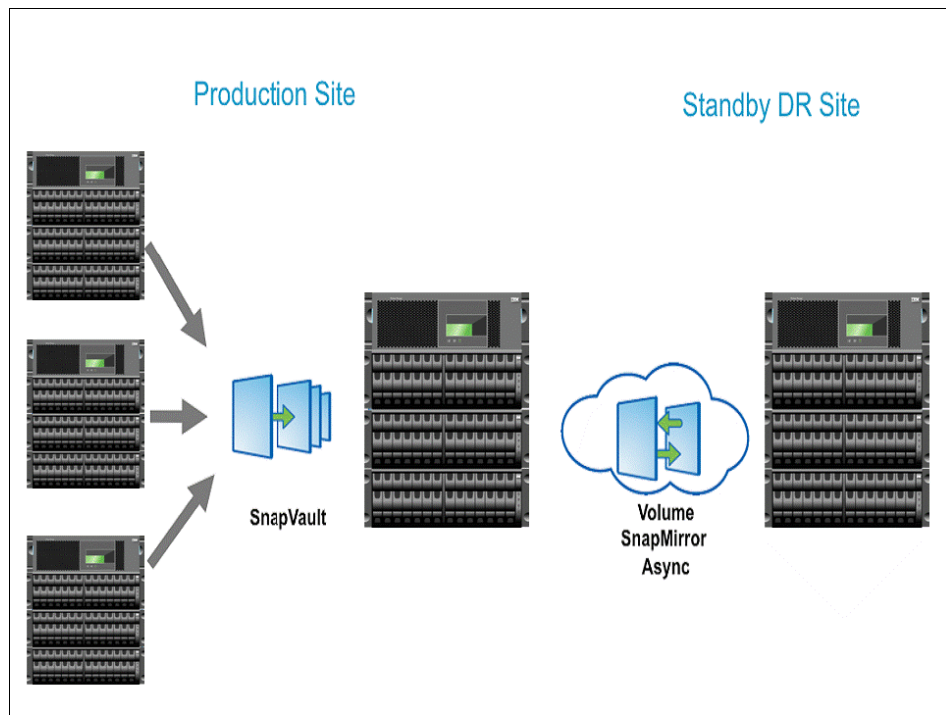


Figure 7-7 SnapVault and SnapMirror integration

## Long-term backup of primary or DR data

Figure 7-8 shows an illustration of backup protection for DR data.

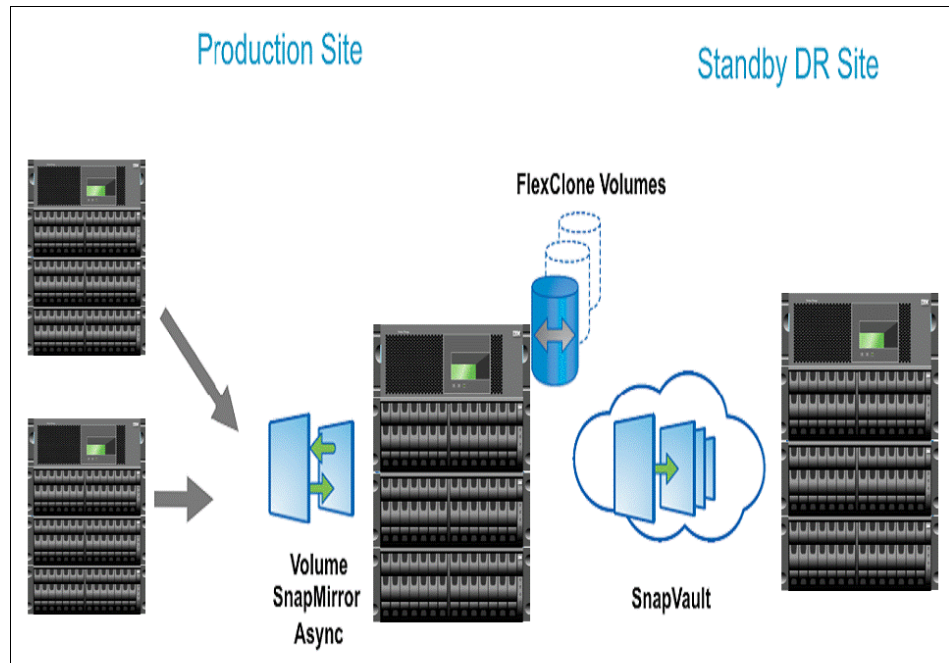


Figure 7-8 Example of backup protection for DR data

Volume SnapMirror provides identical data at the source and destination, which means that if 7-day retention is set up at the production site, data is retained for 7 days on the SnapMirror destination at the DR site as well. If long-term retention (say 90 days) is required at the DR site, SnapVault can be used to back up from the SnapMirror destination. Here is an example:

```
itsotuc1itsotuc2:vol1 (production volume) is being mirrored to itsotuc2:vol2
(DR volume) using volume SnapMirror.
itsotuc2:vol2/qt1 is being backed up to fas3:vol3/qt1 (long-term backup) by
SnapVault.
```

## Behaviors and best practices

This section covers some of the known behaviors and best practices:

- ▶ SnapVault cannot create Snapshot copies on the volume SnapMirror destination because the destination volume is read-only. However, SnapVault transfers the Snapshot copies from the volume SnapMirror destination. The type of Snapshot copies that SnapVault transfers from the volume SnapMirror destination depends on the Data ONTAP version.
  - For versions Data ONTAP 7.3.1 and earlier: SnapVault cannot transfer a specific Snapshot copy from the volume SnapMirror destination even when the `-s` option is used. SnapVault transfers the latest volume SnapMirror Snapshot copy. In this example, SnapVault transfers the latest volume SnapMirror Snapshot copy of vol2.
  - For versions Data ONTAP 7.3.2 and higher: SnapVault is able to back up a specific Snapshot copy from the volume SnapMirror destination using the `-s` option.
- ▶ It is not possible to run SnapVault and SnapMirror simultaneously in the scenario just described. If SnapVault updates are triggered while volume SnapMirror transfers are in progress, the volume SnapMirror transfers are aborted. Therefore, volume SnapMirror transfers must be suspended in order for SnapVault updates to occur. To avoid update failures, schedule SnapVault backups when volume SnapMirror updates are not scheduled to run.

