NETAPP TECHNICAL REPORT

Using SnapMirror for Disaster Protection with Block Access Protocols

Deepak S N, NetApp
January 2009 | TR-3703

TABLE OF CONTENTS

1 INTRODUCTION

This document describes some of the fundamental concepts for implementing disaster protection solutions using synchronous/asynchronous NetApp® SnapMirror® software using block-access protocols (iSCSI, Fibre Channel). Synchronous/asynchronous SnapMirror refers to the operational mode of the SnapMirror software, and not to a separate software product or license.

In the traditional asynchronous mode of operation, updates of new and changed data from the source to the destination occur on a schedule defined by the storage administrator. These updates could be as frequent as once a minute, or as infrequent as once per week, depending on the user's requirement. NetApp has added a synchronous mode to the SnapMirror software, which sends updates from the source to the destination as they occur, rather than on a schedule. This makes sure that in the case of a system failure at the source site the data will be available on the destination site.

2 INTENDED AUDIENCE AND ASSUMPTIONS

This paper is intended as a guide for system and storage designers designing and deploying NetApp storage where resiliency and disaster recovery are among the requirements.

This report assumes familiarity with the following:

- General knowledge of NetApp hardware and software solutions, particularly in the area of block-access protocols such as Fibre Channel or iSCSI
- Host clustering concepts

This paper is based on the functionality available with Data ONTAP® release 7.1 or later.

3 REFERENCES

- Data Protection Online Backup and Recovery Guide on NOW™ for the appropriate Data ONTAP version (SnapMirror chapter)
- TR-3446: SnapMirror Async Overview and Best Practices Guide
- TR-3326: SnapMirror Sync and SnapMirror Semi-Sync Overview and Design Considerations

4 SNAPMIRROR COMMON OPERATIONS AND SETTINGS

This section describes some common operations performed for synchronous and asynchronous modes of replication. Refer to Figure 1 to understand all scenarios described in the sections that follow.

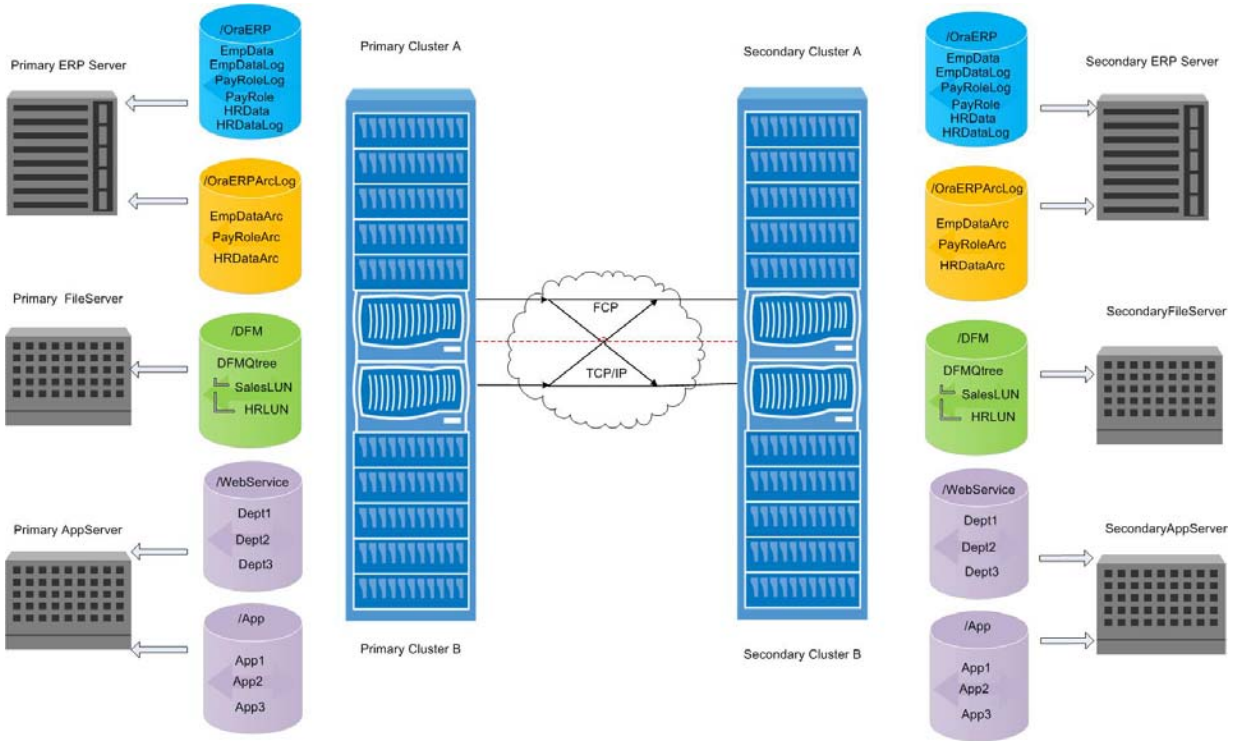


Figure 1) SnapMirror configuration on storage and server systems.

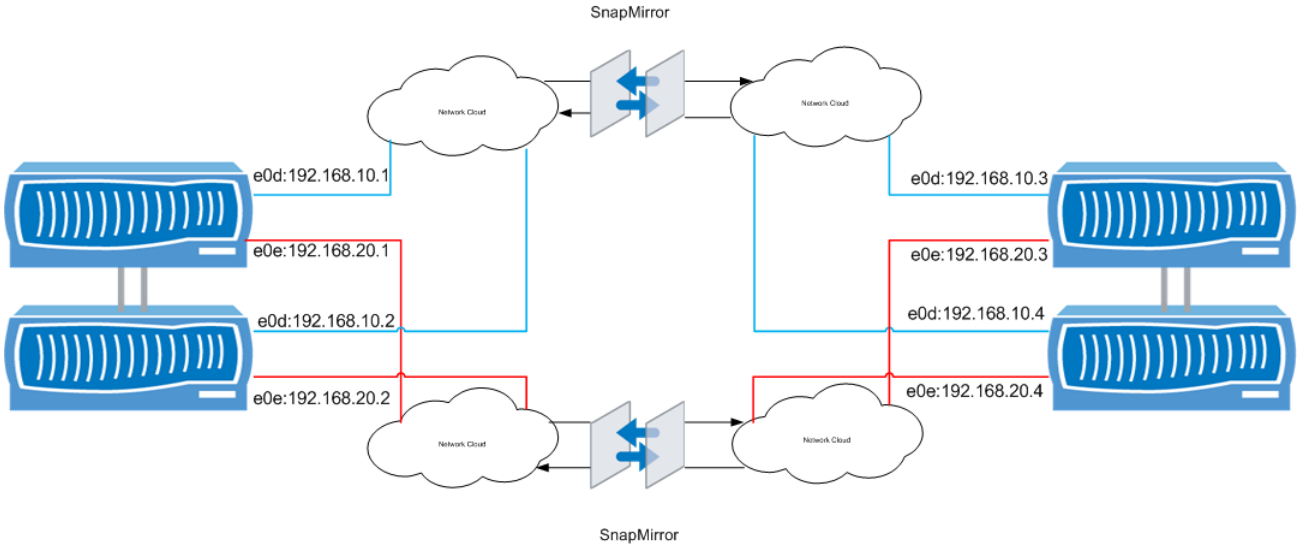


Figure 2) SnapMirror private network configuration.

Table 1) Logical view of storage design.

Storage Name	Roles	Volumes/Qtree	Aggregates	LUNs
PrimaryClusterA	SnapMirror Primary	/OraErp	OraERPAggr	EmpData EmpDataLog PayRole PayRoleLog HRData HRDataLog
		/OraErpArcLog	OraEEPArcLogAggr	EmpDataArc PayRoleArc HRDataArc
PrimaryClusterB	SnapMirror Primary	/DFM/DFMQtree	DFMAggr	salesLUN HRLUN
		/WebService	WEBAPPAggr	Dept1 Dept2 Dept3
		/App	WEBAPPAggr	App1 App2 App3
SecondaryClusterA	SnapMirror Secondary	/OraErp (read-only)	OraERPAggr	EmpData EmpDataLog PayRole PayRoleLog HRData HRDataLLog
		/OraErpArcLog (read-only)	OraEEPArcLogAggr	EmpDataArc PayRoleArc HRDataArc
SecondaryClusterB		/DFM/DFMQtree (Read/Write)	DFM_APPAggr	salesLUN HRLUN
		/WebService (read-only)	WEBAggr	Dept1 Dept2 Dept3
		/App (read-only))	DFM_APPAggr	App1 App2 App3

Table 2) SnapMirror replication mode design.

SnapMirror Primary Volume/Qtree	SnapMirror Secondary Volume/Qtree	Replication Mode
PrimaryClusterA:OraErp	SecondaryClusterA:OraErp	Synchronous SnapMirror
PrimaryClusterA:OraErpArcLog	SecondaryClusterB:OraErpArcLog	Asynchronous Volume SnapMirror
PrimaryClusterB:DFM/DFMQtree	SecondaryClusterB:DFM/DFMQtree	Asynchronous Qtree SnapMirror
PrimaryClusterB:Webservice	SecondaryClusterB:Webservice	Asynchronous Volume SnapMirror
PrimaryClusterA:App	SecondaryClusterB:App	Synchronous Volume SnapMirror

Table 3) SnapMirror private network and cluster failover configuration.

Storage Name	Ethernet Ports/IP Address	Partner IP Address (Cluster Failover)
PrimaryClusterA	e0d:192.168.10.1 e0e:192.168.20.1	e0d:192.168.10.2 e0e:192.168.20.2
PrimaryClusterB	e0d:192.168.10.2 e0e:192.168.20.2	e0e:192.168.10.1 e0e:192.168.20.1
SecondaryClusterA	e0d:192.168.10.3 e0e:192.168.20.3	e0d:192.168.10.4 e0e:192.168.20.4
SecondaryClusterB	e0d:192.168.10.4 e0e:192.168.20.4	e0d:192.168.10.3 e0e:192.168.20.3

Table 4) Server storage design.

SnapMirror Primary /Secondary Servers	LUNs Mapped (Read/Write)	Partner IP Address (Cluster Failover)	LUNs Mapped (Read Only)
PrimERPServer	EmpData EmpDataLog PayRole PayRoleLog HRDataHRDataLog EmpDataArc PatRoleArc HRDataArc	SecERPServer	EmpData EmpDataLog PayRole PayRoleLog HRDataHRDataLog EmpDataArc PatRoleArc HRDataArc
PrimFileServer	SalesLUN HRLUN	SecFileServer	SalesLUN HRLUN
PrimAppServer	Dept1 Dept2 Dept3 App1 App2 App3	SecAppServer	Dept1 Dept2 Dept3 App1 App2 App3

Table 2, Table 3, and Table 4 shows an example of the SnapMirror configuration settings. In the example, the SnapMirror relationships are configured on clustered FAS systems named PrimaryClusterA and PrimaryClusterB, where SnapMirror is the primary system controller and SecondaryClusterA and SecondaryClusterB are the controllers of the SnapMirror secondary systems.

- PrimaryClusterA and PrimaryClusterB are cluster systems that have SnapMirror relationship with clustered SecondaryClusterA and Secondary Cluster B systems. Figure 1 illustrates the physical network connectivity between both of the SnapMirror pairs that are configured to allow only SnapMirror network traffic.

  The physical distance between the primary and secondary SnapMirror systems must be accounted for when assessing the network link's "cable miles." Table 3 shows a logical view of the IP address layout in a private network that has been set up between SnapMirror clustered systems.
- PrimaryClusterA has two aggregates: OraERPAggr and OraERPARCLogAggr, with RAID-DP® protection. The `/OraERP` volume is created on OraERPAggr, which contains six LUNs. The `/OraERPARCLog` volume is created on OraERPARCAggr and contains three LUNs. All LUNs are mounted to PrimERPServer for application (read/write) access at the primary site.
- PrimaryClusterB has two aggregates: DFMAggr and WEBAPPAggr, with RAID-DP protection (see Table 1 and Table 4). The `/DFM` volume is a dfmQtree with two LUNs. The `/WebService` and `/App` volumes are created under WEBAPPAggr, each containing three LUNs mounted to PrimFileServer and PrimAppServer, respectively.
- SecondaryClusterA has two aggregates: OraERPAggr and OraERPARCLogAggr, with RAID-DP protection. The `/OraERP` volume is created on OraERPAgg and the `/ORAERPArcLog` volume is created on OraERPArcLogAggr. Both are set to restrict with no LUNs.

- SecondaryClusterB has two aggregates: DFMAggr and WEBAPPAggr, with RAID-DP layout. The /DFM volume does not have qtrees and LUNs. The `/WebService` and `/App` volumes are created under WEBAPPAggr with no LUNs and are set to restrict.
- After initializing the SnapMirror relationship between PrimaryClusterA with SecondaryClusterA and PrimaryClusterB with SecondaryClusterB (see Figure 6 and Table 2), the LUNs under SecondaryClusterA and SecondaryClusterB volumes are automatically created with read-only permissions. The `/OraERP` and `/ORAERPArcLog` volumes on the SecondaryClusterA and the permission on the `/WEBService` and `/App` volumes on the SecondaryClusterB are changed from restricted to read-only.
- SnapMirror performs a block per block copy, which includes Snapshot™ copies, language setting, and qtrees, from the PrimaryClusterA and PrimaryClusterB volumes to SecondaryClusterA and SecondaryClusterB volumes. Therefore, is important to assess the storage requirements on the primary and secondary SnapMirror systems (see section 8).
- The `/OraERP` volume on the PrimaryClusterA is synchronously mirrored to the `/OraERP` volume on the SecondaryClusterA. The /`OraERPARCLog` volume on PrimaryClusterA is asynchronously mirrored to the `/OraERPARCLog` volume on SecondaryClusterA (Table 2).
- The `/DFM/DFMQtree` qtree on the PrimaryClusterB is asynchronously mirrored to the `/DFM/DFMQtree` qtree on SecondaryClusterB. The `/WebService` volume on PrimaryClusterB is synchronously mirrored to the `/WebService` volumes on SecondaryClusterB. The `/App` volume on PrimaryClusterB is asynchronously mirrored to the `/App` volume on SecondaryClusterB (Table 2).

4.1 SNAPMIRROR BASIC SETUP CONFIGURATION

This section lists the prerequisites to set up the SnapMirror relationship. The examples and naming conventions are explained in section 5. For details, see the *Data Protection and Recovery Guide*.

Before replicating data, you must have proper licenses installed, and the primary and secondary storage systems must know where they are replicating data.

1. The `snapMirror.conf` control file defines the SnapMirror replication mode for volumes or qtrees between PrimaryClusterA with SecondaryClusterA, and PrimaryClusterB with SecondaryClusterB. For asynchronous replication mode, you can define transfer intervals for the next SnapMirror update by creating a schedule on an individual volume or qtree. If the `snapMirror.conf` file does not exist on the storage, create it and configure it only on SnapMirror secondary storage systems. You can also define transfer rate (KBs), restarts/tries, visibility interval, window size (wsize), and mode on each SnapMirror transfer.

   Note: For application consistency at the secondary site, NetApp highly recommends disabling scheduling in the `snapmirror.conf` file and drive replication from the host side using SnapDrive® and SnapManager® for various applications like Exchange, Oracle®, SQL Server®, SAP®, SharePoint®, and VMware®. It is highly recommended to use SnapManager, which completely integrates with various applications to provide rapid online backup and near instantaneous restoration by using the Snapshot technology.

   The following examples define the `snapmirror.conf` files configurations on SecondaryClusterA and SecondaryClusterB storage systems:

   a) The SecondaryClusterA storage system has multiple connections and failover connections defined with Ethernet interfaces connected with PrimaryClusterA e0d and e0e paired with SecondaryClusterA e0d and e0e Ethernet interfaces. The `OraERP` volume in PrimaryClusterA is synchronously replicated to the `ORAERP` volume with multiconnection enabled, and the `ORAERPARCLog` volume is asynchronously replicated to the `ORAERPARCLog` volume with failover connection enabled in SecondaryClusterA.

   b) There is no schedule defined for either volume. SnapMirror scheduling is not application-aware, and it transfers the changed data in the primary storage by creating Snapshot copies without stopping or quiescing application I/O. Therefore, it is recommended to control SnapMirror operations using SnapManager, which uses SnapDrive residing on the host to perform application-specific consistency operations. This report does not discuss operations specific to SnapManager. SnapManager is available for a variety of applications and provides end-to-end SnapMirror

integration with those applications. Section 5 describes how to control SnapMirror scheduling using SnapDrive commands.

SecondaryClusterA>wrfile /etc/snapmirror.conf

```
PrimaryClusterAOraERP = Multi(PrimaryClusterA-e0d, SecondaryCusterA-e0d)
(PrimaryClusterA-e0e,  SecondaryClusterA-e0e)

PrimaryClusterAOraERP:OraERP SecondaryClusterA:OraERP
visibility_interval=5m  – sync

PrimaryClusterAOraERPARCLog = Failover(PrimaryClusterA-e0e,
SecondaryCluster- e0e) (PrimaryClusterA-e0d,SecondaryCluster-e0d)

PrimaryClusterAOraERPARCLog:OraERPARCLog SecondaryClusterA:ORAERPArcLog
Kbs=8000 restart=always  -  -  -  -
```

c) The SecondaryClusterB storage system has multiple connections and failover connections defined with Ethernet interfaces connected with PrimaryClusterB e0d and e0e paired with SecondaryClusterA e0d and e0e Ethernet interfaces. The `App` volume in PrimaryClusterB is synchronously replicated to the `App` volume in SecondayClusterB with multiconnection enabled. The `Webservice` volume is asynchronously replicated to the `Webservice` volume in SecondayClusterB with failover mode enabled. The `/DFM/DFMQtree` qtree is replicated asynchronously to the `/DFM/DFMQtree` qtree with failover mode enabled in SecondaryClusterB. The `/DFM/DFMQtree` qtree is scheduled to replicate every one hour on all days except Saturdays and Sundays. The `Webservice` volume replicates at 2:00 a.m. and 11:00 p.m. every day for all the days in the month except Saturdays and Sundays. As the LUNs under the `/DFM/DFMQtree` and `/Webservice` volumes contain static data, the database does not have to be quiesced or shut down before performing SnapMirror transfers.