For more information, see the *IBM System Storage N series SnapVault Best Practices Guide*, REDP4268.

## 7.5 SnapLock

SnapLock volumes are write-once, read-many (WORM) volumes intended for permanent data archiving. There are two types of SnapLock volumes:

- ▶ SnapLock Compliance volume: For strict regulatory environments, such as SEC 17a-4 compliant environments
- ▶ SnapLock Enterprise volume: For environments without regulatory restrictions SnapMirror can be used to mirror SnapLock volumes with the following restrictions:
  - Data ONTAP 7.0 supports both SnapLock and other than SnapLock destination volumes or qtrees.
  - In all Data ONTAP releases, the SnapMirror resync feature cannot be used to reestablish a volume SnapMirror relationship to a SnapLock Compliance destination volume because this operation would result in the destruction of WORM data on the destination volume and would make SnapLock noncompliant with government regulations regarding non-erasability of records. This important consideration must be kept in mind when planning DR scenarios for SnapLock Compliance volumes.
  - In the case of a qtree SnapMirror relationship to a SnapLock Compliance destination volume, the resync ability was available as a setflag prior to Data ONTAP 7.0. The resync option was generally available starting in Data ONTAP 7.0.
  - A SnapLock Compliance volume cannot be reinitialized because data on the volume cannot be changed. If the SnapMirror relationship is broken by using the snapmirror break command, the SnapLock Compliance destination volume can never be reinitialized. A new empty SnapLock Compliance destination volume can of course be reinitialized.
  - There is no such restriction on resync in the case of SnapLock Enterprise volumes because the administrator is trusted (see Table 7-3).
  - For SnapLock Compliance volumes, additional Data ONTAP version restrictions exist for source and destination systems for volume and qtree SnapMirror operations. Review the release notes and the product documentation for specific restriction details about the desired release.

Table 7-2 SnapMirror resync support

	SnapLock Compliance	SnapLock Enterprise
Qtree SnapMirror resync	Yes	Yes
Volume SnapMirror resync	No	Yes

### 7.5.1 Replication restrictions

Table 7-7 shows the restrictions for replication between non-SnapLock, SnapLock Enterprise, and SnapLock Compliance volumes.

Table 7-3 Replication restrictions between various types of volumes

		SnapMirror destination	
SnapMirror Source	Non-SnapLock volume	SnapLock Enterprise volume	SnapLock Compliance volume
Non-SnapLock volume	Yes	Yes	No
SnapLock Enterprise volume	Yes	Yes	No
SnapLock Compliance volume	Yes	Yes	Yes

## 7.5.2 End-to-end Snaplock Compliance

In Data ONTAP version 7.0 and later, in order to create an end-to-end SnapLock Compliance volume SnapMirror relationship, you simply create both the source and destination volumes as SnapLock Compliance volumes (by using the `-L` option) and then initialize the mirror by using `snapmirror initialize`, just as you would with regular volumes. No special steps are required.

SnapLock Compliance qtree SnapMirror relationships are initialized the same as regular volumes.

## 7.5.3 Synchronous replication

SnapLock Compliance does not yet support SnapMirror Sync and SnapMirror Semi-Sync. SnapLock Enterprise volumes do not have this restriction.

## 7.6 MultiStore

A storage system's hardware is made up of processors, network cards, power supplies, disk drives, and so on. Using MultiStore, the resources can be logically partitioned and dynamically assigned. The result is a virtual storage controller, also referred to as a vFiler controller. MultiStore technology provides an efficient architecture for consolidating multiple physical storage systems into a smaller number of systems. From the user's perspective, each virtual storage controller appears as a separate physical storage system with a unique IP address. Security is a key concern when storage is consolidated either within an organization or by an application service provider.

A virtual storage controller provides a confined environment. The data owned by a virtual storage controller cannot be accessed by any other virtual storage controllers, even though they are hosted on the same physical storage system. All requests for data access owned by a virtual storage controller are tagged with its context, making unauthorized access to data impossible.

### 7.6.1 Virtual storage controller DR

Through integration with SnapMirror, virtual storage controllers can be created and automatically mirrored to other storage systems over a LAN or WAN for the purposes of data migration and DR. Integrated mirror relationships can be established to automate the migration of virtual storage controllers to other storage systems. In a DR scenario a virtual storage controller on a destination storage system can be quickly activated in the event of an outage.

In a DR configuration, the source system remains active, serving data to its clients. The destination system remains inactive but ready to be activated in case of a disaster. It is best to have the disaster recovery site geographically farther from the source to recover from any site-wide disaster. The activation process must be performed manually (see Figure 7-9).