SecondaryClusterB> wrfile /etc/snapmirror.conf

```
PrimaryClusterBApp = Multi(PrimaryClusterB-e0d, SecondaryCusterB-e0d)
(PrimaryClusterB-e0e,  SecondaryClusterB-e0e)

PrimaryClusterBApp:App SecondaryClusterB:App visibility_interval=4m – sync

PrimaryClusterBfWebservice = Falilover(PrimaryClusterB-e0e,
SecondaryClusterB-e0e)

(PrimaryClusterB-e0d, SecondaryClusterB-e0d)

PrimaryClusterBWebservice:Webservice SecondaryClusterB: Webservice
Kbs=8000 restart=never tries=5000 – 15,22 – 1, 2,3,4,5

PrimaryClusterNDFM= failover(PrimaryClusterB-e0d, SecondaryClusterB-e0d)
(PrimaryClustere0e,SecondaryClusterB-e0e)

PrimaryClusterBDFM:DFM/DFMQtree SecondaryB:DFM/DFMQtree Kbs=5000
restart=always tries=5000 59 *  * 1-5
```

4.2 SNAPMIRROR INITIALIZATION

This section describes how to perform a complete (baseline) SnapMirror transfer of the volumes or qtrees specified in the `snapmirror.conf` file residing in PrimaryClusterA and PrimaryClusterB.

To do as baseline transfer, create a Snapshot copy of all the data on the `OraERP`, `OraERPARCLog`, `App`, and `Webservice` volumes, and the `DFMQtree` qtree residing on PrimaryClusterA and PrimaryClusterB.

Before performing this procedure, note the following:

- Create the volumes and qtrees in SecondaryClusterA and SecondaryClusterB with the same or higher size. These volumes must match the PrimaryClusterA and PrimaryClusterB aggregate names or size, use same RAID type, and have the disk type as FC or ATA. NetApp recommends having the same physical and logical configuration as the primary site, because, during failover to the secondary site, the applications access the same volumes. Sizing configuration mismatch at the secondary site might seriously affect application performance.

  For example, theORAERPAggr aggregate in the PrimaryClusterA is created with FC disks that have higher IOPS numbers, resulting in better performance. The SecondaryClusterB theORAERPAggr aggregate is created with ATA disks that have lower IOPS numbers, resulting in slower performance, which leads to significant performance issues during production.

- After creating volumes on the SecondaryClusterA and SecondaryClusterB, restrict the volume permission so that no application other than SnapMirror can access it. After SnapMirror initialization is completed, volume restrict permission is changed to read-only. All LUNs in these volumes are set to read-only. After SnapMirror break operation, LUNs under these volumes will have read/write permissions. In the qtree SnapMirror replication mode, SnapMirror automatically creates the `/DFM/DFMQtree` in the DFM volume. During SnapMirror initialization, the `DFMQtree` qtree is created under the `DFM` volume in SecondaryClusterA and B.

  ```
  SecondaryClusterA> vol restrict ORAERP

  SecondaryClusterA> vol restrict ORAERPARCLog

  SecondaryClusterB>vol restrict Webservice

  SecondaryClusterB>vol restrict App
  ```

- Execute the SnapMirror initialize operation on Secondary SnapMirror systems. When initialization operation is in progress, the `OraERP`, `ORAERPARCLog`, `Webservice`, `App`, and `DFM` volumes on theSecondaryClusterA and SecondaryClusterB cannot be accessed and appear as invalid in the `snapmirror status` output. The volumes in the PrimaryClusterA and PrimaryClusterB have read/write permission, and all the LUNs are online and mapped to the host without any I/O interruptions. During the initialize phase, SnapMirror creates a Snapshot copy of the volumes listed in the `snapmirror.conf` file on the PrimaryClusterA and PrimaryClusterB and transfers it to SecondaryClusterA and SecondaryClusterB. After the initialization operation is completed without any errors, all LUNs with Snapshot copies in the PrimaryClusterA and PrimaryClusterB volumes are replicated to SecondaryClusterA and SecondaryClusterB volumes. In case of the `DFMQtree` qtree, SnapMirror transfers only LUNs under `DFMQtree` and ignores other LUNs and Snapshot copies residing outside the qtree. It does not replicate igroup and LUN mapping information. If the transfer rate is not specified, the transfer rate (in KBs) specified in `snapmirror.conf` is taken by default. To change the transfer rate, see section 4.3.

  Note: The source specified must match the entry for `source_volume` in the `/etc/snapmirror.conf` file. If the entry does not match, the operation displays an error message and terminates.

  ```
  SecondaryClusterA>snapmirror initialize –S PrimaryClusterAOraERP:OraERP
  SecondaryClusterB:OraERP

  SecondaryClusterA>snapmirror initialize –S
  PrimaryClusterAOraERPARCLog:OraERPARCLog   SecondaryClusterB:OraERPARCLog

  SecondaryClusterB>snapmirror initialize –S PrimaryClusterBDFM:DFM/DFMQtree

  SecondaryClusterB:DFM/DFMQtree

  SecondaryClusterB>snapmirror initialize –S
  PrimaryClusterBWebservice:Webserivce –w
  ```

SecondaryClusterB:Webservice

SecondaryClustrerB>snapmirror initialize –S PrimaryClusterBApp:App SecondaryClusterB:App

- Prior to Data ONTAP 7.3, after SnapMirror volume replication initialization operation on the SecondaryClusterA and SecondaryClusterB specified in the `snapmirror.conf` file, the `fs_size_fixed` volume option can be turned off and space reservations option can be set to volume, file, or none. To size your thin provisioned SnapMirror volume, see section 7.
- After SnapMirror initialization, synchronous and asynchronous SnapMirror have different synchronization behavior due to the nature of the replication models. For synchronous SnapMirror, after initialization, each write to LUNs under the `ORAERP` volume in PrimaryClusterA is replicated to the SecondaryClusterA by forwarding the NVLOG and then the CP data under normal conditions. If the NVLOG or CP data is not consistent, synchronous SnapMirror falls back to asynchronous replication mode and then retries every minute to get back to the synchronous phase. The Volume SnapMirror asynchronous replication mode waits for the schedule time defined in the `snapmirror.conf` file to replicate the changed data in the from of Snapshot copies under the LUNs in the `Webservice` volume on PrimaryClusterB to SecondaryClusterB, or it can also be controlled by a manual `SnapMirror update` operation.

4.3 SNAPMIRROR MAXIMUM TRANSFER RATE

This section describes how to set the SnapMirror transfer rate and manually change it for ongoing transfers.

SnapMirror transfer rate specifies the maximum network bandwidth utilized for each transfer. The transfer rate is specified in the `snapmirror.conf` file at SecondaryClusterA and SecondaryClusterB.

In the following examples, the total network link used is a 10Mb T1 line and is configured for multiconnection or failover SnapMirror modes as explained in section 5 for high availability.

Synchronous SnapMirror replication has no effect on transfer rate as it replicates writes that occur in PrimaryClusterA or PrimaryClusterB to SecondaryClusterA or SecondaryClusterB.

The `ORAArcLog` volume has a transfer rate of 8000KBs or 8MB defined in SecondaryClusterA `snapmirror.conf` file. This means that it can utilize 8MB of network bandwidth in a 10MB network link. When multimode is enabled, SnapMirror provides load balance between the two links. The two lines between PrimaryCluster A and B and SecondaryCluster A and B now provides a combined network bandwidth of 20MB. Therefore, the `ORAArcLog` volume will have an effective combined bandwidth of 16MB for the transfer; however, the TCP/IP overheads should also be considered.

You can change SnapMirror maximum transfer rate for ongoing transfers; however, this is value is only for the current transfer. When the next scheduled transfer occurs, it uses the transfer value defined in the `snapmirror.conf` file.

For example, suppose there is a disk failure in PrimaryClusterB where the Webaggr aggregate is created in the `Webservice` volume. The ongoing scheduled SnapMirror transfer has the maximum transfer rate of 5,000KBs as defined in the `snapmirror.conf` file and utilizes 5MB network pipe for a total 20MB network link. As it is configured with the SnapMirror failover mode, the network functions as active/passive, and the secondary link will not be used unless it detects a primary link failure.

To speed up the ongoing SnapMirror transfer, which has delta data changed of 5GB with 5Mb link, it takes 16 minutes to replicate changed data.

The SnapMirror throttle operation can be executed at the PrimaryClusterB or SecondaryClusterB. If the transfer rate is changed to 10,000KBs, the ongoing transfer takes approximately two minutes for the command to take effect. To replicate 5GB changed data it takes 8.3 minutes, which reduces the risk by half when using 5Mb to 10Mb link.

```
SecondaryClusterB>snapmirror throttle 10000 Webserivce
PrimaryClusterB>snapmirror throttle 10000 SecondaryClusterBWebService:Webservice
```

You can also run `snapmirror update` command with `-k` option; however, it cannot change the ongoing SnapMirror transfer. It will start a new transfer. For more information on controlling SnapMirror updates through host for providing application-restartable consistent data inSecondaryCluster A and B, see section 5.

SnapMirror maximum transfer rate can be set at the system-level. System-level transfers can be defined on PrimaryClusterA and PrimaryClusterB, and SecondaryClusterA and SecondaryClusterB. The system-level

transfer rate overrides the individual transfer rate if incoming and outbound individual transfer rate exceeds the system-level transfer rate.

4.4 SNAPMIRROR MANUAL UPDATE

This section describes how to replicate changed data blocks on the PrimaryClusterA and PrimaryClusterB to SecondaryClusterA and SecondaryClusterB, which have asynchronous replication mode enabled. SnapMirror update operation should be controlled using the SnapDrive application to create restartable replicated data at the secondary site. SnapDrive for Windows® (SDW) on the hosts is integrated with SnapMirror update operation to replicate changed data blocks in the primary site by creating consistent rolling Snapshot copies that are file system aware and can be scripted. SnapDrive for UNIX® (SDU) does not provide any integration with the `snapmirror update` operation for the current release.

This section explains how the Perl script can be used for customers that do not have end-to-end integration to drive the `snapmirror update` operation for providing consistent replicated data in SecondaryClusterA and SecondaryClusterB.

Note: NetApp recommends using SnapManager for various applications such as Microsoft® Exchange, Oracle, VMware, and SQL Server to provide end-to-end integration with SnapMirror in failover and failback operations.

- SnapMirror update operations should be invoked only on the destinations in the SecondaryClusterA or SecondaryClusterB systems. Volume SnapMirror or Qtree SnapMirror replication modes only support SnapMirror update operations.
- After performing the initial SnapMirror initialization operation, common SnapMirror Snapshot copies exist on PrimaryClusterA and SecondaryCluster A. The `SnapMirror update` command when invoked on the SecondaryClusterA for `ORAERPArcLog` volume where the schedule is not defined transfers only changed data blocks to the `ORAERPArcLog` volume on SecondaryClusterA. When the `snapmirror update` operation completes without errors, the `ORAERPARCLog` volume in PrimaryClusterA retains the current Snapshot copy and deletes the previous Snapshot copy. On the SecondaryClusterA, the `ORAERPArcLog` volume retains both the current and the previous SnapMirror Snapshot copies.

  For example: Suppose the SnapMirror Snapshot copy on PrimaryClusterA for the `ORAERPArcLog` volume after initialization is `SecondaryClusterA(0012334499)_ORAERPArcLog.0 (SnapMirror)` and on the SecondaryClusterA for the `ORAERPArcLog` volume is `SecondaryClusterA(0012334499)_ORAERPARCLog.0`.

  When a `snapmirror update` operation is performed on the SecondaryClusterA, it creates a SnapMirror Snapshot copy named `SecondaryClusterA(0012334499)_ORAERPArcLog.1` in the `ORAERPArcLog` volume in PrimaryClusterA. It compares this Snapshot copy with `SecondaryClusterA(0012334499)_ORAERPArcLog.0` and transfers the changed data blocks. After the update is completed without errors, SecondaryClusterA has: `SecondaryClusterA(0012334499)_ORAERPArcLog.0` and `SecondaryClusterA(0012334499)_ORAERPArcLog.1`. On the PrimaryClusterA, the `OraERPArcLog` volume has: `SecondaryClusterA(0012334499)_ORAERPArcLog.1(SnapMirror).`

  The `PrimaryClusterA(0012334499)_ORAERPArcLog.0(SnapMirror)` Snapshot copy is deleted.

  Note: The SnapMirror Snapshot copy naming convention is as follows:

  Destination system name(serial number)_volume name

  For example, `SecondaryClusterA(0012334499)_ORAERPARCLog.0` on SecondaryClusterA and `SecondaryClusterA(0012334499)_ORAERPARCLog.0(SnapMirror)` on PrimaryClusterA.

  `SecondaryClusterA>snapmirror update –S PrimaryClusterAOraERPARCLog:OraERPArcLog SecondaryClusterA:OraERPArcLog`

- The Qtree SnapMirror replication mode only replicates LUNs under the `DFM/DFMQtree` qtree residing in the `DFM` volumes in PrimaryClusterB. It does not replicate LUN Snapshot copies or clone LUNs to `DFM/DFMQtree` under `DFM` volume in SecondaryClusterB.

For example, the DFMQtreesnapprim Snapshot copy created in the DFM volume in PrimaryClusterB system is transferred to the DFM volume in SecondaryClusterB. LUNs under the DFM volume in PrimaryClusterB is not transferred to the DFM volume in SecondaryClusterB as part of qtree snapmirror update.

Using the snapmirror update operation, you can transfer Snapshot copies by using the –s *primary Snapshot copy name* and –c *secondary Snapshot copy name* options as follows.

```
SecondaryClusterB>snap create -V DFM DFMQtreesnapprim
SecondaryClusterB>snapmirror update -s DFMQtreesnap -c DFMQtreesnapsec -S
PrimaryClusterBDFM:DFM/DFMQtree SecondaryClusterB:DFM/DFMQtree
```

4.5 SNAPMIRROR RESTART

This section describes when to use the restart and retries options and how to manually clear checkpoints when there is a restart pending on the SnapMirror transfer.

In the SecondaryClusterA and SecondaryClusterB systems, the restart and retries options are set in the snapmirror.conf file as shown in section 5.

The restart function has two options, "always" and "never," which can be set for each SnapMirror individual transfer. The default option is set to "always."

If the restart option is set to "always" for every five minutes after SnapMirror transfer time, or if the changed data is more than 3GB in size, SnapMirror automatically creates a manual checkpoint.

In case of a network outage, or if SecondaryClusterA and SecondaryClusterB is not able to accept SnapMirror data after transfer is started, and if the first checkpoint is five minutes past the schedule defined in the snapmirror.conf file or snapmirror update operations, SnapMirror continues the transfer from the last checkpoint. It does not create a new Snapshot copy but waits for the existing changed data to be transferred.

If restart option is set to "never," for the next schedule or update SnapMirror clears the checkpoint, creates a new SnapMirror Snapshot copy, and transfers the changed data.

For example, suppose on SecondaryClusterA snapmirror.conf file for the ORAERPARCLog SnapMirror transfer, the restart option is set to "always" by default. If a checkpoint occurs, and the next transfer schedule starts, which is driven through the host by the snapmirror update command, SnapMirror does not create a new SnapMirror Snapshot copy on the on ORAERPARCLog volume on PrimaryClusterA till it completes the last data transfer.

If the restart option is set to "never," then the checkpoint does not occur. For next schedule or update operation, a new SnapMirror Snapshot copy is created to transfer the changed data blocks in the Webservice volume on PrimaryClusterB.

Checkpoints can be forcefully cleared by issuing a snapmirror abort command using the –h option only on the SecondaryClusterA or SecondaryClusterB. This command stops the ongoing transfer, clears the restartable transfer log, and allows the next SnapMirror schedule or update operation to create a fresh Snapshot copy in PrimaryClusterA or PrimaryClusterB.

For example, if the ORAERPARCLog **volume** in SecondaryClusterA has a checkpoint created, clear it by entering:

```
SecondaryClusterA>snapmirror abort -h SecondaryClusterAORAERPARCLog:ORAERPARCLog
```

Suppose the ongoing transfer is checkpointed due to network link outage, and the rate of data changed on the ORAERPARCLog volume on PrimaryClusterA is 80% to the last checkpointed SnapMirror transfer. It is recommended to abort the current transfer and wait for the next schedule or manually perform the update to clear the existing checkpointed transfer. Then, create a new Snapshot copy, as it is not required to transfer the changed data blocks, which will again be transferred during the next update.

4.6 SNAPMIRROR QUIESCE AND RESUME OPERATIONS

This section describes how to stop SnapMirror replication for each transfer and manually control resumed transfers. These two operations can be invoked on both synchronous and asynchronous SnapMirror replication modes.

The SnapMirror quiesce operation allows the in-progress transfers to SecondaryClusterA and SecondaryClusterB to complete, after which it stops new transfers.

If there is a planned maintenance network outage on the PrimaryClusterA or PrimaryClusterB, you can manually suspend all the SnapMirror transfers. After SnapMirror quiesce operation, the next schedule or manual update operation will not start the SnapMirror transfer. It can only occur after the resume operation is performed on the volumes or qtrees. When quiesced, the synchronous SnapMirror replication mode switches to asynchronous mode, waits for resume operation, and performs the normal asynchronous replication. When all the data is synchronized, it performs NVLOG and CP forwarding and falls back to synchronous mode. This operation has to be performed on the PrimaryCluster systems.

```
PrimaryClusterB>snapmirror quiesce App

PrimaryClusterA>snapmirror quiesce ORAERPARCLog
```

During quiesce state, the write operation is allowed in the PrimaryClusterA and PrimaryClusterB SnapMirror volumes and qtrees. After the quiesce operation is completed, a common Snapshot copy exists between PrimaryCluster and SecondaryCluster A and B systems, and all SnapMirror transfers are stopped.

The `snapmirror resume` operation can be performed only on quesiced transfers. After this operation is completed, it allows the next SnapMirror schedule or update operation to create a new SnapMirror Snapshot copy at the PrimaryClusterA or PrimaryClusterB to find changed data blocks that are part of SnapMirror transfers to SecondaryClusterA or SecondaryClusterB.

```
Primary ClusterB>snapmirror resume App

PrimaryClsuterA>snapmirror resume ORAERPARCLog
```

4.7 SNAPMIRROR ABORT OPERATION

This section explains how to stop the ongoing SnapMirror transfer, which is started as part of schedule or manual update operations, or options that put the transfer into restart mode.