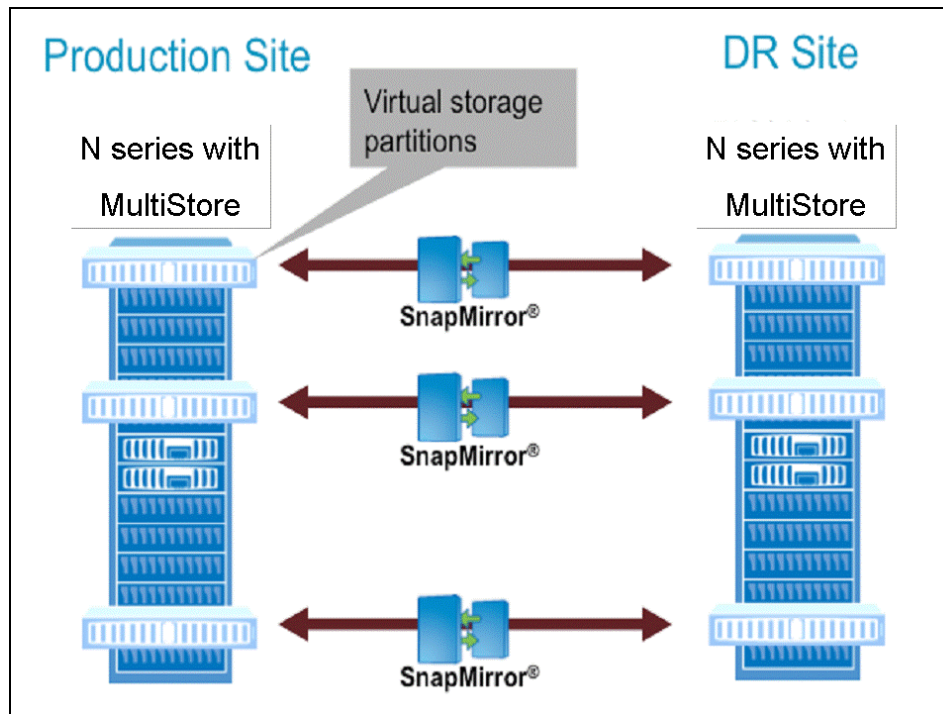


Figure 7-9 vFile for migration with SnapMirror

## 7.6.2 Virtual storage controller migration

Migration moves the specified virtual storage controller from the remote storage system to the local storage system. Migration is initiated on the destination storage system that will host the virtual storage controller after the migration. Migration across storage systems enables workload management. Migration automatically destroys the source virtual storage controller and activates the destination, which starts serving data to its clients automatically. Only the configuration is destroyed on the source, not the data.

The migration process takes more time than activating the DR destination site, because it has to perform a SnapMirror baseline copy of the data. Migration can also be used to perform hardware maintenance on the storage systems with minimum downtime. Depending on the amount of data, the baseline copy can take a long time. However, clients still have access to the source storage system during this copy phase. When the baseline copy has been completed, an incremental SnapMirror update is performed to make sure that all the new data since the baseline copy has been transferred to the new system.

The ability to replicate and move virtual storage controllers from one storage system to another provides the following benefits:

- ▶ SnapMirror can be used to replicate virtual storage systems to one or more target storage systems, where the mirrored virtual files can be quickly activated within minutes for DR purposes.
- ▶ Migrating or moving virtual files to less busy or more powerful systems allows administrators to easily load-balance user activity and to relocate data for optimum performance and efficiency.

- Data management is simplified because changes to network storage services can be quickly and transparently redeployed to suit business needs.

For more information, see *IBM System Storage N series with MultiStore and SnapMover*, REDP4170.

## 7.7 MetroCluster

MetroCluster is a cost-effective, integrated, high-availability and disaster recovery solution that protects against site failures resulting from human error, HVAC failures, power failures, building fire, architectural failures, and planned maintenance downtime. MetroCluster provides site protection within a metro, and supports replication up to 100 km. In some instances, campus DR might not be sufficient. In these scenarios, it is feasible to use SnapMirror in conjunction with MetroCluster to extend the protection over a long distance (see Figure 7-10).

**Tip:** SnapMirror replication can be performed only from the controller that has the data disk ownership.

Using FlexClone technology, DR testing can be performed at the DR site without interrupting production operations and SnapMirror replication. When the DR testing has been completed, these clones can be either split or destroyed.

For more information about how MetroCluster and SnapMirror can be used to achieve extended data protection, see the Redpaper™ publication, *IBM System Storage N series MetroCluster*, REDP4259.

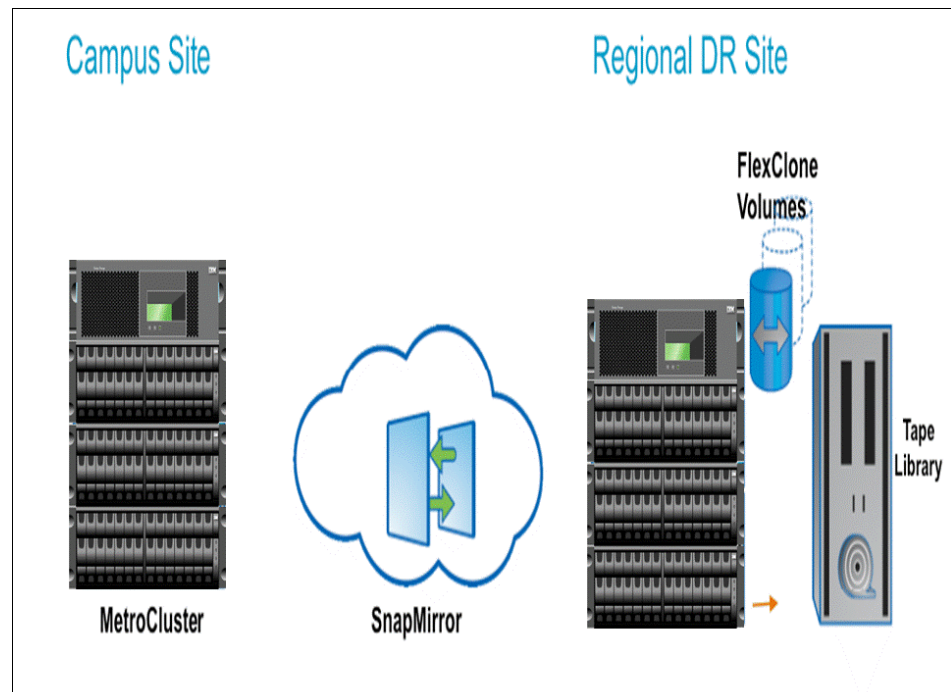


Figure 7-10 Campus and regional protection with MetroCluster and SnapMirror

## 7.8 FlexShare

FlexShare is a Data ONTAP software feature that prioritizes workload for a storage system. It prioritizes processing resources for key services when the system is under heavy load. FlexShare works on N series storage systems running Data ONTAP version 7.2 or later. The key features are relative priority of different volumes, per-volume user versus system priority, and per-volume cache policies. User operations are any data access operations that use NFS, CIFS, iSCSI, FCP, HTTP, or FTP. Examples of system operations are SnapMirror, SnapVault, WAFL scanners, SnapRestore, and NDMP.