Suppose a SnapMirror transfer occurring in SecondaryClusterB for the `/DFM/DFMQtree` qtree as defined in the `snapmirror.conf` file has the `restart` option set to "`always`." Suppose the data on `/DFMDFMQtree` on PrimaryClusterB, which is part of SnapMirror transfer to `/DFM/DFMQtree` on SecondaryClusterB, is corrupted, the `snapmirror abort` operation can be issued at the PrimaryClusterB to stop the current transfer. This creates a checkpoint and waits for next schedule or manual update to start the transfer from the checkpoint time. If you do not want to restart the checkpoint and stop the entire transfer, use the `-h` option, which is a hard abort and cannot be restarted. This is because we can break the `DFM/DFMQtree` qtree SnapMirror at the SecondaryClusterB and allow applications to access LUN created on SecondaryClusterB `DFM/DFMQtree` qtree that is not corrupted as the SnapMirror transfer is aborted and then perform break and resync operation on SecondaryClusterB to transfer the clean data to PrimaryClusterB `DFM/DFMQTree` qtree. If `-h` option is not specified for the next schedule or manual update, SnapMirror restarts from the checkpoint time and replicates the corrupted data to SecondaryClusterB `DFM/DFMQtree` qtree.

```
SecondaryClusterB>snapmirror abort -h SecondaryClusterB:/DFM/DFMQtree
PrimaryClusterB>snapmirror abort DFM/DFMQtree SecondaryClusterB:/DFM/DFMQtree
```

If the abort operation is performed twice on SnapMirror transfer, then checkpoint will be cleared. You can issue this command twice as alternate for `-h` option.

```
PrimaryClusterB>snapmirror abort DFM/DFMQtree SecondaryClusterB:/DFM/DFMQtree

PrimaryClusterB>snapmirror abort DFM/DFMQtree SecondaryClusterB:/DFM/DFMQtree
```

4.8 SNAPMIRROR BREAK/RESYNC/RELEASE OPERATIONS

This section explains how to perform failover and failback operations between PrimaryCluster A and B and SecondaryCluster A and B in case of a planned or unplanned event.

BREAK OPERATION

The `snapmirror break` operation can be performed on both synchronous and asynchronous replication modes. When this command is executed, SnapMirror performs a failover to secondary site and changes the permission on all the SnapMirror volumes or qtrees on the SecondaryCluster A and B to writable from read-only permission.

SnapMirror synchronous or asynchronous replication does not copy LUN mapping and igroups information to SecondaryCluster A and B. Create igroups and map LUNs to particular igroups to enable the application residing on the host to access it. Before the operation, if the host accesses all LUNs unders the volumes and qtrees in read-only mode, perform manual rescan on hosts to enable read/write permission.

Example:

Unplanned failover: Suppose the ORAERPAggr aggregate on PrimaryClusterA has a disk failure and is not usable and LUNs under `ORAERP` volumes created on ORAERPAggr aggregate are mounted to host PrimORAERPServer. This causes all LUNs to disappear from the host and the Oracle Database to go offline. To address this scenario, first, perform a manual break operation on SecondaryClusterA, which performs SnapMirror failover. Then, map all the LUNs to the required igroups on SecondaryClusterB that have host access and bring up the Oracle application on the SecORAERPServer host. This paper does not describe an application-specific recovery procedure.

```
SecondaryClusterA>snapmirror break OraERP
```

Planned failover: Suppose a hardware maintenance job is performed on PrimaryClusterB due to controller replacement activity. You can perform a planned failover of the SnapMirror volumes or qtree to SecondaryClusterB or you can perform a cluster failover to PrimaryClusterA. However, due to SnapMirror operations, the load on the CPU and memory load will increase, which might decrease performance. Before performing a planned failover, do as follows:

1) If there is an ongoing SnapMirror transfer for the `App` and `Webservice` volumes, and the `DFMQtree` qtree, wait for transfer to complete.
2) Stop the applications and unmount all the LUNs exposed to applications at PrimaryClusterB system.
3) Perform a manual SnapMirror update operation to replicate any recent changed data blocks to the `App` and `Webservice` volumes and DFMQtree qtree on SecondaryClusterB. This is only for asynchronous replication mode.
4) Perform SnapMirror break operation on SecondaryClusterB on volumes and qtrees.
5) Mount all the LUNs to required igroup and bring up the host SecFileServer and SecAppServer.
6) Check for any errors after failover to SecondaryClusterB by accessing the applications:

```
SecondaryClusterB>snapmirror update –S PrimaryClusterAWeb:Webservice

SecondaryClusterB:Webservice

SecondaryClusterB>snapmirror update –S PrimaryClusterADFM:/DFM/DFMQtree
SecondaryClusterB:/DFM/DFMQtree

SecondaryClusterB>snapmirror break Webservice

SecondaryClusterB>snapmirror break App

SecondaryClusterB>snapmirror break /DFM/DFMQtree
```

RESYNC OPERATION

Resync operation can be performed to restore or redefine a SnapMirror relationship that was broken with a **`snapmirror break`** command. The **`resync`** command can be performed on primary or secondary SnapMirror systems in synchronous or asynchronous mode. The **`resync`** operation creates new SnapMirror Snapshot copy to compares with latest common Snapshot copy in PrimaryCluster A and B or SecondaryCluster A and B to find changed data blocks and transfers to it to the volumes or qtree.

For all the examples below, the SnapMirror relationship between PrimaryClusterB and SecondaryClusterB systems is considered broken and SecondaryClusterB acts as the primary system serving data to applications. Suppose the rate of changed data after failover to SecondaryClusterB is 50%. Suppose there is one common SnapDrive created consistent Snapshot copy and SnapMirror Snapshot copy before and after break operations as shown in Table 5.

Table 5) Resync operation example.

SnapMirror Storage Volume/Qtree Name	SnapMirror Snapshot Copies	SnapDrive Consistent Snapshot Copies
PrimaryClusterB:App	SecondaryClusterB(0012334488)_App.2	App.1
SecondaryClusterB:App	SecondaryClusterB(0012334488)_App.2	App.1
PrimaryClusterB:Webservice	SecondaryClusterB(0012334488)_Webservice.2	Web.1
SecondaryClusterB:Webservice	SecondaryClusterB(0012334488)_Webservice.2	Web.1
PrimaryClusterB:/DFM/DFMQtree	SecondaryClusterB(0012334488)_DFM.2	DFMQtree.2
SecondaryClusterB:/DFM/DFMQtree	SecondaryClusterB(0012334488)_DFM.2	DFMQtree.2

For example, suppose the resync operation is performed on the original source PrimaryClusterB or at destination SecondaryClusterB. Suppose after the break operation on PrimaryClusterB SnapMirror the `App` and `Webservice` volumes, and the `DFMQtree` qtree have write permissions on all volumes and the application is writing to LUNs residing under these volumes or qtree. If you perform resync to original source system PrimaryClusterA, then the roles of source and destination are reversed; this is known as flip resync. As part of the SnapMirror failback operation, it is required to copy all data modified after the break operation at SecondaryClusterB to the `App` and `Webservice` volumes and from the `DFMQtree` qtree to PrimaryClusterB. If you do not want to copy the modified data from SecondaryClusterB, you can perform a resync at existing destination SecondaryClusterB, which will overwrite all the modified data from the original source PrimaryClusterB. It destroys all the modified data at SecondaryClusterB, which might cause data loss.

The following steps describe how to perform flip resync:

1) Before performing SnapMirror resync operation on PrimaryClusterB and SecondaryClusterB, for the `Webservice` volume capture **`s`**`napmirror status -l & vol` status. In the `vol status` output, you can observe that the `Webservice` volume in PrimaryClusterB is writable, and is read-only in SecondaryClusterB. The `snapmirror status` output shows that the PrimaryClusterB and SecondaryClusterB have a common SnapMirror Snapshot copy.

```
PrimaryClusterB> snapmirror status -l Webservice

Snapmirror is on.

Source:                 PrimaryClusterB: WebService

Destination:            SecondaryClusterB:Webservice

Status:                 Idle

Progress:               -
```

```
State:                  Source
Lag:                    00:09:50
Mirror Timestamp:       Wed Jun 25 09:54:13 IST 2008
Base Snapshot:          SecondaryClusterB(0012334488)_Webservice1
PrimaryClusterB> vol status Webservice
Volume State      Status            Options
Webservice online     raid4, flex       snapmirrored=on,
snapmirrored      fs_size_fixed=on
SecondaryClusterB> snapmirror status -l Webservice
Snapmirror is on.
Source:                 PrimaryClusterB: WebService
Destination:            SecondaryClusterB:Webservice
Status:                 Idle
Progress:               -
State:                  Source
Lag:                    00:09:50
Mirror Timestamp:       Wed Jun 25 09:54:13 IST 2008
Base Snapshot:          SecondaryClusterB(0012334488)_Webservice1
SecondaryClusterB> vol status Webservice
Volume State      Status            Options
Webservice online     raid4, flex       snapmirrored=on,
snapmirrored      fs_size_fixed=on
read-only
```

2) SnapMirror break operation is performed on SecondaryClusterB for the `Webservice` volume, which causes the SnapMirror failover and converts the `Webservice` volume as writable. This volume can be accessed by applications residing in host SecAppServer for access data due to SnapMirror failover. On PrimaryClusterB, by default, the `Webservice` volume has write permission.

```
SecondaryClusterB> vol status Webservice
Volume State          Status            Options
Webservice online     raid4, flex       snapmirrored=on,
snapmirrored          fs_size_fixed=on
```

3) Applications that access the LUNs under the `Webservice` volume on SecondaryClusterB system will have overwritten data blocks by 50% after the failover operation.

4) As part of SnapMirror failback to original source PrimaryClusterB system, the resync operation has to be performed to copy all changed data in the `Webservice` volume on SecondaryClusterB. After the resync operation is successful, the permission on the `Webservice` volume on original PrimaryClusterB is changed to read-only. In `snapmirror status` output, you can see two entries. The first entry shows reversing of source and destination SnapMirror relation and the change in the base Snapshot copy. This is because resync operation creates a new SnapMirror Snapshot copy on SecondaryClusterB and compares with last SnapMirror Snapshot copy, finds the changed data, and transfers it to PrimaryClusterB. The second entry as the original relationship as shown above.

```
PrimaryClusterB> snapmirror resync -S SecondaryB:Webservice Webservice
The resync base Snapshot will be: SecondaryClusterB(0012334488)_Webservice.1
Are you sure you want to resync the volume? Yes
```

```
Revert to resync base Snapshot was successful

PrimaryClusterB> snapmirror status -l Webservice
Snapmirror is on.
Source:                 SecondaryClusterB: WebService
Destination:            PrimaryClusterB:Webservice
Status:                 Idle
Progress:               -
State:                  Source
Lag:                    00:09:50
Base Snapshot: PrimaryClusterB(0012334499)_Webservice1
Source:                 PrimaryClusterB: WebService
Destination:            SecondaryClusterB:Webservice
Status:                 Idle
Progress:               -
State:                  Source
Lag:                    00:09:50
 Base Snapshot:  SecondaryClusterB(0012334488)_Webservice1
PrimaryClusterB> vol status Webservice
Volume State          Status            Options
Webservice online     raid4, flex       snapmirrored=on,
snapmirrored          fs_size_fixed=on
read-only
```

5) If you do not want to copy the changed data blocks from the `Webservice` volume in SecondaryClusterB to the `Webservice` volume in PrimaryClusterB, you can perform the resync operation directly on SecondaryClusterB. This directly copies data in the `Webservice` volume in PrimaryClusterB to the `Webservice` volume in SecondaryClusterB. There is no need to perform the `release` or `flip` resync operations as mentioned below.

```
SecondaryClusterB> snapmirror resync -S SecondaryClusterB:Webservice
Webservice
The resync base Snapshot will be: SecondaryClusterB(0012334488)_Webservice.1
Are you sure you want to resync the volume? Yes
Revert to resync base Snapshot was successful
```

6) After `resync` operation is successfully, the SnapMirror `Webservice` volume in PrimaryClusterB will be replicated with the data blocks that were changed at the SecondaryClusterB after the break operation as part of failover.

7) To get the original source PrimaryClusterB, which now acts as a destination due to `snapmirror resync` operation, the manual break operation has to be performed on `Webservice` volume in PrimaryClusterB, which changes read-only mode to writable mode for all applications to be accessed. Before executing the break operation, the application should be quiesced and shut down. The application can resume or be restarted when the `Webservice` volume in PrimaryClusterB has write permissions enabled. After the break operation performed on PrimaryClusterB, the resync operation should be performed on SecondaryClusterB to get back the original SnapMirror relationship.

```
PrimaryClusterB> snapmirror break Webservice
PrimaryClusterB> vol status Webservice
```

```
Volume State          Status            Options
Webservice online     raid4, flex       snapmirrored=on,
snapmirrored          fs_size_fixed=on
```

SecondaryClusterB>snapmirror resync –S PrimaryClusterBWebservice:Webservice –w SecondaryClusterB:Webservice

SecondaryClusterB> vol status Webservice

```
Volume State          Status            Options
Webservice online     raid4, flex       snapmirrored=on,
snapmirrored          fs_size_fixed=on
read-only
```

8) The resync operation on SecondaryClusterB results in a duplicate entry shown in SnapMirror status output. To avoid confusion, the `snapmirror release` command can be executed on the SecondaryCluster system as follows:

SecondaryClusterB> snapmirror status -l Webservice

```
Snapmirror is on.
Source:                 SecondaryClusterB: WebService
Destination:            PrimaryClusterB:Webservice
Status:                 Idle
Progress:               -
State:                  Source
Lag:                    00:09:50
Base Snapshot: PrimaryClusterB(0012334499)_Webservice1
Source:                 PrimaryClusterB: WebService
Destination:            SecondaryClusterB:Webservice
Status:                 Idle
Progress:               -
State:                  Source
Lag:                    00:09:50
Base Snapshot:  SecondaryClusterB(0012334488)_Webservice1
```

SecondaryClusterB>snapmirror release Webservice PrimaryClusterBWebservice:Webservice

SecondaryClusterB>snapmirror update –S PimaryClusterBWebservice:Webservice –w SecondaryClusterB:Webservice

SecondaryClusterB>snapmirror status –l Webservice

```
Source:                 PrimaryClusterB: WebService
Destination:            SecondaryClusterB:Webservice
Status:                 Idle
Progress:               -
State:                  Source
```

```
Lag:                    00:09:50

Base Snapshot:  SecondaryClusterB(0012334488)_Webservice1
```

9) The resync operation can be performed only if the `Webservice` volume in PrimaryClusterB and SecondaryClusterB have a common Snapshot copy. The resync operation finds the current common Snapshot copy between the source and destination systems and transfers the changed data blocks to the destination volume or qtree.

   For example: Suppose the `SecondaryClusterB(0012334488)_Webservice.1` SnapMirror Snapshot copy on the `Webservice` volume in PrimaryClusterB is deleted or corrupted. However, a common SnapDrive Snapshot copy `Web.1` on PrimaryClusterB and the `Webservice` volume on SecondaryClusterB exists. The resync operation uses the next most recent common Snapshot copy to find the delta change. When performing resync on the `Web.1` common Snapshot copy, a message appears warning that the data written between Web.1 and latest SnapMirror `SecondaryClusterB(0012334488)_Webservice.1` Snapshot copy might be lost.

10) Resync operation can be a planned event after SnapMirror failover to SecondaryClusterB site. You can determine the number of changed data blocks in the SecondaryClusterB volumes and control the network bandwidth to transfer the changed data blocks to PrimaryClusterB.

    For example: Suppose the LUNs under the `Webservice` volume are changed by 100% and LUNs under the `App` volume are changed by 20% compared to the data residing in `Webservice` and `App` volumes in PrimaryClusterB. To manually determine the amount of data changed, create a Snapshot copy at the `App` and the `Webservice` volumes on SecondaryClusterB, and perform a `snap delta change` operation on the SnapMirror Snapshot copy and newly created Snapshot copy. After determining the amount of data changed in percentage, manually delete the Snapshot copy.

    By this method, the network bandwidth can be controlled to first transfer critical data.

```
SecondaryClusterB>snap create –V Webservice tempWeb

SecondaryClusterB>snap create –V App tempApp

SecondaryClusterB>snap delta –V Webservice
SecondaryCluster(0012334488)_Webservice.1 tempdb

SecondaryClusterB>snap delta –V Webservice
SecondaryCluster(0012334488)_App.1 App

SecondaryClusterB>snap delete –V Webservice tempdb

SecondaryClusterB>snap delete –V Webservice tempApp
```

4.9 SNAPMIRROR FAILOVER AND FAILBACK OPERATIONS

This section describes how to perform the SnapMirror failover and failback operations. Refer to Figure 1, Figure 2, and Table 1 through 4 for logical and physical SnapMirror connectivity.

FAILOVER OPERATION

When PrimaryClusterB is inaccessible, the following steps need to be executed to access the LUNs on the SecondaryClusterB. The `App` volume contains the App1, App2, and App3 LUNs. The `Webservice` volume contains Dept1, Dept2, and Dept3 LUNs.

Table 6) Failover procedure.