The per-volume system setting affects all system activities, including SnapMirror operations. FlexShare treats all SnapMirror operations pertaining to a volume as a group, not as individual entities. For example, if a volume has many qtree SnapMirror relationships, the group of relationships is prioritized by FlexShare, not each individual relationship. In other words, FlexShare does not prioritize individual SnapMirror transfers; all SnapMirror transfers for a volume are prioritized together.

During high system load, storage administrators might want to prioritize user activity compared to system activity by minimizing SnapMirror operation impact. This can be accomplished by setting the system priority to be lower. Keep in mind that when the system priority is reduced, the amount of time that SnapMirror transfers can increase.

For more information about FlexShare, see the Redpaper publication, *IBM System Storage N series with FlexShare*, REDP4291.

## 7.9 Deduplication for N series

Deduplication for N series provides block-level deduplication within the entire flexible volume on N series storage systems. Deduplication only stores unique blocks in the flexible volume and creates a small amount of metadata in the process. This section discusses some best practices when deduplication is used along with SnapMirror.

SnapMirror takes a Snapshot copy before performing an update transfer. Any blocks in the Snapshot copy are locked and cannot be deduplicated. Therefore, for maximum space savings, run the deduplication process (by using the `sis start` command) before performing a SnapMirror update.

### 7.9.1 Volume SnapMirror

When the source volume is deduplicated, the destination volume automatically attains the deduplication savings. The deduplication benefits also extend to the bandwidth savings because volume SnapMirror only transfers unique blocks. The destination volume can only be deduplicated by replication from a deduplicated source volume; it cannot be deduplicated independently.

For example, a 100 GB source volume is deduplicated and now consumes 50 GB achieving 50% storage savings. Volume SnapMirror replication transfers only 50 GB of data, and the destination volume consumes only 50 GB.



## 7.9.2 Qtree SnapMirror

The source and destination volumes can be deduplicated independently. If the source volume is deduplicated, a qtree being replicated does not automatically result in space savings on the destination, and the replication does not result in bandwidth savings due to deduplication on the source. If deduplication savings are desired on the destination qtree, the deduplication process must be run independently on the destination volume.

For example, a 100 GB source volume with one source qtree of 100 GB data is deduplicated and now consumes 50 GB. Qtree SnapMirror replication still sends 100 GB of data, and the destination qtree consumes 100 GB of data. If deduplication is then run independently on the destination, its consumption will be reduced to 50 GB.

### Volume sizes

When deduplication is not enabled, the maximum volume size is 16 TB as of Data ONTAP 7.3. Deduplication for N series further restricts the volume sizes. The maximum volume size depends on the storage system model and Data ONTAP version. For example, as of Data ONTAP 7.3, maximum deduplication volume size is 6 TB in a N5600 system, and 16 TB in a N7800 system.

### Best practice

When volume SnapMirror is used with deduplication, ensure that the maximum deduplication volume size on both source and destination is the lower of the two maximum volume sizes. Therefore, in the above example, make sure a deduplication volume is no larger than 6 TB on both SnapMirror source and destination systems. This best practice helps with making sure that SnapMirror can be successfully resynchronized (by using the `snapmirror resync` command) in either direction (source to destination and destination to source in case of failback).

## 7.9.3 Data ONTAP 7.3 and volume SnapMirror

An N series deduplication volume contains associated metadata such as the fingerprint database and change log files. Before Data ONTAP 7.3, this metadata resided in the deduplicated volume. This resulted in two disadvantages:

- ▶ The metadata was replicated each time with the volume SnapMirror update.
- ▶ Every Snapshot copy contained the metadata, thus minimizing the space savings.

To overcome the above stated disadvantages, starting with Data ONTAP 7.3, the deduplication metadata has been moved out of the deduplicated volume and into the aggregate. Even though the metadata is not part of the volume, the destination volume is still in a deduplicated state and the data in the destination volume can be accessed just as in a regular volume. However, due to the absence of the metadata in the volume, there are some side effects to keep in mind in the failover and failback scenarios described next.

### Failover

In this scenario the destination (DR) volume is made writable for DR purposes. These are the deduplication characteristics for the DR volume:

- ▶ The DR volume is already in a deduplicated state because it is a volume SnapMirror destination volume for a deduplicated primary volume.
- ▶ Any new data written to the DR volume since the failover enjoys space savings within the new data itself but does not enjoy additional space savings with the old data. In other words, deduplication savings are not shared between the new and the old data.

- If deduplication savings are desired across the entire volume (new and old data), the deduplication metadata must be rebuilt by running the deduplication operation on the entire volume (by using the **sis start -s** command). The volume is available for read and write operations during the deduplication operation phase. For information about free space requirements in the aggregate and completion time for **sis start -s** command, see *IBM System Storage® N series A-SIS Deduplication*, SG24-7709, which is available at the following website:

<http://www.redbooks.ibm.com>

## Guidelines

If you plan to use the DR volume for production purposes for a short period of time (say < 1 month), a deduplication operation is not necessary on the DR volume to establish the metadata because the amount of new data might not be large and therefore result in potentially minimal additional space savings.

Some clients choose to run production operations six months at the primary and six months at the DR site. In these configurations, it is best to run the deduplication operation on the entire volume (by using the **sis start -s** command) upon failover to the DR site because the amount of new data would be significant and this could result in considerable space savings. The volume is available for read and write operations during the deduplication operation phase.

## 7.9.4 Failback

In this scenario the primary volume is made writable at the production site after a SnapMirror resync operation. It is also assumed that all the new data written at the DR site since failover is replicated from the DR volume to the primary volume.

### Deduplication characteristics

These are the deduplication characteristics for the primary volume:

- The primary volume is already in the deduplicated state because it is a volume SnapMirror destination volume for a deduplicated DR volume.
- Any new data written to the primary volume since the failback enjoys space savings with the old data that exists at the time of failover, but not with the data written from the DR volume.
- If deduplication savings are desired across the entire volume, the deduplication metadata must be rebuilt by running the deduplication operation on the entire volume (by using the **sis start -s** command). The volume is still available for read and write operations during the deduplication operation phase.

### Failback guidelines

If the amount of data written at the DR site is not significant, it is not necessary to run the deduplication operation on the entire volume upon failback to the primary site.