Step	Location	Command
1	SecondaryClusterB	Create an igroup to map LUNs by entering: **`igroup create –f –t windows AppWeb 10:00:00:00:c9:30:70:72`**
2	SecAppServer	Bring application to consistent state and flush host file systems/volume. If the application is running, before splitting the mirror, the application data has to be in a consistent state. The most common example would be putting a database in backup mode and synchronizing the file systems. In other cases, an application might need to be halted or momentarily suspended.
3	SecondaryClusterB	Stop the updating of the mirror from the primary to the secondary storage systems to prepare for splitting the mirror by entering: **`snapmirror quiesce`** SecondaryClusterB:App **`Snapmirror quiesce SecondaryClusterB:Webservice`**
4	SecondaryClusterB	Suspend or break the mirror by entering: **`snapmirror break SecondaryClusterB:App`** **`snapmirror break SecondaryClusterB:Webservice`** This does not remove the entire mirror relationship, which allows for resyncing of only the changed blocks when joining the mirror again.
5	Host	Resume application processing. If an application is halted, suspended, or put into backup mode, it can now return to a normal process status.
6	SecondaryClusterB	After the mirror is split, the LUNs are placed in an offline state. To put the LUN in the online state, enter: **`lun online /voll/App/App1`** **`/voll/App/App2`** **`/voll/App/App3`** **`/vol/Webservice/Dept1`** **`/vol/Webservice/Dept2`** **`vol/Webservice/Dept3`**
7	SecondaryClusterB	Map all the LUNs to the `AppWeb` igroup with the same LUN ID how it was mapped at the PrimaryClusterB system: **`lun map /vol/App/App1 AppWeb 0`** **`lun map /vol/App/App2 AppWeb 1`** **`lun map /vol/App/App3 AppWeb 2`** **`lun map /vol/App/Dept1 AppWeb 3`** **`lun map /vol/App/Dept2 AppWeb 4`** **`lun map /vol/App/Dept3 AppWeb 5`**

FAILBACK OPERATION

Table 7 describes the procedure to copy the data back to original site and reestablish the original configuration. It is assumed that the SnapMirror relationship from the PrimaryClusterB to the SecondaryClusterB is already split and the SecondaryClusterB is accessed. It is also assumed that PrimaryClusterB is intact or has been replaced and reconfigured to imitate the original storage system. If the storage system has to be replaced, in addition to the configuring IP addresses, volumes, and so on, the igroups must also be recreated, since these are not copied back with the data in the volume.

Table 7) Failback procedure.

Step	Command	Location	Description
1	**snapmirror resync -S SecondaryClusterB:App -w PrimaryClusterBApp:App** **snapmirror initialize -S SecondaryCluster -w PrimaryClusterBApp** **snapmirror resync -S SecondaryClusterB:Webservice -w PrimaryClusterBWebservice:Webservice** **snapmirror initialize -S SecondaryCluster -w PrimaryClusterBApp**	SecondaryClusterB	If you do not want to copy modified data on SecondaryClusterB (that is currently being accessed) to the PrimaryClusterBApp and PrimaryClusterWebservice, run resync on SecondaryClusterB instead of PrimaryClusterB storage. By performing resync operation on SecondaryClusterB there is no need to perform any release operations.
2	**snapmirror resync -S SecondaryClusterB:App -w PrimaryClusterBApp:App** **snapmirror initialize -S SecondaryCluster -w PrimaryClusterBApp** **snapmirror resync -S SecondaryClusterB:Webservice -w PrimaryClusterBWebservice:Webservice** **snapmirror initialize -S SecondaryCluster -w PrimaryClusterBApp**	PrimaryClusterB	Copies the data from the SecondaryClusterB (which is currently being accessed) to the PrimaryClusterBApp and PrimaryClusterWebservice. Only one of the commands should be executed, either with "`resync`" if the data is intact on PrimaryClusterBApp:App and PrimaryClusterBWebservice or with "`initialize`" if the data is no longer accessible. The "`-w`" option causes the command to wait to return until the transfer is completed.
3	**snapmirror update -S SecondaryClusterB:App -w PrimaryClusterBApp:App** **snapmirror update -S SecondaryClusterB:Webservice -w PrimaryClusterBWebservice:Webservice**	PrimaryClusterB	Copies changes that are been made since the copy in step 1 was started. Doing the extra update in this step minimizes the time the application needs to be offline for switching back to the primary site.
4	**Stop access to the SecondaryClusterB storage**	SecAppServer Host	Applications or users accessing the secondary storage system should be stopped.
5	**snapmirror update -S SecondaryClusterB:App -w PrimaryClusterBApp:App** **snapmirror update -S SecondaryClusterB:Webs**	PrimaryClusterB	Copies any changes made since the last update.

Step	Command	Location	Description
	ervice -w PrimaryClusterBWebservice:Webservice		
6	**snapmirror break PrimaryClusterBApp:App** **snapmirror break PrimaryClusterBWebservice:Webservice**	PrimaryClusterB	Splits the mirrors between PrimaryClusterB and SecondaryClusterB.
7	**lun map /vol/App/App1 AppWeb 0** **lun map /vol/App/App2 AppWeb 1** **lun map /vol/App/App3 AppWeb 2** **lun map /vol/App/Dept1 AppWeb 3** **lun map /vol/App/Dept2 AppWeb 4** **lun map /vol/App/Dept3 AppWeb 5**	PrimaryClusterB	Maps the LUNs to the appropriate igroup. This step is not required if the LUNs are already mapped to an igroup with the same name on the destination storage system.
8	**start application**	Host(s)	The users or application can now begin accessing the data from the original storage systems again.
9	**snapmirror resync -S PrimaryClusterBApp:App -w SecondaryClusterB:App** **snapmirror resync -S PrimaryClusterBWebservice:Webservice -w SecondaryClusterB:Webservice**	SecondaryClusterB	Restores the mirror to its original configuration.
10	**snapmirror release App PrimaryClusterB:App:** **snapmirror release Webservice PrimaryClusterB:Webservice**	SecondaryClusterB	Removes the relationship connecting the secondary storage system to the primary storage system, as this was only used to transfer the data back to the primary storage system. Having this relationship makes the SnapMirror status command output somewhat confusing.
11	**snapmirror update -S PimaryClusterBWebservice:Webservice -w SecondaryClusterB:Webservice** **snapmirror update -S PrimaryClusterBApp:App -w SecondaryClusterB:App**	SecondaryClusteraB	Deletes the `Primary(0012334499)_Webservice1` and `Primary(0012334499)_App1` base Snapshot copies which were part of flip resync, and you do not see any duplicate entries in SnapMirror status.

12	**snap list -V App**
	
	`  %/used  %/total date  name`
	`---------- ---------- ------------ --------`
	`  0% ( 0%)  0% ( 0%) Jul 14 13:55 SecondaryClusterB(0118056828)_App.3 (snapmirror)`
	
	`  9% ( 8%)  0% ( 0%) Jul 14 10:45 PrimaryClusterB(0118056873) App.3`
		PrimaryClusterB	The `PrimaryClusterB(0118056873)_App.3`, `PrimaryClusterB(0118056873)_App.2`, `PrimaryClusterB(0118056873)_Webservice.3`, and `PrimaryClusterB(0118056873)_Webservice.2` Snapshot copies are left as is. Their time stamp is three hours older than the current `SecondaryClusterB(0118056828)_App.3 (SnapMirror)` and `SecondaryClusterB(0118056828)_We`

Step	Command	Location	Description

	(snapmirror)
	
	  9% (0%) 0% (0%)
	Jul 14 10:44
	PrimaryClusterB(011805
	6873)_App.2
	
	Snap list -V
	Webservice
	
	%/used %/total
	date name
	---------- ----------
	---------- --------
	  0% (0%) 0% (0%)
	Jul 14 13:55
	SecondaryClusterB(0118
	056828)_WebService.3
	(snapmirror)
	
	  9% (8%) 0% (0%)
	Jul 14 10:45
	PrimaryClusterB(011805
	6873)_Webservice.3
	(snapmirror)
	
	  9% (0%) 0% (0%)
	Jul 14 10:44
	PrimaryClusterB(011805
	6873)_Webservice.2
	
	snap delete -V App
	PrimaryClusterB(011805
	6873)_App.3
	snap delete -V App
	PrimaryClusterB(011805
	6873)_App.2
	snap delete -V App
	PrimaryClusterB(011805
	6873)_Webservice.3
	snap delete -V App
	PrimaryClusterB(011805
	6873)_Webservice.2
			bservice.3 (SnapMirror) SnapMirror Snapshot copies, which can be deleted after you perform release operation. These SnapMirror Snapshot copies can be deleted to free up space.

5 USING SNAPMIRROR WITH SNAPDRIVE

This section explains how SnapDrive for Windows and UNIX can be integrated with SnapMirror operations. SnapDrive for UNIX cannot perform any SnapMirror operations; however, you can drive volume or LUN cloning operations in destination site for test, development, or backup purposes.

Before performing SnapMirror with SnapDrive, the following are required:

- SnapMirror must be licensed on the source and destination storage systems. For information on how license and set up SnapMirror, see the *Data ONTAP Data Protection Online Backup and Recovery Guide*.
- Depending on the LUN protocols you are using, enable the iSCSI and FCP licenses on the destination storage systems to enable LUN connect and LUN management operations.
- You must manually create and initialize a mirror between the source and destination volumes, but you must not create a SnapMirror replication schedule. When setting up SnapMirror on your storage system, you can avoid schedule conflicts with SnapDrive by setting the replication schedule on the storage system to "- - - -," which disables any scheduled transfers. When you set the replication schedule, make sure that the destination volume is in a restricted state. For more information, see the *Data ONTAP Data Protection Online Backup and Recovery Guide*.
- You must create your SnapMirror relationship using storage system names (either the fully qualified DNS name or the storage system name alone) and the network interface to be used for SnapMirror transfers (for example, storage1-e0), not IP addresses.
- The system must contain one or more SnapMirror source volumes hosting LUNs.
- The system must contain one or more SnapMirror destination volumes for each source volume.

  Note: SnapDrive supports the use of SnapMirror at the volume level only; it does not support qtree-level SnapMirror operations.
- The destination volume must be at least as large as the source volume.

  If you want to use a Windows host to access the replicated LUNs on the destination volume, the destination storage system must have at least one LUN access protocol licensed (iSCSI or FCP).

5.1 SNAPDRIVE FOR WINDOWS

This section explains how SnapDrive for Windows can be integrated to perform SnapMirror operations. SnapDrive for Windows cannot perform end-to-end SnapMirror management; however, it can perform SnapMirror failover operation to destination site if source LUNs are not accessible. It can control SnapMirror updates operations to manually replicate changed data blocks of LUNs on the source to destination that are integrated with applications to provide consistent data. NetApp recommends turning off SnapMirror schedule in `snapmirror.conf` file in destination storage for providing application restartable copy at destination site. To schedule the SnapMirror replication interval, see section 5.

Note: If your data on the source has not been accessed by any applications or if it is not modified during replication interval, then SnapMirror replication can be performed using scheduler. For example, in section 5, `Webservice` volume as SnapMirror schedule replicates static Web pages and HTML files created by the developer for development purpose.

Figure 3 shows a logical view of synchronous/asynchronous volumes with SnapMirror and SDW Snapshot copies, which will be referred to in the following examples. The same naming conventions are used for synchronous/asynchronous replication mode.

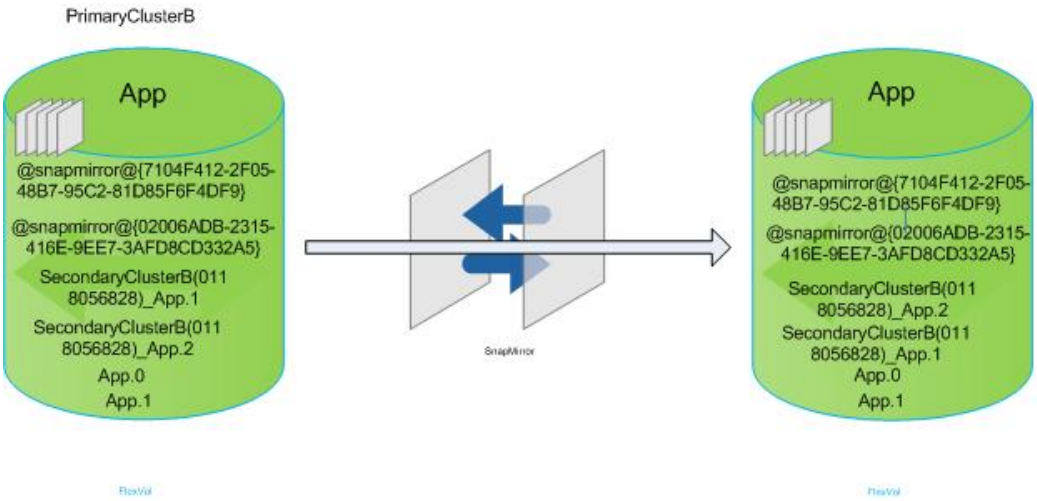


Figure 3) Logical view of asynchronous replication volume.

Table 8) SnapDrive for Windows.

Storage Systems	SDW Rolling Snapshot Copies	SnapMirror Snapshot Copies	SDW User-Created Snapshot copies
PrimaryClusterB	@snapmir@{7104F412-2F05-48B7-95C2-81D85F6F4DF9} @snapmir@{02006ADB-2315-416E-9EE7-3AFD8CD332A5}	SecondaryClusterB(0118056828)_App.2	App.0 App.1
SecondaryClusterB	@snapmir@{7104F412-2F05-48B7-95C2-81D85F6F4DF9} @snapmir@{02006ADB-2315-416E-9EE7-3AFD8CD332A5}	SecondaryClusterB(0118056828)_App.2 SecondaryClusterB(0118056828)_App.1	App.0 App.1

5.2 SNAPDRIVE ROLLING SNAPMIRROR SNAPSHOT COPIES

This section describes rolling SnapMirror Snapshot copies created in asynchronous replication mode. Figure 3 illustrates the SDW rolling Snapshot copies and SnapMirror Snapshot copies and SDW user-created Snapshot copies (see Table 8).

SnapDrive for Windows allows creation of rolling Snapshot copies on the `App` volume in PrimaryClusterB and then performs SnapMirror update that replicates these rolling Snapshot copies to the `App` volume SecondaryClusterB in synchronous or asynchronous mode.

If `snapmirror update` operation is performed using SnapDrive for Windows, it first checks if any SnapMirror rolling Snapshot copy exists in the `App` volume on PrimaryClusterB. If none exists, it creates a new rolling Snapshot copy named `@snapmir@{7104F412-2F05-48B7-95C2-81D85F6F4DF9}`. For the next SDW mirror update command it creates another `@snapmir@{02006ADB-2315-416E-9EE7-3AFD8CD332A5}` rolling Snapshot copy. At any given time, SnapDrive for Windows maintains two SnapMirror rolling Snapshot copies in the `App` volumes in PrimaryClusterB and SecondaryClusterB. The next SDW SnapMirror update deletes the previous rolling Snapshot copy and creates a new rolling Snapshot copy. Using this feature, SnapDrive for Windows maintains two SnapMirror common golden Snapshot copies for redundancy, which can be used if the common `SecondaryClusterB (0118056828) _App.2` SnapMirror Snapshot copy is deleted.

When creating volume clone using SDW rolling Snapshot copy, make sure to split the volume clone before next SDW SnapMirror update. If not, the SnapMirror update operation will fail. For more information, see section 7.

Turn off SnapMirror scheduler in the `snapmirror.conf` file and drive the SnapMirror update operation through hosts using SnapDrive for Windows, which is integrated with application to provide restartable copy

at the destination SecondaryClusterB App volume. Below is a sample Perl program that is not officially supported by NetApp. Before executing this Perl script you need to quiesce application I/O and then run script using Windows scheduler or command line.

Script: Copy this line in a notepad and save it with .pl extension and you can run using Windows scheduler.

```
system("sdcli snap update_mirror -d h:");
```

Before using rolling Snapshot copies, correctly size your storage requirements, to avoid running out of space on PrimaryClusterB or SecondaryClusterB system (see section 7)

5.3 SNAPDRIVE FOR UNIX

This section explains how SnapDrive for UNIX can be integrated with SnapMirror operations to perform failover and failback operations. SnapDrive for UNIX does not provide any commands to control SnapMirror configurations or operations; however, it can perform storage management such as creating LUNs, Snapshot copies, and cloning. It cannot perform SnapMirror failover to destination as parts of LUN connect and can perform SnapMirror update by creating rolling Snapshot copies, which are file system consistent with what SnapDrive for Windows offered.

DRIVING SNAPMIRROR UPDATE

Turn off SnapMirror scheduler in the `snapmirror.conf` file and drive SnapMirror update operation through SnapDrive for UNIX on the host, which provides a mechanism to replicate data that is application consistent at destination system. This can be achieved using a sample script. Before you run the script, make sure application is quiesced, and after the script is completed you can resume application I/O. Thus script does not provide any logic to quiesce and resume application I/O.

Below is a sample Perl program that is not supported by NetApp. Before executing this Perl script, you need to quiesce application I/O and then run script using corn or shell and then resume I/O.

Script: Copy this line in a notepad and save it with .pl extension and you can run using cron.

```
system("rsh snapmirror update App");
```

CLONING OPERATIONS ON DESTINATION

This section explains how to perform volume and LUN cloning in destination systems using SnapDrive for UNIX residing on UNIX hosts. Using the `snap mount` command, you can connect to any Snapshot copies residing on the SnapMirror `App` volume in PrimaryClusterB or SecondaryClusterB systems for read/write access. Connecting to the SnapMirror volumes in using Snapshot copies has different obligation and is explained in section 7. This section explains how to create volume clones on synchronous/asynchronous volumes or LUN clones on Qtree SnapMirror (QSM).

Volume clones can be created SnapDrive for UNIX using CLI on the host rather than from the storage. Before performing these operations make sure you have network connectivity to PrimaryClusterB and SecondaryClusterB storage systems. The command below creates a volume clone on SecondaryClusterB destination storage using latest `SecondaryClusterB(0118056828)_App.3` SnapMirror Snapshot copy for `App` volume and mount using `/Appclone` on host. This command can be performed on synchronous or asynchronous replication modes.

```
snapdrive snap connect –fs /Appclone –snapname
SecondaryClusterB:/vol/App:SecondaryClusterB(0118056828)_App.3
```

Note: You need to define san-clone-method to unrestrict the `snapdrive.conf` file.

SnapDrive for Windows can also delete volume clone after user has finished testing or backup operations. In the below command, the volume clone LUN mounted to the drive is deleted in the `App` volume in SecondaryClusterB.

```
snapdrive snap disconnect –fs /Appclone
```

LUN cloning can be performed on LUNs residing on volumes that are replicated using Qtree SnapMirror (QSM) mode. The following command is used for creating LUN cloning operations:

```
snapdrive snap connect –fs /Appclone –snapname
SecondaryClusterB:/vol/App:SecondaryClusterB(0118056828)_App.3
```

Note: You need to define san-clone-method to default in the `snapdrive.conf` file.

FAILOVER AND FAILBACK OPERATIONS

This section explains how to perform failover and failback using SDU in synchronous and asynchronous replication mode. SDU connect operation cannot break the mirror, and also you do not have rolling Snapshot copies, which are part of SnapDrive for Windows, to replicate consistent Snapshot copies during SnapMirror update.