If the amount of data written to the DR site is significant such as in the scenario where primary and DR sites are used six months at a time, it is best to run the deduplication operation on the entire volume upon failback to the primary site.

## 7.9.5 Migration

In this scenario volume SnapMirror is used to migrate a volume between aggregates within a system or between two different systems. Upon migration, it is best to run the deduplication operation on the entire volume to rebuild the deduplication metadata (by using the **sis start -s** command) for maximum space savings.

For more information about N series deduplication and how it works, see *IBM System Storage N series A-SIS Deduplication*, SG24-7709, which is available at the following website:

<http://www.redbooks.ibm.com>





## Tips for troubleshooting

A brief list of things to remember when SnapMirror issues are encountered follows:

- ▶ SnapMirror log files are located in the `/etc/log` directory. The log files are `snapmirror`, `snapmirror.0`, `snapmirror.1`, and so forth.
- ▶ Make sure that the SnapMirror license is installed on both source and destination systems.
- ▶ Make sure that SnapMirror is enabled by using the **`snapmirror on`** command.
- ▶ If you are using names instead of IP addresses, make sure that the host names can be resolved.
- ▶ During initialization, the destination volume must be online and in a restricted state.
- ▶ The storage systems must be given permission in order for SnapMirror to work. Access is given with the **`options snapmirror.access`** command.
- ▶ The source volume must be online and writable.
- ▶ Volume SnapMirror requires the destination system's Data ONTAP version to be the same or higher than that of the source system.
- ▶ The destination volume must be same as or larger than the source volume for volume SnapMirror.

- The **snap delta** command can be used to calculate the changed data rate or amount without transferring the data (see Figure 8-1).

```
itsotuc4> snap delta itso_1

Volume itso_1
working...

From Snapshot  To                KB changed  Time          Rate (KB/hour)
-----
itsotuc4(0101181370)_itso_1.10 Active File System  0              14d 04:20  0.000
nightly.0      itsotuc4(0101181370)_itso_1.10 380              0d 10:25  36.437
hourly.0       nightly.0                    456              0d 06:00  76.000
hourly.1       hourly.0                    1622380          0d 06:00  270396.666
hourly.2       hourly.1                    3679032          0d 01:00  3679032.000
itsotuc4(0101181370)_itso_1.9 hourly.2          6244              0d 09:00  692.772
nightly.1      itsotuc4(0101181370)_itso_1.9 384              0d 01:59  193.261
nightly.2      nightly.1                    237908           1d 00:00  9907.673
weekly.0       nightly.2                    212640           0d 23:59  8860.102
nightly.3      weekly.0                     12748            0d 23:59  531.258
nightly.4      nightly.3                     1684             1d 00:00  70.166
nightly.5      nightly.4                    1574676          0d 23:59  65622.892
```

Figure 8-1 *snap delta example*

- Throttling can be used to limit the network bandwidth being used by SnapMirror.
- SnapMirror Sync and SnapMirror Semi-Sync cannot be used to replicate qtrees.
- SnapMirror Sync and SnapMirror Semi-Sync require an additional free license (*snapmirror\_sync*).
- SnapMirror Sync and SnapMirror Semi-Sync cannot be used to replicate within the same system or between systems within the same HA configuration.
- Performance data can be viewed by observing **systat**, **statit** and **perfstat**.



# Examples

This chapter provides some scenario examples with SnapMirror.

## 9.1 Failover and failback with SnapMirror

This section outlines the high-level steps required to perform planned and unplanned failover to the DR site. The steps also include a planned failback to the original production site. The steps assume that SnapMirror Async is being used for failover and failback. For the following scenarios itso1 is the primary storage system and vol1 is the primary volume; itso2 is the DR storage system and vol2 is the DR volume.

### 9.1.1 Planned failover (no disaster)

This example assumes that a disaster has not occurred.

#### Failover

This scenario assumes that there are ongoing SnapMirror updates between the production site and the DR site.

This is how the SnapMirror configuration file would look on itso2 (asynchronous volume SnapMirror updates every 30 minutes):

```
itso1:vol1 itso2:vol2 - 0,30 * * *
```

1. Shut down all applications at the production site.
2. Perform a final SnapMirror update to transfer all the changes to the DR site. Make the DR volumes writable by breaking the SnapMirror relationships:
  - a. On itso2: `snapmirror update -w vol2`
  - b. On itso2: `snapmirror break vol2`
3. Bring up the applications at the DR site. This assumes that all DNS changes, NFS and CIFS exports, and LUN mapping are completed.
4. Failover is now complete.

#### Replicating to the primary site

Because this is a planned failover, it is assumed that the data at the production site is intact at the time of failover:

1. Now that there is new data at the DR site, this data needs to be replicated back to the production site to prevent data loss in case of a disaster at the DR site. This is achieved with the **snapmirror resync** command. This is always done at the desired destination site; in this step, the production site. The resynchronization step sends only the changes since the last common Snapshot copy between the production and the DR sites.

```
On itso1: snapmirror resync -S itso2:vol2 itso1:vol1
```

2. Set up the primary site (now standby) for replication. The SnapMirror configuration file can be edited to add replication entries to perform this. After the configuration file is set up for asynchronous replication, SnapMirror performs asynchronous updates from the DR site to the primary site per the schedule specified in the configuration file. The SnapMirror configuration file on itso1 looks like this:

```
itso2:vol2 itso1:vol1 - 0,30 * * *
```



## Performing failback to the primary site

Follow these steps:

1. Shut down all applications at the DR site.
2. Perform a final SnapMirror update to transfer all the changes to the primary site. Make the primary volumes writable by breaking the SnapMirror relationship. This is always done at the destination; in this step, at the primary site.
  - a. On itso1: `snapmirror update -w vol1`
  - b. On itso1: `snapmirror break vol1`
3. Bring up the applications at the primary site. This assumes that all DNS changes, NFS and CIFS exports, and LUN mapping are completed.
4. Failback is now complete.

## Replicating to the DR site

Now that the primary site is active, there is new data at this site that needs to be replicated to the DR site.

1. This is achieved by using the **snapmirror resync** command. This is always done at the desired destination site; in this step, the DR site. The resynchronization step sends only the changes since the last common Snapshot copy between the primary and DR sites.

```
On itso2: snapmirror resync -S itso1:vol1 itso2:vol2 1
```

2. Set up the DR site (now standby) for replication by restoring the original SnapMirror configuration file. After the original configuration file is in place, the DR site (standby site) receives asynchronous updates from the DR site as per the specified schedule in the configuration file. The SnapMirror configuration file on itso2 looks like this:

```
itso1:vol1 itso2:vol2 -0,30 * * *
```

## 9.2 Failover in the event of a real disaster

This scenario assumes that the production site is lost and is not accessible.

### 9.2.1 Failover

1. Because the primary site is inaccessible, applications cannot be shut down. Therefore, make the DR volumes writable by breaking the SnapMirror relationships.

```
On itso2: snapmirror break vol2
```

2. Bring up the applications at the DR site. This assumes that all DNS changes, NFS and CIFS exports, and LUN mapping are completed.
3. Failover is now complete.

## 9.2.2 Replicating to the primary site

After the primary site is accessible, the first step is to determine whether the data is intact or lost.

1. If there is complete loss of data, the production site needs to be reinitialized (by using **snapmirror initialize**) from the DR site. The reinitialization is always performed at the destination site; in this case, the primary site. If there is no loss of data, only the changes can be transferred to the production site. This is achieved with the **snapmirror resync** command. This is always done at the desired destination site; in this step, the primary site. The resynchronization step sends only the changes since the last common Snapshot copy between the production and the DR sites.

- a. Data loss case. On itso1: `snapmirror initialize -S itso2:vol2 itso1:vol1`

- b. Data intact case. On itso1: `snapmirror resync -S itso2:vol2 itso1:vol1`

2. Set up the primary site (now standby) for replication. The SnapMirror configuration file can be edited to add replication entries to perform this. After the configuration file is set up for asynchronous replication, SnapMirror performs asynchronous updates from the DR site to the primary site per the schedule specified in the configuration file. The SnapMirror configuration file on itso1 looks like this:

```
itso2:vol2 itso1:vol1 - 0,30 * * *
```

## 9.2.3 Performing failback to the primary site

This section covers failback to the primary site from the DR site

1. Shut down all applications at the DR site.
2. Perform a final SnapMirror update to transfer all the changes to the production site. Make the production volumes writable by breaking the SnapMirror relationship. This is always done at the destination; in this step, at the production site.
  - a. On itso1: `snapmirror update -w vol1`
  - b. On itso1: `snapmirror break vol1`
3. Bring up the applications at the primary site. This assumes that all DNS changes, NFS and CIFS exports, and LUN mapping are completed.
4. Failback is now complete.

## 9.2.4 Replicating to the DR site

Now that the production site is active, there is new data at this site that needs to be replicated to the DR site.

1. This is achieved with the **snapmirror resync** command. This is always done at the desired destination site; in this step, the DR site. The resynchronization step sends only the changes since the last common Snapshot copy between the production and the DR sites.

On itso2: `snapmirror resync -S itso1:vol1 itso2:vol2`

2. Set up the DR site (now standby) for replication by restoring the original SnapMirror configuration file. After the original configuration file is in place, the DR site (standby site) receives asynchronous updates from the DR site as per the specified schedule in the configuration file. The SnapMirror configuration file on itso2 looks like this:

```
itso1:vol1 itso2:vol2 -0,30 * * *
```

## 9.2.5 Housekeeping

After the failback is completed, old SnapMirror relationships can be deleted by using the **snapmirror release** command. This command removes the relationships going from the DR storage system (itso2) to the production storage system (itso1). The **release** command is always run on the SnapMirror source system.

## 9.3 SnapLock and qtree SnapMirror resync

This section presents a couple of disaster recovery scenarios that require SnapMirror resync. Depending on the scenario, the appropriate storage system is chosen as the source system for the resync operation.

### 9.3.1 Production failure

In this scenario, the production system (nodeA) failed and the users are failed over to the DR site.

1. DR system (nodeB) is failed over for operations by breaking the SnapMirror relationship.
2. Users are now actively writing to the DR node (nodeB).
3. When the production site is back up, the data that has been written to the DR node during the outage needs to be transferred back to the production site. This requires the storage administrator to perform a **snapmirror resync** operation at the production site (nodeA).
4. This resync operation transfers all the data that has been changed since the last qtree SnapMirror transfer from the production site to the DR site.
5. When the data transfer has been completed, the SnapMirror relationship is broken to make the production site writable. The administrator can now redirect all users to the production site (nodeA), because it now has all the changes written at the DR site.
6. There is one last step that needs to be performed to place the SnapMirror relationship back to its original state. To start replicating data from the production site (nodeA) to the DR site (nodeB), **snapmirror resync** needs to be performed at the DR site (nodeB). This brings any new changes written to the production site after the failback operation. From now on, SnapMirror updates can be performed at desired intervals to continue to protect the data at the production site.

### 9.3.2 DR testing

In this scenario, there is no failure at the production site; the administrator simply wants to perform DR testing. This scenario assumes that users are actively accessing the production site (nodeA) while the DR testing is being done at the DR site (nodeB).

1. NodeA is a production system and nodeB is the DR system. There is a qtree SnapMirror relationship for a given qtree from nodeA to nodeB.
2. The user breaks the qtree SnapMirror relationship to make the qtree on nodeB writable for testing.
3. The user modifies data in the qtree on nodeB.
4. The user has now completed testing and wants to reestablish the qtree SnapMirror relationship to the original state; that is, start replication from the production site to the DR site. Therefore, the user issues a **snapmirror resync** on the DR node (nodeB).

5. The **resync** command overwrites any new data that was written on the DR system (nodeB). For SnapLock Compliance volumes, files can never be deleted before their expiration date; therefore, the resync operation saves all the changes made by the user to the qtree since the last successful qtree SnapMirror transfer.
6. The dump image is stored on the WORM volume where the qtree exists on the DR node (nodeB) in `/etc/logs/snapmirror_resync_archive/volname_UUID_qtree`.