1. Failover and failback in synchronous replication mode.

Table 9) Failover and failback in synchronous replication mode.

Step	Command
1	On PrimHost: 1. Stop application access to LUNs and shut down the server or quiesce application to hold I/O until you perform failover: **snapdrive storage disconnect -fs PrimaryClusterB:/vol/App/App1** 2. Unmap LUNs mapped to the PrimApp igroup that have access to PrimHost server on PrimaryClusterB storage. This makes sure that application will not perform write in PrimaryClusterB leading to duplicate data: **PrimaryClusterB>lun unmap /vol/App/App1 PrimHost**
2	Perform SnapMirror quiesce operation in the App SnapMirror volume in SecondaryClusterB on so that the replicated data getting transferred according to the SnapMirror schedule or SnapDrive for Windows SnapMirror update is applied to the App SnapMirror volume in SecondaryClusterB and stops further SnapMirror updates: **SecondaryClusterB>snapmirror quiesce App** On SecondaryClusterB: 1. Perform a SnapMirror break operation in the App SnapMirror volume in SecondaryClusterB which will perform failover by changing read-only to read/write access on App volume: **SecondaryClusterB>snapmirror break -S PrimaryClusterB:App** 2. On SecondaryClusterB storage, using CLI, map the LUN App1 under the App volume to the SecdApp igroup: **SecondaryClusterB>lun map /vol/App/App1 SecdApp** 3. Start the application. On SecdHost: 1. SDU cannot perform SnapMirror break operation on the App volume in SecondaryClusterB as part of disk connect operation: **snapdrive storage connect -fs /App PrimaryClusterB:/vol/App/App1** **2.** Start the application.
3	On PrimaryClusterB: SnapMirror resync operation performed on the App volume in PrimaryClusterB, replicates all changed data blocks to the App volume in SecondaryClusterB, which was part of failover back to the App volume in PrimaryClusterB. Doing the extra update in this step minimizes the time the application needs to be offline for switching back to the primary site. **PrimaryClusterB>snapmirror resync -S SecondaryClusterB:App App** **PrimaryClusterB>lun map /vol/App/App1 PrimApp** **PrimaryClusterB>snapmirror update -S SecondaryClusterB:App -w PrimaryClusterB:App** **PrimaryClusterB>snapmirror break App** On PrimHost: 1. After the resync and updates SnapMirror operations performed on PrimaryClusterB system connect to App1 LUN residing in the App volume in PrimaryClusterB: **snapdrive storage connect -fs /App PrimaryClusterB:/vol/App/App1**

Step	Command
	2. Start the application. On SecdHost: Stop application access to App1 LUN: **snapdrive storage disconnect -fs PrimaryClusterB:/vol/App/App1** On PrimHost: Start the application.
4	On SecondaryClusterB: After the application has started in PrimHost, perform a `flip resync` in SecondaryClusterB so that it reverses the replication relationship. Then, unmap App1 LUN in the `App` volume in SecondaryClusterB. Perform a few SnapMirror updates to replicate the changes performed on the `App` volume in PrimaryClusterB before synchronous SnapMirror performs CP and NVLOG forwarding. **SecondaryClusterB>snapmirror resync -S PrimaryClusterB:App App** **SecondaryClusterB>lun unmap /vol/App/App1 SecdApp** **SecondaryClusterB>snapmirror update -S PrimaryClusterB:App App**

5	On SecondaryClusterB and PrimaryClusterB:
	After the synchronous SnapMirror performs CP and NVLOG forwarding to destination SecondaryClusterB system it automatically reverts to the asynchronous mode during failover:
	snapmirror status App
	`Source Destination State Lag Status`
	`PrimaryClusterB:App SecondaryClusterB:App Snapmirrored - In-sync`

6	On SecondaryClusterB: Remove the relationship from the secondary storage system to the primary storage system, as this was only used to transfer the data back to the primary storage system. This relationship makes the SnapMirror status command output somewhat confusing: **snapmirror release App PrimaryClusterB:App**

7	On PrimaryClusterB:
	snap list -V App
	`%/used %/total date name`
	`---------- ---------- ------------ --------`
	`0% ( 0%) 0% ( 0%) Jul 14 13:55 SecondaryClusterB(0118056828)_App.3 (snapmirror)`
	`9% ( 8%) 0% ( 0%) Jul 14 13:45 PrimaryClusterB(0118056873)_App.3 (snapmirror) 9% ( 0%) 0% ( 0%) Jul 14 13:44 PrimaryClusterB(0118056873)_App.2`
	The `PrimaryClusterB(0118056873)_App.3, PrimaryClusterB(0118056873)_App.2, PrimaryClusterB(0118056873)_Webservice.3`, and `PrimaryClusterB(0118056873)_Webservice.2` Snapshot copies are left as is.
	Their time stamp is three hours older than the current `SecondaryClusterB(0118056828)_App.3 (SnapMirror)` and `SecondaryClusterB(0118056828)_Webservice.3 (SnapMirror)` SnapMirror Snapshot copies, which can be deleted after you perform release operation. These SnapMirror Snapshot copies can be deleted to free up space:
	snap delete -V App PrimaryClusterB(0118056873)_App.3
	snap delete -V App PrimaryClusterB(0118056873)_App.2

2. If there is a disaster event on the PrimaryClusterB App volume in asynchronous mode, to perform failover and failback using SnapDrive for Windows, do as follows:

Step	Command
1	On PrimHost: 1. Stop application access to LUNs and shut down the server or quiesce application to hold I/O until you perform failover: **snapdrive storage disconnect –fs PrimaryClusterB:/vol/App/App1** 2. Unmap LUNs mapped to igroup PrimApp that have access to PrimHost server on PrimaryClusterB storage. This makes sure that application will not perform write in PrimaryClusterB leading to duplicate data: **PrimaryClusterB>lun unmap /vol/App/App1 PrimHost**
2	On SecondaryClusterB: 1. Perform SnapMirror quiesce operation in SecondaryClusterB on the App SnapMirror volume so that the replicated data getting transferred according to the SnapMirror schedule or SnapDrive for Windows SnapMirror update is applied to SecondaryClusterB App volume and stops further SnapMirror updates: **SecondaryClusterB>snapmirror quiesce App** 2. Perform a SnapMirror break operation on SecondaryClusterB App volume: **SecondaryClusterB>snapmirror break –S PrimaryClusterB:App** 3. On SecondaryClusterB storage using CLI map the LUN App1 under App volume to SecdApp igroup. **SecondaryClusterB>lun map /vol/App/App1 SecdApp** On SecdHost: 1. SDU cannot perform SnapMirror break operation on SecondaryClusterB App volume as part of disk connect operation: **snapdrive storage connect –fs /App PrimaryClusterB:/vol/App/App** **2.** Start the application.
3	On PrimaryClusterB: SnapMirror resync operation performed on PrimaryClusterB on App volume replicates all changed data blocks on the SecondaryClusterB App volume, which was part of failover back to PrimaryClusterB App volume. Doing the extra update in this step minimizes the time the application needs to be offline for switching back to the primary site. **PrimaryClusterB>snapmirror resync –S SecondaryClusterB:App App** **PrimaryClusterB>lun map /vol/App/App1 PrimApp** **PrimaryClusterB>snapmirror update –S SecondaryClusterB:App –w PrimaryClusterB:App** **PrimaryClusterB>snapmirror break App** On PrimHost: 1. After the resync and update SnapMirror operations performed on PrimaryClusterB system connect to App1 LUN residing in PrimaryClusterB under the App volume: **snapdrive storage connect –fs /App PrimaryClusterB:/vol/App/App1** 2. Start the application. On SecdHost: Stop application access to App1 LUN: **snapdrive storage disconnect –fs PrimaryClusterB:/vol/App/App** On PrimHost: Start the application.

Step	Command
4	On SecondaryClusterB: After the application has started in PrimHost, perform a flip resync in SecondaryClusterB so that it reverses the replication relationship. Then, unmap App1 LUN in SecondaryClusterB under App volume. Perform a few SnapMirror updates to replicate the changes performed on the `App` volume on PrimaryClusterB before synchronous SnapMirror performs CP and NVLOG forwarding. **`SecondaryClusterB>snapmirror resync –S PrimaryClusterB:App  App`** **`SecondaryClusterB>lun unmap /vol/App/App1 SecdApp`** **`SecondaryClusterB>snapmirror update –S PrimaryClusterB:App App`**
5	On SecondaryClusterB: Remove the relationship from the secondary storage system to the primary storage system, as this was only used to transfer the data back to the primary storage system. This relationship makes the `snapmirror status` command output somewhat confusing: **`snapmirror release App PrimaryClusterB:App`**
6	On PrimHost: After you perform SnapMirror release operation in SecondaryClusterB system which had a duplicate entry caused by flip resync performed on PrimaryClusterB system, results in `PrimaryClusterB (0118056873)_App.3` and `PrimaryClusterB(0118056873)_App.2` SnapMirror Snapshot copies on the `App` volume in PrimaryClusterB and SecondaryClusterB. To delete these duplicate Snapshot copies to free up space, enter: **`rsh snapmirror update App`**

6	On PrimaryClusterB:
	`snap list –V App`
	`%/used  %/total  date  name`
	`----------  ----------  ------------  --------`
	`0% ( 0%)  0% ( 0%)  Jul 14 13:55 SecondaryClusterB(0118056828)_App.3 (snapmirror)`
	`9% ( 8%)  0% ( 0%)  Jul 14 13:45  PrimaryClusterB(0118056873)_App.3 (snapmirror)`
	`9% ( 0%)  0% ( 0%)  Jul 14 13:44  PrimaryClusterB(0118056873)_App.2`
	The `PrimaryClusterB(0118056873)_App.3`, `PrimaryClusterB(0118056873)_App.2`, `PrimaryClusterB(0118056873)_Webservice.3`, and `PrimaryClusterB(0118056873)_Webservice.2` Snapshot copies are left as is.
	Their time stamp is three hours older than the current `SecondaryClusterB(0118056828)_App.3 (SnapMirror)` and `SecondaryClusterB(0118056828)_Webservice.3 (SnapMirror)` SnapMirror Snapshot copies, which can be deleted after you perform release operation. These SnapMirror Snapshot copies can be deleted to free up space:
	`snap delete –V App PrimaryClusterB(0118056873)_App.3`
	`snap delete –V App PrimaryClusterB(0118056873)_App.2`

6 CLONING THE DESTINATION SNAPMIRROR SYSTEM

This section explains the obligations when creating FlexClone® flexible volumes on SnapMirror volumes for test and development purposes. It describes how SnapDrive can be leveraged to create FlexClone volumes on destination SnapMirror volumes.

6.1 VOLUME CLONING IN SYNCHRONOUS OR ASYNCHRONOUS MODE

LUNs in the SnapMirror volumes at the destination system can be cloned for various purposes depending on the customer's requirements. There are different behaviors observed when choosing Snapshot copies in the SnapMirror destination volumes for cloning volumes. Below examples list different results achieved after cloning in a SnapMirror environment. Figure 4 and Figure 5 provide a logical view of Snapshot copies created by SnapMirror and rolling Snapshot copies created by SnapDrive for Windows and user-created Snapshot copies on a Webservice SnapMirror volume.

Note: In the asynchronous replication mode, the latest SnapMirror Snapshot copy is available at the source site. In the synchronous replication mode, there are two SnapMirror Snapshot copies at the source and destination SnapMirror volume. However, after cloning, synchronous and asynchronous replication mode exhibit same behavior.

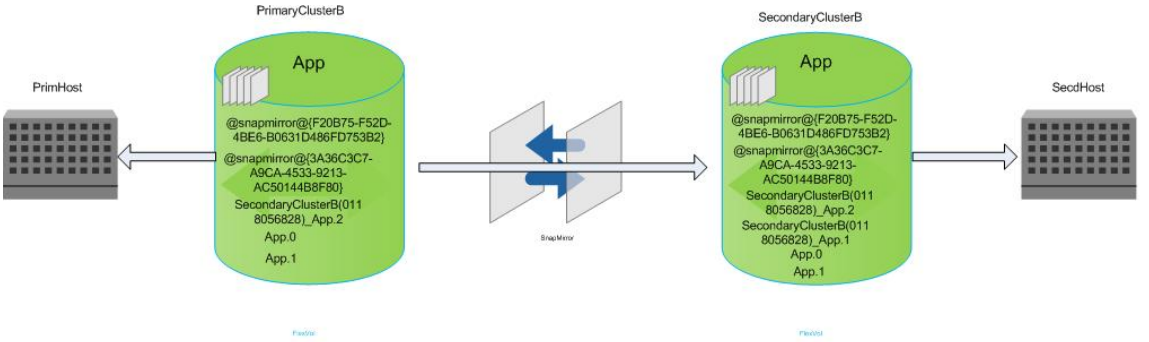


Figure 4) Asynchronous replication.

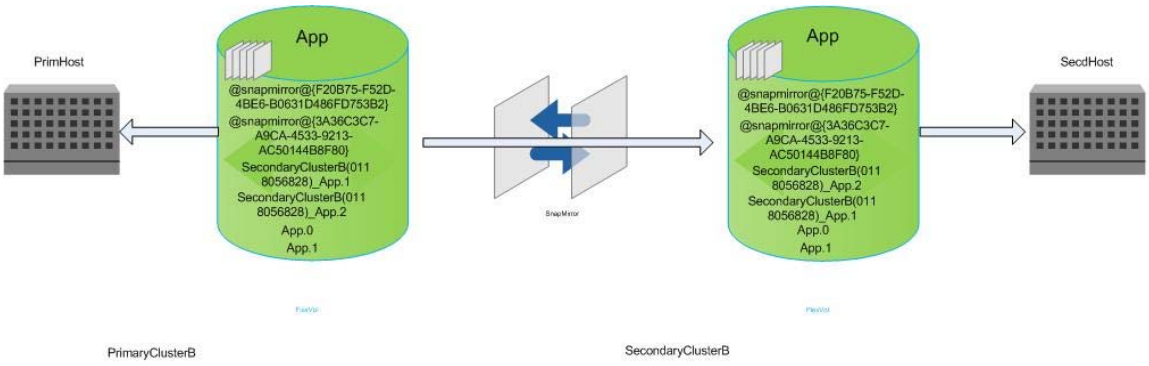


Figure 5) Synchronous replication.

SCENARIO 1: CLONING USING THE SNAPMIRROR SNAPSHOT COPY THAT IS DELETED IN THE SOURCE SYSTEM

The App volume on SecondaryClusterB is cloned using SecondaryClusterB(0118056828)_App.1 SnapMirror Snapshot copy. When the next SnapMirror update or schedule starts, because the Snapshot copy is used by clone volume and hence locked , SnapMirror replication fails displaying the (SecondaryClusterB: snapmirror.dst.snapDelErr:error): Snapshot SecondaryClusterB(0118056828)_App.1 error.

When the SnapMirror update operation starts, the SecondaryClusterB(0118056828)_App.1 Snapshot copy has to be deleted from the App volume in SecondaryClusterB as it is already deleted from the the App volume in PrimaryClusterB. The App volume in PrimaryClusterB only retains the current SnapMirror Snapshot copy and deletes the previous one as shown in the snap list output below. In synchronous/asynchronous replication mode in the App volume in PrimaryClusterB and SecondaryClusterB blocks have to be same. The SecondaryClusterB(0118056828)_App.1 Snapshot copy on SecondaryClusterB which is cloned can be changed as part of write operation, and data inconsistency can occur in SecondaryClusterB. To avoid this, the clone volume has to be split to release the lock on SecondaryClusterB(0118056828)_App.1 Snapshot copy, which will be deleted in next SnapMirror update.

```
SecondaryClusterB> vol clone create CloneApp -b App

SecondaryClusterB(0118056828)_App.1

SecondaryClusterB> snapmirror update -S PrimaryClusterB:App App

Tue Jul  1 18:40:25 IST (SecondaryClusterB: snapmirror.dst.snapDelErr:error):
Snapshot SecondaryClusterB(0118056828)_App.1 in destination volume App is in
use, cannot delete.Tue Jul  1 18:40:25 IST (SecondaryClusterB:
replication.dst.err:error): SnapMirror: destination transfer from
PrimaryClusterB:App to App : replication transfer failed to complete.

SecondaryClusterB> snap list -V App

%/used      %/total   date          name

----------  ----------  ------------  --------

0% ( 0%)    0% ( 0%)  Jul 01 14:00  SecondaryClusterB(0118056828)_App.2

0% ( 0%)    0% ( 0%)  Jul 01 13:41  SecondaryClusterB(0118056828)_App.1
(busy,snapmirror,vclone)

PrimaryClusterB> snap list -V App

%/used      %/total   date            name

----------  ----------  ------------  --------

0% ( 0%)    0% ( 0%)  Jul 01 18:34  SecondaryClusterB(0118056828)_App.3

0% ( 0%)    0% ( 0%)  Jul 01 14:00  SecondaryClusterB(0118056828)_App.2
(snapmirror)

SecondaryClusterB> vol clone split start CloneApp
```

SCENARIO 2: CLONING USING THE LATEST SNAPMIRROR SNAPSHOT COPY IN THE DESTINATION SYSTEM

Based on the schedule or when SnapMirror update operation is performed, the `App` volume is cloned in SecondaryClusterB using the `SecondaryClusterB(0118056828) _App.2` SnapMirror Snapshot copy, which is the latest SnapMirror Snapshot copy. The SnapMirror update operation does not delete the `SecondaryClusterB(0118056828) _App.2` Snapshot copy in PrimaryClusterB, as SnapMirror applies a soft lock to it. It creates a new Snapshot copy incremented by one that is, `SecondaryClusterB(0118056828)_App.3`, and continues replicatiing without errors. NetApp recommends always creating the FlexClone volume with the latest Snapshot copy, and splitting this volume if rate of change of data to the original App volume is more than 90%. After clone volume is split, SnapMirror removes the soft lock on the `SecondaryClusterB(0118056828)_App.2` Snapshot copy.

```
SecondaryClusterB> vol clone create CloneApp -b App

SecondaryClusterB(0118056828)_App.2

SecondaryClusterB> snapmirror update -S PrimaryClusterB:App App

SecondaryClusterB> snap list -V App
%/used       %/total  date          name
----------  ----------  ------------  --------
0% ( 0%)    0% ( 0%)  Jul 01 20:53  SecondaryClusterB(0118056828)_App.4
0% ( 0%)    0% ( 0%)  Jul 01 20:53  SecondaryClusterB(0118056828)_App.3
0% ( 0%)    0% ( 0%)  Jul 01 19:01  SecondaryClusterB(0118056828)_App.2
(busy,snapmirror,vclone)

PrimaryClusterB> snap list -V App
%/used       %/total  date          name
----------  ----------  ------------  --------
0% ( 0%)    0% ( 0%)  Jul 01 20:53  SecondaryClusterB(0118056828)_App.4
(snapmirror)
0% ( 0%)    0% ( 0%)  Jul 01 19:01  SecondaryClusterB(0118056828)_App.2
(snapmirror)
l

SecondaryClusterB> snap list -V App
%/used      %/total  date          name
----------  ----------  ------------  --------
0% ( 0%)    0% ( 0%)  Jul 01 20:53  SecondaryClusterB(0118056828)_App.4
0% ( 0%)    0% ( 0%)  Jul 01 20:53  SecondaryClusterB(0118056828)_App.3
0% ( 0%)    0% ( 0%)  Jul 01 19:01  SecondaryClusterB(0118056828)_App.2
```

SCENARIO 3: CLONING USING THE SNAPSHOT COPY CREATED BY SNAPDRIVE IN THE DESTINATION SYSTEM

For testing purposes, the App volume on SecondaryClusterB is cloned using the App.2 user created Snapshot copy.