A sample log file would be like this:

```
/etc/log/snapmirror_resync_archive/slcsec_1374c60e-44ba-11d9-99910050560669_
e7_qd
```

To later extract the data from the dump file, perform the following steps:

1. Using a UNIX or Windows client, create a directory called `temp` on the SnapLock volume; or create a new directory called `temp` and copy the dump file into this directory, giving the file the new name `dump file`. Although this is not necessary, it makes running the **restore** command much easier, because the leading path information from the dump file's original location is long.

2. To view files contained in the dump file, run the following command:

```
restore -tf /vol/<volume_name>/temp/dumpfile
```

3. To restore files contained in the dump file to their original location, run the following command:

```
restore -rfQ /vol/<volume_name>/temp/dumpfile
```

4. To restore files contained in the dump file to a different location, such as the `temp` directory where the dump file resides, run the following command:

```
restore -rfQD /vol/<volume_name>/temp/dumpfile/vol/<volume_name>/temp
```

Extracted files are in their original SnapLock state, regardless of the approach used.

5. If it is desirable to migrate the dump file to a different appliance, use two UNIX or Windows clients to transfer the file with a utility such as `ftp`.

## 9.4 Making the SnapVault destination writable

Perform the following steps to convert an Open Systems SnapVault or SnapVault secondary backup destination to a usable (writable) destination (typically for DR situations). All the commands are run on the SnapVault secondary (destination) system.

1. Secondary: Turn SnapMirror and SnapVault off.
2. Secondary: Switch to privileged mode (**priv set diag**).
3. Secondary: Convert SnapVault qtree to SnapMirror qtree (**snapmirror convert<sec\_qtree\_path>**).
4. Secondary: Turn SnapMirror on.
5. Secondary: Quiesce the qtree.
6. Secondary: Break the mirror, making it writable.
7. Secondary: Turn SnapVault on.

## 9.5 Migrating SnapVault by using SnapMirror

In this section we give an example of migrating a volume that contains SnapVault destination qtrees from one secondary system to another secondary system without having to perform another baseline transfer.

In this procedure, assume that a baseline of the bno:C:\500MB directory was backed up to itso-old:/vol/old\_vol/bno\_C\_500MB.

Complete the following steps:

1. Identify the SnapVault baselines of the qtrees or directories that need migration.
2. Using SnapMirror, replicate the volume from the present secondary system to a volume on the new secondary system. To replicate the old\_vol volume from the itso-old secondary system to the new\_vol volume on the itso-new secondary system, complete the following steps on the new secondary system (itso-new):
  - a. Create the new\_vol volume.
  - b. Restrict the new volume (new\_vol).
  - c. Transfer the **old\_vol** volume to the new\_vol volume by using SnapMirror initialization:  
**snapmirror initialize -S itso-old:old\_vol new\_vol.**

**Tip:** This is a SnapMirror replication of a volume, not a qtree.

3. Quiesce and break the SnapMirror relationship between the old secondary system and the new secondary system. To quiesce and break the SnapMirror relationship between itso-old and itso-new, complete the following steps on itso-new:
  - a. **snapmirror quiesce new\_vol**
  - b. **snapmirror break new\_vol**
4. Check the SnapMirror status and the SnapVault status on the new secondary. The SnapMirror state should show as *Broken-off*. The SnapVault state should show as *Snapvaulted*. Perform the following steps from itso-new:
  - a. **snapmirror status**  
Source Destination State  
itso-old:old\_vol itso-new:new\_vol Broken-off
  - b. **snapvault status**  
Source Destination State  
bno:C:\500MB itso-new:/vol/new\_vol/bno\_C\_500MB Snapvaulted
5. Confirm that SnapVault configuration information is not present on the new secondary system with the **snapvault status -c** command. Perform the following step from itso-new:  
**snapvault status -c**  
Snapvault secondary is ON.
6. Add SnapVault configuration information to the registry on the new secondary system with the **snapvault start** command. This does not start a new baseline; it updates the registry. Perform the following step from itso-new  
**snapvault start -S bno:C:\500MB itso-new:/vol/new\_vol/bno\_C\_500MB** Snapvault configuration for the qtree has been set. Qtree /vol/new\_vol/bno\_C\_500MB is already a replic

7. Confirm that SnapVault configuration information is present on the new secondary system with the **snapvault status -c** command. Perform the following step from itso-new:

```
snapvault status -c
Snapvault secondary is ON.
/vol/new_vol/bno_C_500MB source=bno:C:\500MB
```

8. Test the new SnapVault relationship by manually updating itso-new. Perform the following step from itso-new:

```
snapvault update itso-new:/vol/new_vol/bno_C_500MB
Transfer started.
Monitor progress with 'snapvault status' or the snapmirror log.
```

9. Recreate any schedules used on the old secondary system to the new secondary system and make sure that access permissions are in place.

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

## IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only:

- ▶ *Managing Unified Storage with IBM System Storage N series Operation Manager*, SG24-7734
- ▶ *IBM System Storage N series with Microsoft Clustering*, SG24-7671
- ▶ *IBM System Storage N series Hardware Guide*, SG24-7840
- ▶ *IBM System Storage N series Software Guide*, SG24-7129

You can search for, view, download or order these documents and other Redbooks publications, Redpaper publications, Web Docs, draft and additional materials, at the following website:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Other publications

These publications are also relevant as further information sources:

- ▶ *IBM System Storage N series Introduction and Planning Guide*, GA32-0543
- ▶ *IBM System Storage N series Platform Monitoring Guide*, GC27-2088
- ▶ *IBM System Storage N series Gateway Installation Requirements and Reference Guide*, GA32-1049-00

## Online resources

These websites are also relevant as further information sources:

- ▶ IBM System Storage N series Information Center:  
<http://publib.boulder.ibm.com/infocenter/nasinfo/nseries/index.jsp>
- ▶ Best Practices for Using the IBM System Storage N series Support Website:  
<https://www-304.ibm.com/support/docview.wss?uid=ssg1S7003279>
- ▶ Important information for N series support:  
<https://www-304.ibm.com/support/docview.wss?uid=ssg1S1003659>