SnapMirror applies a soft lock to the App.2 Snapshot copy when the manual snapmirror update command or the schedule defined in the snapmirror.conf file is used to replicate data. If the App.2 Snapshot copy is deleted from the App volume in PrimaryClusterB, the SnapMirror update operation fails. As a workaround, split the volume clone on SecondaryClusterB, or do not delete the App.2 Snapshot copy.

```
SecondaryClusterB> vol clone create cloneapp -b App App.2

SecondaryClusterB> snap list -V App
%/used      %/total  date          name
----------  ----------  ------------  --------
0% ( 0%)    0% ( 0%)  Jul 01 22:32  SecondaryClusterB(0118056828)_App.10
0% ( 0%)    0% ( 0%)  Jul 01 22:32  SecondaryClusterB(0118056828)_App.9
0% ( 0%)    0% ( 0%)  Jul 01 21:09  App.2          (busy,snapmirror,vclone)
0% ( 0%)    0% ( 0%)  Jun 30 09:01  App.1

PrimaryClusterB> snap delete -V App App.2

SecondaryClusterB> snapmirror update -S PrimaryClusterB:App App

(SecondaryClusterB: snapmirror.dst.snapDelErr:error): Snapshot App.2 in
destination volume App is in use, cannot delete. (SecondaryClusterB:
replication.dst.err:error): SnapMirror: destination transfer from
PrimaryClusterB:App to App : replication transfer failed to complete.

SecondaryClusterB> vol clone split start cloneapp

SecondaryClusterB> snap list -V App
%/used      %/total  date          name
----------  ----------  ------------  --------
0% ( 0%)    0% ( 0%)  Jul 01 22:42  SecondaryClusterB(0118056828)_App.11
0% ( 0%)    0% ( 0%)  Jul 01 22:32  SecondaryClusterB(0118056828)_App.1
0% ( 0%)    0% ( 0%)  Jun 30 09:01  App.1
```

SCENARIO 4: CLONING USING THE SDW ROLLING SNAPSHOT IN ASYNCHRONOUS REPLICATION MODE IN DESTINATION SYSTEM

This scenario is applicable in an asynchronous replication mode between the the App volume in PrimaryClusterB and the App volume in SecondaryClusterB. For testing purposes, the App volume on SecondaryClusterB is cloned using the SDW created @snapmir@{F20B751F-F52D-4BE6-B063-1D486FD753B2} rolling Snapshot copy. When SnapDrive for Windows performs a snapmirror update operation to replicate data, the @snapmir@{F20B751F-F52D-4BE6-B063-1D486FD753B2} rolling Snapshot copy is deleted from the App volume in PrimaryClusterB as there is no hard lock applied. When the next SnapMirror update operation is performed on the App volume in PrimaryClusterB, SnapMirror replication fails as the @snapmir@{F20B751F-F52D-4BE6-B063-1D486FD753B2} rolling Snapshot copy on PrimaryClusterA is deleted. This Snapshot copy is not deleted in SecondaryClusterA as FlexClone has applied a hard lock to it.

1. Suppose SnapDrive for Windows has created the following SnapMirror rolling Snapshot copy on the App volume in PrimaryClusterB and SecondaryClusterB:

   ```
   PrimaryClusterB> snap list -V App
   %/used       %/total  date          name
   ```

```
----------  ----------  ------------  --------
0% ( 0%)    0% ( 0%)  Jul 02 09:23  SecondaryClusterB(0118056828)_App.3
1% ( 1%)    0% ( 0%)  Jul 02 09:22  SecondaryClusterB(0118056828)_App.2

1% ( 0%)    0% ( 0%)  Jul 02 09:25  @ snapmir@{F20B751F-F52D-4BE6-
B0631D486FD753B2}
2% ( 1%)    0% ( 0%)  Jul 02 09:24  @snapmir@{3A36C3C7-A9CA-4533-9213-
AC50144B8F80}
```

0% (0%) 0% (0%) Jun 02 09:01 App.1

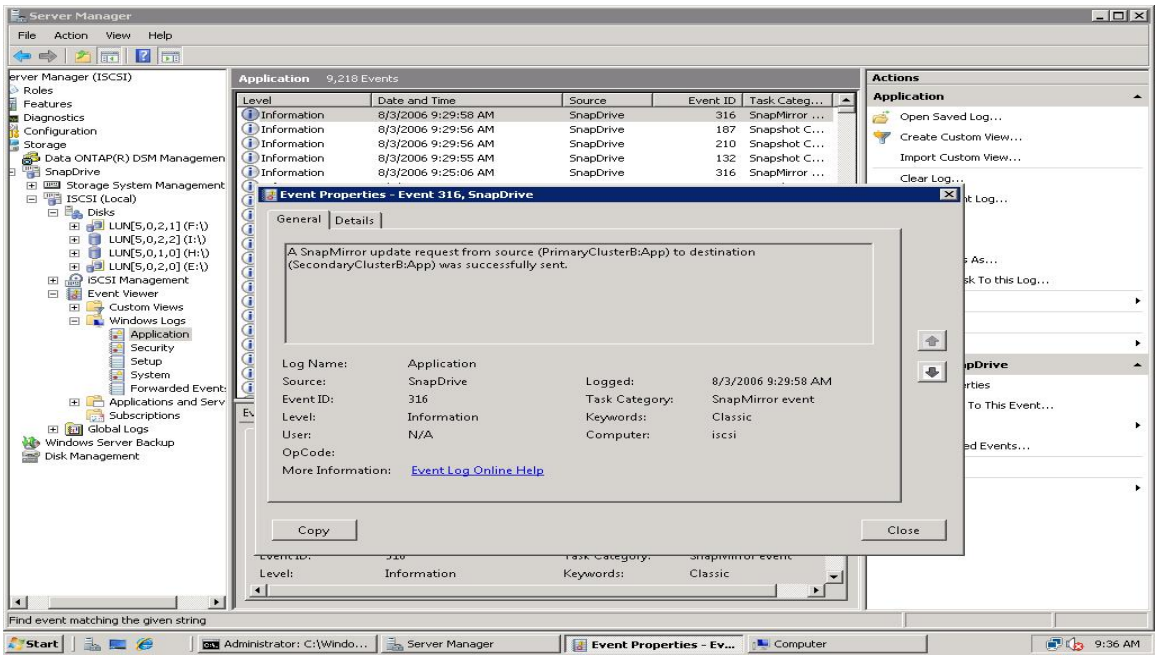
SecondaryClusterB> snap list -V App

```
%/used       %/total  date          name
----------  ----------  ------------  --------
0% ( 0%)    0% ( 0%)  Jul 02 09:23  SecondaryClusterB(0118056828)_App.3
1% ( 1%)    0% ( 0%)  Jul 02 09:22  SecondaryClusterB(0118056828)_App.2
1% ( 0%)    0% ( 0%)  Jul 02 09:25  @ snapmir@{F20B751F-F52D-4BE6-
B0631D486FD753B2}
2% ( 1%)    0% ( 0%)  Jul 02 09:24  @snapmir@{3A36C3C7-A9CA-4533-9213-
AC50144B8F80}
0% ( 0%)    0% ( 0%)  Jun 02 09:01  App.1
```

2. The SnapMirror update operation using SnapDrive for Windows deletes the `@snapmir@{F20B751F-F52D-4BE6-B063-1D486FD753B2}` rolling Snapshot copy from the App volume in PrimaryClusterB as it can only retain two rolling Snapshot copies at a time. It does not check for a hard lock on the `@snapmir@ {F20B751F-F52D-4BE6-B063-1D486FD753B2}` (Busy,snapmirror,vclone) rolling Snapshot copy in SecondaryClusterB. As part of SnapMirror update operation, the `@snapmir@{F20B751F-F52D-4BE6-B063-1D486FD753B2}` Snapshot copy in the `App` volume in SecondaryClusterB has to be deleted. However, it cannot be deleted as SnapMirror applies a hard lock. Due to this, the SnapMirror update operation fails as in the asynchronous replication mode, the source and destination should have identical data blocks. SnapDrive for Windows reports any error during SnapMirror update command and continues to create rolling Snapshot copies as shown in the below output:

C:\Users\Administrator>sdcli snap update_mirror -d e:

```
ISCSI : Checking policies
ISCSI : Preparing virtual disks for snapshot creation
ISCSI : Ready to create snapshot copy
ISCSI : Creating a snapshot for the virtual disk
ISCSI : Initiating snapmirror update for any source volumes.
The operation completed successfully
```




```
PrimaryClusterB> snap list -V App
%/used       %/total  date          name
---------  ----------  ------------  --------
1% ( 1%)    0% ( 0%)  Jul 02 09:25
SecondaryClusterB(0118056828)_App.4(snapmirror)
1% ( 0%)    0% ( 0%)  Jul 09 09:59  @snapmir@{274CE948-62C0-41F6-8F3D-
E43855FB84CD}
1% ( 1%)    0% ( 0%)  Jul 09 09:59  @snapmir@{92F295BE-A27F-4D24-BA64-
C25537DDFEB8}
0% ( 0%)    0% ( 0%)  Jun 02 09:01  App.1
SecondaryClusterB> snap list -V App
%/used       %/total  date          name
----------  ----------  ------------  --------
0% ( 0%)    0% ( 0%)  Jul 02 09:23  SecondaryClusterB(0118056828)_App.4
1% ( 1%)    0% ( 0%)  Jul 02 09:22  SecondaryClusterB(0118056828)_App.3
1% ( 0%)    0% ( 0%)  Jul 02 09:25  @snapmir@{F20B751F-F52D-4BE6-B063-
1D486FD753B2}
(Busy,snapmirror,vclone)
2% ( 1%)    0% ( 0%)  Jul 02 09:24  @snapmir@{3A36C3C7-A9CA-4533-9213-
AC50144B8F80}
0% ( 0%)    0% ( 0%)  Jun 02 09:01  App.1
```

3. The error message appears on the storage console or in telnet session if connected and is also recorded in the SnapMirror log file.

```
SecondaryClusterB> Wed Jul  2 09:31:22 IST (SecondaryClusterB:
snapmirror.dst.snapDelErr:error): Snapshot @snapmir@{F20B751F-F52D-4BE6-
B063-1D486FD753B2} in destination volume App is in use,Wed Jul  2 09:31:22
```

```
IST (SecondaryClusterB: replication.dst.err:error): SnapMirror: destination
transfer from PrimaryClusterB:App to App : replication transfer failed to
complete.
```

4. If you execute the SnapMirror update command from the telnet session, a message appears informing that the SnapMirror replication has failed.

```
SecondaryClusterB> snapmirror update -S PrimaryClusterB:App App

Transfer started. Wed Jul  2 09:31:22 IST (SecondaryClusterB:
snapmirror.dst.snapDelErr:error): Snapshot @snapmir@{F20B751F-F52D-4BE6-
B063-1D486FD753B2} in destination volume App is in use,Wed Jul  2 09:31:22
IST (SecondaryClusterB: replication.dst.err:error): SnapMirror: destination
transfer from PrimaryClusterB:App to App : replication transfer failed to
complete.
```

5. When the SnapMirror update operation using FilerView® or CLI on the SecondaryClusterB storage, is performed SnapMirror applies a soft lock and on the `@snapmir@{F20B751F-F52D-4BE6-B063-1D486FD753B2}` Snapshot copy to the `App` volume in PrimaryClusterB. The `snapmirror update` operation retains only two rolling Snapshot copies in the `App` volume in PrimaryClusterB.

```
PrimaryClusterB> snap list -V App

---------- ---------- ------------ --------

1% ( 1%)    0% ( 0%)  Jul 02 09:25
SecondaryClusterB(0118056828)_App.3(snapmirror)

1% ( 0%)    0% ( 0%)  Jul 02 09:25  @snapmir@{F20B751F-F52D-4BE6-B063-
1D486FD753B2}(snapmirror)

2% ( 1%)    0% ( 0%)  Jul 02 09:24  @snapmir@{3A36C3C7-A9CA-4533-9213-
AC50144B8F80}

0% ( 0%)    0% ( 0%)  Jun 02 09:01  App.1

SecondaryClusterB> snap list -V App

  %/used       %/total  date          name

----------  ----------  ------------  --------

0% ( 0%)    0% ( 0%)  Jul 02 09:23  SecondaryClusterB(0118056828)_App.3

1% ( 1%)    0% ( 0%)  Jul 02 09:22  SecondaryClusterB(0118056828)_App.2

1% ( 0%)    0% ( 0%)  Jul 02 09:25  @snapmir@{F20B751F-F52D-4BE6-B063-
1D486FD753B2}

(Busy,snapmirror,vclone)

2% ( 1%)    0% ( 0%)  Jul 02 09:24  @snapmir@{3A36C3C7-A9CA-4533-9213-
AC50144B8F80}

0% ( 0%)    0% ( 0%)  Jun 02 09:01  App.1
```

6. To prevent the `snapmirror update` operation from failing due the problem above, rename the `@snapmir@{F20B751F-F52D-4BE6-B063-1D486FD753B2}` SDW rolling Snapshot copy to `old.1` in the `App` volume in PrimaryClusterB before creating a volume clone clone. When the SnapDrive for Windows `snapmirror update` operation is performed, it does not delete the `old.1` Snapshot copy. Instead, it creates a new rolling Snapshot copy as it only retains two rolling Snapshot copies. After the `rename` operation, perform the SnapMirror update and then create the FlexClone volume.

```
C:\ snap rename -d e -o @snapmir@{F20B751F-F52D-4BE6-B063-1D486FD753B2}  -n
old.1

PrimaryClusterB> snap rename -V App @snapmir@{F20B751F-F52D-4BE6-B063-
1D486FD753B2}     old.1

PrimaryClusterB> snap list -V App
```

```
%/used       %/total  date          name
----------  ----------  ------------  --------
0% ( 0%)    0% ( 0%)  Jul 02 09:23  SecondaryClusterB(0118056828)_App.3
1% ( 1%)    0% ( 0%)  Jul 02 09:22  SecondaryClusterB(0118056828)_App.2
1% ( 0%)    0% ( 0%)  Jul 02 09:25  old.1
2% ( 1%)    0% ( 0%)  Jul 02 09:24  @snapmir@{3A36C3C7-A9CA-4533-9213-
AC50144B8F80}
0% ( 0%)    0% ( 0%)  Jun 02 09:01  App.1
```

PrimaryClusterB> snap list -V App

```
%/used       %/total  date          name
----------  ----------  ------------  --------
0% ( 0%)    0% ( 0%)  Jul 02 09:23  SecondaryClusterB(0118056828)_App.3
1% ( 1%)    0% ( 0%)  Jul 02 09:22  SecondaryClusterB(0118056828)_App.2
1% ( 0%)    0% ( 0%)  Jul 09 16:53  @snapmir@{31A58170-6B9F-4385-92D9-
F27AF4412CE0}
1% ( 1%)    0% ( 0%)  Jul 09 16:52  @snapmir@{5FEB3F7D-4118-451D-B07D-
09E9C6B3836A}           1% ( 1%)    0% ( 0%)  Jul 09 16:41  old.1
(snapmirror)
0% ( 0%)    0% ( 0%)  Jun 02 09:01  App.1
```

SecondaryClusterB> snap list -V App

```
%/used       %/total  date          name
----------  ----------  ------------  --------
0% ( 0%)    0% ( 0%)  Jul 02 09:23  SecondaryClusterB(0118056828)_App.3
1% ( 1%)    0% ( 0%)  Jul 02 09:22  SecondaryClusterB(0118056828)_App.2
1% ( 0%)    0% ( 0%)  Jul 09 16:53  @snapmir@{31A58170-6B9F-4385-92D9-
F27AF4412CE0}
1% ( 1%)    0% ( 0%)  Jul 09 16:52  @snapmir@{5FEB3F7D-4118-451D-B07D-
09E9C6B3836A}
1% ( 1%)    0% ( 0%)  Jul 09 16:41  old.1 (busy,snapmirror,vclone)
0% ( 0%)    0% ( 0%)  Jun 02 09:01  App.1
```

7. When the `cloneapp` volume clone is locked on the SecondaryClusterB system, SnapMirror locks the `@snapmir@{F20B751F-F52D-4BE6-B0631D486FD753B2}(busy,snapmirror,vclone)` Snapshot copy. Hence, failover to destination `App` volume in SecondaryClusterB restores the data present just before the SnapMirror replication failure. Hence, always split the `cloneapp` volume clone or rename the`@snapmir@{F20B751F-F52D-4BE6-B0631D486FD753B2` Snapshot copy on the `App` volume in PrimaryClusterB. Perform the SnapMirror updates using SnapDrive for Windows or using SnapMirror commands on telnet or console, and then perform a SnapMirror failover to `App` volume in SecondaryClusterB.

8. If failover is performed using SnapDrive for Windows in GUI or CLI mode in the `App` volume in SecondaryClusterB without splitting the clone, the following is observed:

a) SnapDrive for Windows restores the `App` volume using the latest `snapmir@{67B64AA6-CEFC-402B-8C2C-C8FF413D0C2C}` rolling Snapshot copy and deletes the `SecondaryClusterB(0118056828)_App.4` common SnapMirror Snapshot copy. Hence, SnapMirror resync operation cannot be performed. If there are no common Snapshot copies in the `App`

volume in PrimaryClusterB and SecondaryClusterB, perform a SnapMirror initialize operation to replicate the whole App volume.

- **Before Failover**

```
SecondaryClusterB> snap list -V App
%/used       %/total  date          name
----------  ----------  ------------  --------
0% ( 0%)    0% ( 0%)  Jul 08 11:54  SecondaryClusterB(0118056828)_App.4
0% ( 0%)    0% ( 0%)  Jul 08 11:54  @snapmir@{67B64AA6-CEFC-402B-8C2C-
C8FF413D0C2C}
1% ( 0%)    0% ( 0%)  Jul 02 09:25  @snapmir@{F20B751F-F52D-4BE6-B063-
1D486FD753B2}
(Busy,snapmirror,vclone)
```

- **After Failover**

  Using SnapDrive for Windows GUI, connect disk to the App volume in SecondaryClusterB.

```
PrimaryClusterB> snap list -V App
%/used       %/total  date          name
----------  ----------  ------------  --------
0% ( 0%)    0% ( 0%)  Jul 08 11:58  SecondaryClusterB(0118056828)_App.5
1% ( 0%)    0% ( 0%)  Jul 08 11:58  @snapmir@{17002CAE-773E-48C7-97DC-
618FDFD935E2}
1% ( 1%)    0% ( 0%)  Jul 08 11:56  @snapmir@{C70B7943-4AC6-433D-9708-
80606611ADC3}
2% ( 1%)    0% ( 0%)  Jul 08 11:54  SecondaryClusterB(0118056828)_App.4
(snapmirror)
3% ( 1%)    0% ( 0%)  Jul 07 15:23  app.1


SecondaryClusterB> snap list -V App
%/used       %/total  date          name
----------  ----------  ------------  --------
1% ( 1%)    0% ( 0%)  Jul 08 11:54  @snapmir@{67B64AA6-CEFC-402B-8C2C-
C8FF413D0C2C}
3% ( 1%)    0% ( 0%)  Jul 08 10:30  SecondaryClusterB(0118056828)_App.3
1% ( 0%)    0% ( 0%)  Jul 02 09:25  @snapmir@{F20B751F-F52D-4BE6-B063-
1D486FD753B2}
(Busy,snapmirror,vclone)
3% ( 1%)    0% ( 0%)  Jul 07 15:23  app.1
```

b) You can perform failover to the SecondaryClusterB using the SnapDrive for Windows command line where you can specify the Snapshot copy to perform the App volume restore. SnapDrive for Windows only works with its consistent Snapshot copies and not the SnapMirror or user created Snapshot copies.

c) In this above case SnapMirror break operation using SnapDrive for Windows GUI restores the App volume using the latest @snapmir@{67B64AA6-CEFC-402B-8C2C-C8FF413D0C2C} rolling Snapshot copy and deletes the SecondaryClusterB(0118056828)_App.5 as it is older than the rolling Snapshot copy. This results in deleting the common

`SecondaryClusterB(0118056828)_App.5` SnapMirror Snapshot copy from SecondaryClusterB App volume and requires fresh SnapMirror initialization.

9. As a best practice, always create the volume clone using the latest SnapMirror Snapshot copy. To create more than one volume clone, use the SnapDrive rolling Snapshot copies. In this case, rename the rolling Snapshot copies on source and then create a volume clone using the renamed Snapshot copy name or split the volume clone before the next SnapMirror update operation.

6.2 LUN CLONING IN QTREE SNAPMIRROR REPLICATION

This section describes how to perform LUN cloning on destination SnapMirror volumes that have Qtree SnapMirror replication mode. LUNs under `DFMQtree` residing on destination on the `DFM` volume in SecondaryClusterB can be cloned outside `DFMQtree` as it is a read-only destination qtree. SnapDrive for Windows or UNIX does not support QSM replication mode. However, you can create LUN clones. Figure 5 shows the logical view with Snapshot copies on DFM volumes. The `/vol/DFM/DFMQtree` qtree in PrimaryClusterB is replicated to `/vol/DFM/DFMQtree` in SecondaryClusterB. There are one QSM created Snapshot copy in the `DFM` volume in PrimaryClusterB and two hanging Snapshot copies in in the `DFM` volume in SecondaryClusterB.

Note: The `DFM` volume in SecondaryClusterB is writable; however, `DFMQtree` under the `DFM` volume is read-only.

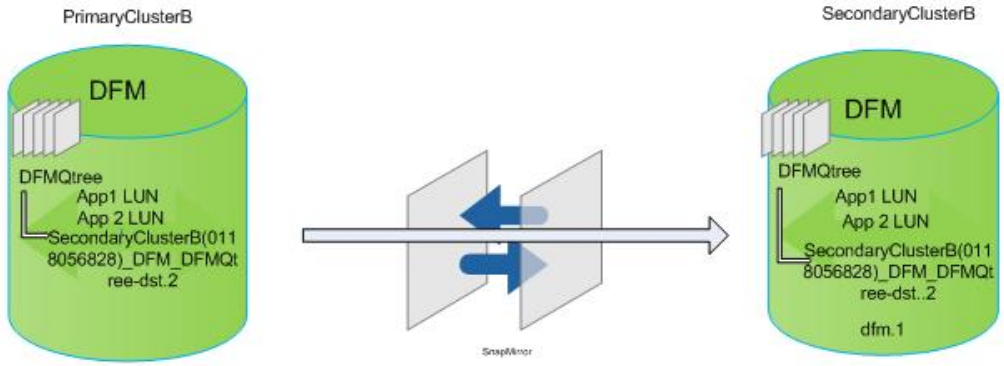


Figure 6) Logical view with Snapshot copies on DFM volumes.

SCENARIO 1

The App1 LUN under the `/vol/DFM/DFMQtree` qtree is cloned using the `DFM.1` Snapshot copy created in the `DFM` volume in SecondaryClusterB. You can use SnapDrive for Windows or UNIX to create consistent DFM.1 and Snapshot copies and use the `mount` command to create a LUN clone outside `/vol/DFM/DFMQtree` as follows:

1. Before creating a `DFM.1` Snapshot copy on the `DFM` volume in SecondaryClusterB, perform a SnapMirror quiesce operation on the `/vol/DFM/DFMQtree` qtree so that the ongoing replication transfer is forced to complete and any other scheduled or manual Snapshot technology is blocked.

```
SecondaryClusterB> snapmirror quiesce /vol/DFM/DFMQtree
snapmirror quiesce: in progress
```

2. Monitor `snapmirror status` to check if the quiesce is complete:

```
SecondaryClusterB> snapmirror status -l /vol/DFM/DFMQtree

Source:                 PrimaryClusterB:/vol/DFM/DFMQtree

Destination:            SecondaryClusterB:/vol/DFM/DFMQtree

Status:                 Idle

Progress:               -

State:                  Quiesced
```

```
Lag:                        03:42:27
```

3. After the quiesce operation is completed, create a Snapshot copy in the `DFM` volume in SecondaryClusterB.

   SecondaryClusterB> snap create -V DFM DFM.1

4. Perform `snapmirror resume` after creating the `DFM.1` Snapshot copy that allows subsequent SnapMirror update schedule.

   SecondaryClusterB> snapmirror resume /vol/DFM/DFMQtree

   ```
   snapmirror resume: /vol/DFM/DFMQtree : Successfully resumed
   ```

5. You can use SnapDrive for Windows or UNIX to create LUN clone using `mount` or `connect` commands.

6.3 VOLUME/LUN CLONING OPERATIONS

This section describes how to create volume clones on synchronous or asynchronous volumes or LUN clones on qtree SnapMirror.

Using the SDW `snap mount` command, you can connect to any Snapshot copies residing on the `App` volume on PrimaryClusterB or SecondaryClusterB systems for read/write access. Connecting to SecondaryClusterB SnapMirror volumes using Snapshot copies has different obligation and is explained in section 7.

- SnapDrive for Windows can perform volume cloning using the GUI or CLI. Before performing these operations check that you have network connectivity to PrimaryClusterB and SecondaryClusterB. The following command creates a volume clone on SecondaryClusterB destination storage using the latest `SecondaryClusterB(0118056828)_App.3` for `App` volume SnapMirror Snapshot copy and mounts it using the "t" drive on host. This command can be performed on synchronous or asynchronous replication modes.

  c:\sdcli snap mount -k e -s SecondaryClusterB(0118056828)_App.3 -d t -smdest -sdf SecondaryClusterB -sdv App

- SnapDrive for Windows can also delete the volume clone after the user has finished testing or backup operations. To delete the volume clone LUN mounted to "t" drive in the the `App` volume in SecondaryClusterB, enter:

  c:\sdcli snap unmount –d t

- LUN cloning can be performed on LUNs residing under volumes that are replicated using the qtree as follows:

  c:\sdcli snap mount -k e -s app.2 -d t -smdest -sdf SecondaryClusterB -sdv App

  c:\sdcli umount –d t

  c:\sdcli disk connect -p SecondaryClusterB:/vol/App/App1 -d k: -s SecondaryClusterB(0118056828)_App.4 -IG iscsi iscsi -dtype dedicated

FAILOVER AND FAILBACK SNAPMIRROR OPERATIONS USING SNAPDRIVE

This section explains how SnapDrive performs SnapMirror failover and failback operations during planned or unplanned events. Figure 7 shows logical view of synchronous and asynchronous replication modes.

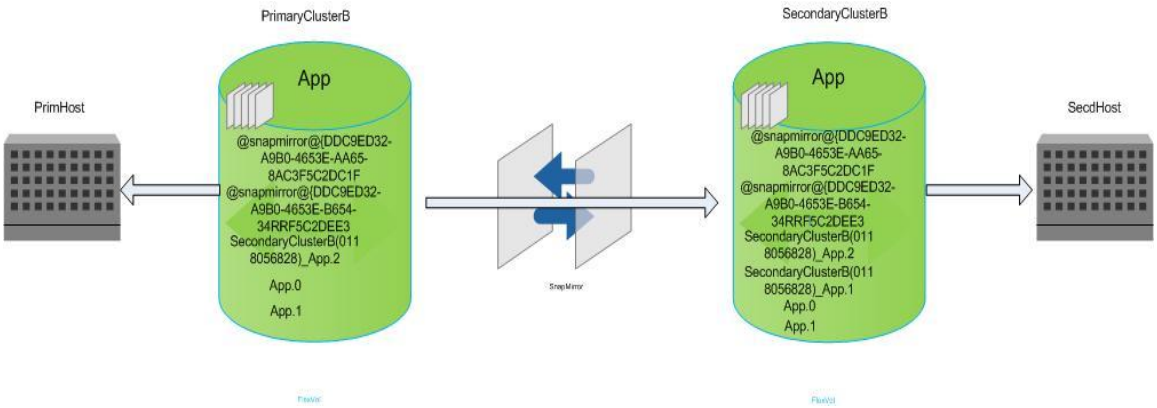


Figure 7) Logical view of synchronous and asynchronous replication modes.

1. Failover and failback operations in synchronous replication mode.

   When SnapDrive for Windows performs a LUN connect on the `App` volume in SecondaryClusterB, it performs a SnapMirror break and changes the `App` volume access from read-only to read/write access. When you connect any LUN residing on the `App` volume, SnapDrive for Windows will perform the SnapMirror break operation. As best practice, always group similar application LUNs under same volume to be part of SnapMirror failover during planned or unplanned event.

   Suppose PrimHost is accessing an application using LUNs created under the `App` volume in PrimaryClusterB and igroups are created under both PrimaryClusterB and SecondaryClusterB storage system. SecdHost is a standby server placed in destination site, which will be used only when disaster occurs.

   To perform failover and failback using SnapDrive for Windows during a disaster event at the source site PrimaryClusterB App volume, do as follows:

Step	Command
1	<u>On PrimHost and PrimaryClusterB:</u> 1. Stop application access to LUNs and shut down the server or quiesce application to hold I/O until you perform failover: **`c:\sdcli disk disconnect -p PrimaryClusterB:/vol/App/App1`** 2. Unmap LUNs mapped to the `PrimApp` igroup that have access to PrimHost server on PrimaryClusterB storage. This makes sure that application will not perform write in PrimaryClusterB leading to duplicate data: **`SecondaryClusterB>lun map /vol/App/App1 SecdApp`**
2	Perform SnapMirror quiesce operation on the `App` SnapMirror volume in SecondaryClusterB so that the replicated data getting transferred according to the SnapMirror schedule or SDW SnapMirror update is applied to the `App` volume in SecondaryClusterB, and stops further SnapMirror updates:p **`SecondaryClusterB>snapmirror quiesce App`** <u>On SecondaryClusterB:</u> 1. Perform a SnapMirror break operation on the `App` volume in SecondaryClusterB, which will perform failover by changing read-only to read/write access: **`SecondaryClusterB>snapmirror break -S PrimaryClusterB:App`** 2. On SecondaryClusterB storage using CLI map the App1 LUN under the `App` volume to the

Step	Command
	SecdApp igroup. **SecondaryClusterB>lun map /vol/App/App1 SecdApp** 3. Start the application. On SecdHost: 1. SDU cannot perform SnapMirror break operation on the App volume in SecondaryClusterB as part of disk connect operation: **c:\ sdcli disk connect -p SecondaryClusterB:/vol/App/App1 -d w -IG SecdHost SecdApp -dtype dedicated** **2.** Start the application.
3	On PrimaryClusterB: SnapMirror resync operation performed on the App volume in PrimaryClusterB replicates all changed data blocks on the App volume in SecondaryClusterB, which was part of failover back to the App volume in PrimaryClusterB. Doing the extra update in this step minimizes the time the application needs to be offline for switching back to the primary site. **PrimaryClusterB>snapmirror resync –S SecondaryClusterB:App App** **PrimaryClusterB>lun map /vol/App/App1 PrimApp** **PrimaryClusterB>snapmirror break App** **PrimaryClusterB>snapmirror update –S SecondaryClusterB:App –w PrimaryClusterB:App** On PrimHost: 1. After the resync and update SnapMirror operations performed on PrimaryClusterB system connect to the App1 LUN residing in the App volume in PrimaryClusterB. 2. Start the application. On SecdHost: Stop application access to App1 LUN: **snapdrive storage disconnect –fs PrimaryClusterB:/vol/App/App1** On PrimHost: Start the application: **C:\sdcli disk connect –p PrimaryClusterB:/vol/App/App1 –d w –IG PrimHost PrimApp -dtype dedicated**
4	On SecondaryClusterB: After the application has started in PrimHost, perform a flip resync in SecondaryClusterB so that it reverses the replication relationship. Then, unmap the App1 LUN in the App volume in SecondaryClusterB. Perform a few SnapMirror updates to replicate the changes performed on the App volume in PrimaryClusterB before synchronous SnapMirror performs CP and NVLOG forwarding. **SecondaryClusterB>snapmirror resync –S PrimaryClusterB:App App** **SecondaryClusterB>lun unmap /vol/App/App1 SecdApp** **SecondaryClusterB>snapmirror update –S PrimaryClusterB:App App**
5	On SecondaryClusterB and PrimaryClusterB: After the synchronous SnapMirror performs CP and NVLOG forwarding to destination SecondaryClusterB system it automatically reverts to the asynchronous mode during failover: **snapmirror status App** Source Destination State Lag Status PrimaryClusterB:App SecondaryClusterB:App Snapmirrored - In-sync

Step	Command
6	On SecondaryClusterB: Remove the relationship from the secondary storage system to the primary storage system, as this was only used to transfer the data back to the primary storage system. This relationship makes the SnapMirror status command output somewhat confusing: **snapmirror release App PrimaryClusterB:App**
7	On PrimaryClusterB: **snap list -V App** %/used %/total date name ---------- ---------- ------------ -------- 0% (0%) 0% (0%) Jul 14 13:55 SecondaryClusterB(0118056828)_App.3 (snapmirror) 9% (8%) 0% (0%) Jul 14 13:45 PrimaryClusterB(0118056873)_App.3 (snapmirror) 9% (0%) 0% (0%) Jul 14 13:44 PrimaryClusterB(0118056873)_App.2 The PrimaryClusterB(0118056873)_App.3, PrimaryClusterB(0118056873)_App.2, PrimaryClusterB(0118056873)_Webservice.3, and PrimaryClusterB(0118056873)_Webservice.2 Snapshot copies are left as is. Their time stamp is three hours older than the current SecondaryClusterB(0118056828)_App.3 (SnapMirror) and SecondaryClusterB(0118056828)_Webservice.3 (SnapMirror) Snapshot copies, which can be deleted after you perform release operation. These SnapMirror Snapshot copies can be deleted to free up space: **snap delete -V App PrimaryClusterB(0118056873)_App.3** **snap delete -V App PrimaryClusterB(0118056873)_App.2**

2. Failover and failback operations in asynchronous mode replication.

   Figure 8 displays the App SnapMirror volume in PrimaryClusterB and SecondaryClusterB systems. The SDW SnapMirror update operation transfers the changed data blocks on the App1 LUN residing in the App volume in PrimaryClusterB to the App volume in SecondaryClusterB in form of rolling Snapshot copies. This in turn performs SnapMirror update operations and results in two rolling Snapshot copies and Snapshot copies based on SnapMirror.