## Help from IBM

IBM Support and downloads:

[ibm.com/support](https://ibm.com/support)

IBM Global Services

[ibm.com/services](https://ibm.com/services)



# Index

## Symbols

(LUNs) 10

## A

active-active configuration 21  
asynchronous modes 5  
asynchronous replication 18  
ATA 5

## B

backup operations 16  
backup windows 4  
baseline copy 18  
baseline Snapshot copy 13  
baseline transfer 18  
block-level updates 4  
busy lock 10

## C

cascade mirrors 10, 16  
centralized backup operations 4  
changed blocks 13, 18  
changed data blocks 14  
compressed blocks 24  
compression 24  
compression engine 24  
compression processing 24  
compression threads 24  
consistency points 20  
continuous backup applications 16  
CP Sync mode 20  
CPU contention 4  
CPU cycles 24

## D

data blocks 18, 24  
data consistency 4  
data loss 23  
Data ONTAP 13–14, 18  
Data ONTAP 7.2 5, 13  
Data ONTAP 7.2, 5  
Data ONTAP 7.2.x 13  
Data ONTAP 7.3 5, 13, 20  
Data ONTAP 7.3.2 5, 24  
Data ONTAP release 14  
data replicated 23  
data replication 23  
database server resources 4  
decompress 24  
decompression processing 24  
deduplicated 17  
designated qtree 14

destination compression engine 24  
destination file system 9, 14  
destination mirror 10  
destination qtree 17  
destination qtree. 16  
destination storage 20  
destination system 17, 20, 24  
destination systems 9  
destination volume 8, 11–12, 15, 17  
DR 3  
DR facility 2  
DR plan 4  
DR site 2, 23  
DR testing 3

## F

FC network 5  
FC networks 2  
Fibre Channel (FC) 2  
file system 18, 20  
FlexCache 4  
FlexClone 3  
flexible volumes 13

## I

inodes 14

## L

library investments 16  
load sharing 4  
load sharing implementation 4  
load-balancing transfers 22  
LREP tool 16  
LUN clones 17

## M

mirroring 2  
mirroring asynchronously 8  
multipath 22–23  
multipath connection 22

## N

N series 4, 10, 20  
N series storage systems 5  
N7800 5  
NearStore option license 5  
network bandwidth 23  
network bandwidth costs 23  
network compression 23  
network utilization 3  
NVLOG 20

## O

off-site storage 4  
options snapmirror.access 21

## P

partition network access 4  
performance goals 4  
point-in-time copy 14  
point-in-time image 18  
primary path 23  
production storage systems 4

## Q

qtree 8, 13–14, 16  
qtree initialization 15  
qtree SnapMirror 14–16, 18  
qtree-based replication 13  
qtrees 8, 14, 16

## R

read-only dataset 9  
read-only objects 8  
read-only volume 9  
recovery point objective 23  
Redbooks website 91  
    Contact us x  
remote destination system 9  
remote storage system 2  
replicate volumes 18  
replicating 8  
replication 4, 12, 23  
replication process 10  
replication relationship 22  
RPO 20, 23  
RPO values 23

## S

SAN 10  
Semi-Sync 20  
Semi-Sync mode 18  
semi-synchronous mode 20, 24  
SnapMirror 1–5, 7–8, 10, 13–24  
SnapMirror access 20  
SnapMirror Async 5, 17, 22  
SnapMirror Async mode 18  
SnapMirror Async technology 1  
SnapMirror commands 8  
SnapMirror configuration file 20  
SnapMirror deployment 10  
SnapMirror destination 8  
SnapMirror destination system 20–21  
SnapMirror native network 23  
SnapMirror network compression 24  
SnapMirror operations 21  
SnapMirror qtree 14  
SnapMirror qtree replication 14  
SnapMirror replication 17

SnapMirror Semi-Sync 5, 17, 20  
SnapMirror Snapshot copy 11  
SnapMirror Snapshot copy name 12  
SnapMirror software 12, 16  
SnapMirror source 8, 13, 16  
SnapMirror Sync 5, 18, 20, 22  
SnapMirror Sync mode 18  
SnapMirror transfers 5, 23–24  
SnapMirror update 5, 9  
SnapMirror volume replication 12  
snapmirror.access option 20  
Snapshot 10, 15  
Snapshot copy 5, 9–16, 18, 20  
Snapshot creations 20  
Snapshot technology 14  
SnapVault software 16  
SNMP 21  
source storage 21  
source storage system 18, 21  
source system 18  
source volume 8, 10, 14, 16–17, 20  
source volumes 8  
status query 19  
synchronous mode 19  
synchronous replication 19  
synchronously 18

## T

traditional backup 16

## V

visibility interval 20  
volume replication 11  
volume SnapMirror 12–14, 18  
volume SnapMirror source 13–14  
volume-based SnapMirror 13

## W

writable objects 8  
write latency 18









# N series SnapMirror Async Guide



**Redbooks®**

**Disaster recovery  
with SnapMirror**

**Data replication with  
SnapMirror**

**Application  
development with  
SnapMirror**

This IBM Redbooks publication presents an overview of implementing N series SnapMirror Async technology, with step-by-step configuration examples and guidelines to assist the reader in designing an optimal SnapMirror solution.

There are several approaches to increasing data availability in the face of hardware, software, or even site failures. Backups provide a way to recover lost data from an archival medium (tape or disk). Redundant hardware technologies also help mitigate the damage caused by hardware issues or failures. Mirroring provides a third mechanism to facilitate data availability and minimize downtime.

SnapMirror offers a fast and flexible enterprise solution for mirroring or replicating data over local area, wide area, and Fibre Channel (FC) networks. SnapMirror can be a key component in implementing enterprise data protection strategies. If a disaster occurs at a source site, businesses can access mission-critical data from a replica on a remote N series storage system for uninterrupted operation.

## **INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

### **BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
**[ibm.com/redbooks](http://ibm.com/redbooks)**

SG24-7993-00

ISBN 073843602X