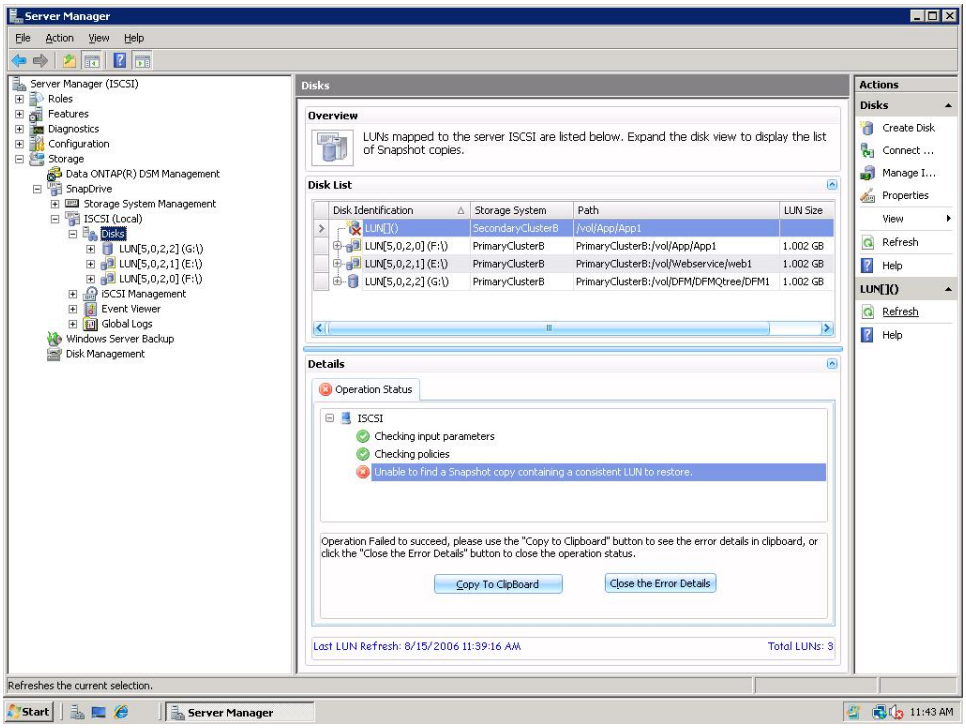


Figure 8) SnapMirror volume.

When SnapDrive for Windows is used to perform a LUN connect on the destination the `App` volume in SecondaryClusterB as part of LUN connect operation, it performs a SnapMirror break and restores the `App` volume using the latest consistent SDW Snapshot copy or the rolling Snapshot copy. When the restore operation is completed, it changes the `App` volume access permission from read-only mode to read/write mode. When you connect to any LUN residing on the `App` volume, SnapDrive for Windows performs the `snapmirror break` operation on the whole volume. As best practice always group similar application LUNs under same volume to be part of SnapMirror failover during planned or unplanned event.

When SnapDrive for Windows is involved in failover to destination SecondaryClusterB system with no rolling Snapshot copies on the `App` volume, the SDW connect command performs a `snap restore` using a consistent Snapshot copy created by SnapDrive for Windows on the `App` volume. If you do not have any consistent Snapshot copies created by SnapDrive for Windows, then LUN connect fails, displaying the below error in SnapDrive for Windows. The only way to perform SnapMirror failover is by using the Data ONTAP CLI explained in section 6.

```
SecondaryClusterB> snap list -V App

%/used       %/total  date          name

0% ( 0%)    0% ( 0%)  Jul 11 16:33  SecondaryClusterB(0118056828)_App.3

0% ( 0%)    0% ( 0%)  Jul 11 16:33  SecondaryClusterB(0118056828)_App.2

PrimaryClusterB> snap list -V App

%/used       %/total  date          name

----------  ----------  ------------  --------

0% ( 0%)    0% ( 0%)  Jul 11 16:33  SecondaryClusterB(0118056828)_App.3

0% ( 0%)    0% ( 0%)  Jul 11 16:33  SecondaryClusterB(0118056828)_App.2
```

Suppose all rolling Snapshot copies created by SnapDrive for Windows are deleted by user or by using `snap autodelete` and the SnapDrive for Windows created consistent Snapshot copy `App.0` is 24 hours older than the current `SecondaryClusterB(0118056828)_App.3` SnapMirror Snapshot copy.

When SDW LUN connect operation is performed on the SecdHost server, SnapDrive for Windows performs `snap restore` using the `App.0` Snapshot copy. This restores data that is one day older to the `App` volume on SecondaryClusterB system, and it deletes the `SecondaryClusterB(0118056828)_App.3` SnapMirror Snapshot copy as its time stamp is newer than the `App.0` Snapshot copy. At this point, data in SecondaryClusterB is old, and when failback is performed, the new data in PrimaryClusterA is deleted as well.

In this situation, you can perform a manual SnapMirror break on the the `App` volume in SecondaryClusterB and then map the App1 LUN to SecdHost server.

```
PrimaryClusterB> snap list -V App

%/used       %/total    date           name

----------  ----------  ------------  --------

0% ( 0%)     0% ( 0%)  Jul 11 16:33    SecondaryClusterB(0118056828)_App.3

0% ( 0%)     0% ( 0%)  Jul 10 16:33    App.0
```

PrimHost server is accessing an application using LUNs created under the `App` volume in PrimaryClusterB, and igroups are created under both PrimaryClusterB and SecondaryClusterB storage system with SecdHost as a standby server at the destination site and is used only when disaster occurs.

If there is a disaster in the `App` volume in PrimaryClusterB, perform failover and failback using SnapDrive for Windows as follows:

Step	Command
1	On PrimHost: 1. Stop application access to LUNs and shut down the server or quiesce application to hold I/O until you perform failover: **`c:\sdcli disk disconnect -p PrimaryClusterB:/vol/App/App1`** 2. Unmap LUNs mapped to `PrimApp` igroup that have access to PrimHost server on PrimaryClusterB storage. This makes sure that application will not perform write in PrimaryClusterB leading to duplicate data.
2	On SecondaryClusterB: 1. Perform `snapmirror quiesce` operation on the `App` volume in SecondaryClusterB so that the replicated data getting transferred according to the SnapMirror schedule or SnapDrive for Windows SnapMirror update is applied to the `App` volume in SecondaryClusterB and stops further SnapMirror updates: **`SecondaryClusterB>snapmirror quiesce App`** **`SecondaryClusterB>lun map /vol/App/App1 SecdApp`** On SecdHost: 1. SDW connect performs the `snapmirror break` and `volume restore` operation using latest rolling Snapshot copy created using SDW SnapMirror update on the `App` volume in SecondaryClusterB which will perform failover by changing read-only to read/write access permission on the `App` volume in SecondaryClusterB as part of disk connect operation. 2. Start the application: **`c:\ sdcli disk connect -p SecondaryClusterB:/vol/App/App1 -d w -IG SecdHost SecdApp dtype dedicated`**
3	On PrimaryClusterB: The `snapmirror resync` operation on the `App` volume in PrimaryClusterB, replicates all changed data blocks to the `App` volume in SecondaryClusterB, which was part of failover back to the `App` volume in PrimaryClusterB. Doing the extra update in this step minimizes the time the application needs to be offline for switching back to the primary site.

Step	Command
	PrimaryClusterB>snapmirror resync –S SecondaryClusterB:App App **PrimaryClusterB>lun map /vol/App/App1 PrimApp** **PrimaryClusterB>snapmirror break App** **PrimaryClusterB>snapmirror update –S SecondaryClusterB:App –w PrimaryClusterB:App** On PrimHost: After the resync and updates SnapMirror operations performed on PrimaryClusterB system connect to App1 LUN residing in PrimaryClusterB under the App volume: **C:\sdcli disk connect –p PrimaryClusterB:/vol/App/App1 –d w –IG PrimHost PrimApp dtype dedicated** On SecdHost: Stop application access to App1 LUN: **snapdrive storage disconnect –fs PrimaryClusterB:/vol/App/App** On PrimHost: Start the application.
4	On SecondaryClusterB: After the application has started in PrimHost, perform a flip resync in SecondaryClusterB so that it reverses the replication relationship. Then, unmap App1 LUN in the App volume in SecondaryClusterB. Perform a few SnapMirror updates to replicate the changes performed on the App volume in PrimaryClusterB. **SecondaryClusterB>snapmirror resync –S PrimaryClusterB:App App** **SecondaryClusterB>lun unmap /vol/App/App1 SecdApp** **SecondaryClusterB>snapmirror update –S PrimaryClusterB:App App**
5	On SecondaryClusterB: Remove the relationship from the secondary storage system to the primary storage system, as this was only used to transfer the data back to the primary storage system. This relationship makes the SnapMirror status command output somewhat confusing: **snapmirror release App PrimaryClusterB:App**
6	On PrimHost: After you perform SnapMirror release operation in SecondaryClusterB system which had a duplicate entry caused by `flip resync` operation, will result in the `(0118056873)_App.3` and `PrimaryClusterB(0118056873)_App.2` SnapMirror Snapshot copies on the App volume in PrimaryClusterB and SecondaryClusterB. To delete these duplicate Snapshot copies to free up space, enter: **sdcli snap update_mirror -d m**

6	On PrimaryClusterB:
	snap list –V App
	`%/used  %/total  date  name`
	`0% ( 0%)  0% ( 0%)  Jul 14 13:55 SecondaryClusterB(0118056828)_App.3 (snapmirror)`
	`0% ( 0%)  0% ( 0%)  Jul 14 13:55  @snapmir@{CA72D864-E191-4562-AA6C-61629CB014BC}`
	`1% ( 0%)  0% ( 0%)  Jul 14 13:53  @snapmir@{174375A7-B03F-4613-A2FA-D1F558C7D5F7}`
	`9% ( 8%)  0% ( 0%)  Jul 14 10:45  PrimaryClusterB(0118056873)_App.3`

Step	Command
	(snapmirror) 9% (0%) 0% (0%) Jul 14 10:44 PrimaryClusterB(0118056873)_App.2 The PrimaryClusterB(0118056873)_App.3, PrimaryClusterB(0118056873)_App.2, PrimaryClusterB(0118056873)_Webservice.3, and PrimaryClusterB(0118056873)_Webservice.2 Snapshot copies are left as is. Their time stamp is three hours older than the current SecondaryClusterB(0118056828)_App.3 (SnapMirror) and SecondaryClusterB(0118056828)_Webservice.3 (SnapMirror) SnapMirror Snapshot copies, which can be deleted after you perform release operation. These SnapMirror Snapshot copies can be deleted to free up space: **snap delete –V App PrimaryClusterB(0118056873)_App.3** **snap delete –V App PrimaryClusterB(0118056873)_App.2**

7 STORAGE SPACE SIZING CONSIDERATIONS

It is important to size storage space before setting up the source and destination SnapMirror systems to make sure SnapMirror update will not fail due to the lack of storage space in destination system. There are various requirements to be considered before sizing storage space. The storage sizing differs with synchronous and asynchronous replication modes and if you are using SnapDrive for Windows to drive SnapMirror update in asynchronous mode. SnapDrive for UNIX does not support any SnapMirror operations.

This section describes various scenarios with default space guarantee and thin provision volumes on SnapMirror systems. It also talks about how to size storage if the destination volume is a FlexClone volume.

The below tables should be referred to for all naming conventions throughout this section. Table 10 and Table 11 list the default Snapshot copies created in synchronous and asynchronous replication modes. Table 12 lists the SDW rolling Snapshot copies and the default SnapMirror Snapshot copies in asynchronous replication mode. Table 13 lists the aggregate, volume, and LUN layouts on source and destination systems.

Table 10) Synchronous replication mode.

PrimaryClusterA (Source)	SecondaryClusterA (Destination)
SecondaryClusterA(0118056828)_space.2 SecondaryClusterA(0118056828)_space.1	SecondaryClusterA(0118056828)_space.2 SecondaryClusterA(0118056828)_space.1

Table 11) Asynchronous replication mode.

PrimaryClusterA (Source)	SecondaryClusterA (Destination)
SecondaryClusterA(0118056828)_space.2	SecondaryClusterA(0118056828)_space.2 SecondaryClusterA(0118056828)_space.1

Table 12) SDW rolling Snapshot copies.

PrimaryClusterA (Source)	SecondaryClusterA (Destination)
@snapmir@{A9152684-D327-48A8-8915-E3C43AF59AB3} @snapmir@{896B9B2E-60AA-4EEF-B32E-BEC063B55113} SecondaryClusterA(0118056828)_space.0	@snapmir@{A9152684-D327-48A8-8915-E3C43AF59AB3} @snapmir@{896B9B2E-60AA-4EEF-B32E-BEC063B55113} SecondaryClusterA(0118056828)_space.1 SecondaryClusterA(0118056828)_space.0

Table 13) Aggregate and volume logical layout.

PrimaryClusterA (Source)		SecondaryClusterA (Destination)	
Aggregate Name	Size (GB)	Aggregate Name	Size (GB)
PrimAggr1	120	PrimAggr1	120
Volume Name	Size (GB)	Volume Name	Size (GB)
TestVol	100	TestVol	100

The various scenarios to consider before performing storage sizing in a SnapMirror environment are as follows:

The VSM SnapMirror mode on the source PrimaryClusterA volume1 always retains one SnapMirror Snapshot copy, and on SecondaryClusterB volume1 it retains two SnapMirror Snapshot copies.

If SnapDrive for Windows is used in Windows for driving SnapMirror update, it creates two rolling Snapshot copies on source PrimaryClusterA and destination SecondaryClusterB volume1. These two rolling Snapshot copies are always retained.

SnapDrive for Windows rolling Snapshot copies and SnapMirror Snapshot copies are created with the same time stamp all the time, so both Snapshot copies point to the same data blocks. Therefore, when considering retaining number of Snapshot copies during storage sizing, you can size only for two Snapshot copies in the source and destination SnapMirror systems. For example, to retain 10 Snapshot copies in the source, two Snapshot copies must be considered for storage space sizing for SnapMirror and SDW rolling Snapshot copies. If SnapDrive for UNIX application is used, only one SnapMirror Snapshot copy must be considered for storage space sizing on SnapMirror source volumes and two SnapMirror Snapshot copies on destination volumes.

If the source and destination SnapMirror volumes are FlexClone volumes, created by the user, or by using SnapMirror, or SDW rolling Snapshot copies, Data ONTAP applies a hard lock on these Snapshot copies. SnapMirror or SnapDrive for Windows cannot delete these Snapshot copies during the next update, which forces them to create new Snapshot copies. Therefore, when performing storage sizing you need to consider the space consumed by locked Snapshot copies.

By default, the SnapMirror source volume autosize is `on` and option `try_first=volume_grow`. When the available free space is 20%, Data ONTAP increases the volume using the value defined with `-m` option in increments defined using `-i` option defined in `vol autosize`.

For example, suppose the volume size is 100GB with space and fractional reserve set to zero, and the LUN with snap reserve enabled to 20%. In this case the available space is 80GB. If the available space reduces to 60GB, then volume autosize is triggered. If `-m` value is set to 150GB with the `-I` value set to 10GB, then volume will be increased by 10GB. Now available free space is 40GB, and volume grows maximum up to 150GB.

If SnapMirror source PrimaryClusterA volume grows due to above reason, the next SnapMirror update will fail, as the volume size on SnapMirror destination and source volume is not same. The only way to resolve this problem is to grow the destination volume, or you can create destination volume to maximum potential source volume might grow in SecondaryClusterB.

The `snap autodelete` command cannot delete Snapshot copies locked by FlexClone on source PrimaryClusterA and destination SecondaryClusterA volumes even if the commitment option is set to `disrupt`. This scenario has to be considered when retaining number of Snapshot copies during storage sizing in SnapMirror environment when FlexClone exist.

7.1 STORAGE SIZING WITH DEFAULT SPACE GUARANTEE ON VOLUMES

This section describes how much storage space is consumed if you have created volume and LUN using the default setting. Table 14 shows the default space guarantee settings used during volume creation. Table 13 shows the two LUNs and volume on source PrimaryClusterA and destination SecondaryClusterA.

Table 14) Default space guarantee.

Space Guarantee	Volume
LUN Reservation	ON
Space Reservation	20% of Volume Size
Fractional Reservation	100% of LUN Size

Suppose the `TestVol` volume on PrimaryClusterA and SecondaryClusterBare is created with default space guarantee setting (Table 14).

Suppose LUN1 of 30GB is already 50% full before SnapMirror is initialized on the `TestVol` volume in SecondaryClusterB system. SnapMirror starts replicating data from the `TestVol` volume in PrimaryClusterB by creating the `SecondaryClusterB(0118056873)_TestVol.0` Snapshot copy in the `TestVol` volume in PrimaryClusterB, which reserves 30GB space from `TestVol`. At this point only 20GB is available on `TestVol` volume, as default fractional reserve is set to 100% and space reserve set to 20% as shown in Table 14.

When SnapMirror initialization is completed, the `TestVol` volume in PrimaryClusterB and SecondaryClusterB will have LUN1 with 50% data occupied and `SecondaryClusterB(0118056873)_TestVol.0` SnapMirror Snapshot copy, which reserves 30GB space from `TestVol`.

As the fractional reserve by default is set to 100% (Table 14), this makes sure that the volume has space for Snapshot copies if the LUN is completely overwritten. If the fractional reserve space is set to zero, the volume might run out of space as Snapshot copies consume space. The LUNs might go offline or result in write failures, which causes applications to hang until LUNs are back online to serve I/O.

If both storage systems have Data ONTAP 7.3 or higher running, you can set space guarantee on destination volume in SecondaryClusterB. For versions earlier than Data ONTAP 7.3, the destination space guarantee is disabled by default and is only enabled after the `snapmirror break` operation is performed, which the inherent space guarantee set on parent volume from PrimaryClusterB.

Suppose on LUN1, 15GB of data is overwritten by the application, which causes the `SecondaryClusterB(0118056873)_TestVol.0` SnapMirror Snapshot copy in the `TestVol` volume in PrimaryClusterA to consume 15GB from available space to prevent point in time data. At this point, the available space is 5GB.

Suppose the application writes an additional 12GB of data on LUN1. Now LUN1 is 27GB full. Suppose the user creates a `Testsnap.1` Snapshot copy using SnapDrive or CLI on in the `TestVol` volume in PrimaryClusterA or if after a certain time interval SnapDrive for Windows created rolling Snapshot copies `@snapmir@{896B9B2E-60AA-4EEF-B32E-BEC063B55113}` and `@snapmir@{A9152684-D327-48A8-8915-E3C43AF59AB3}`. After a certain time interval, LUN1 is overwritten by more than 50%, which is around 27GB, due to this, the `@snapmir@ {A9152684-D327-48A8-8915-E3C43AF59AB3}`, `snapmir@{896B9B2E-60AA-4EEF-B32E-BEC063B55113}` and `Testsnap1` Snapshot copies point to the same data blocks and consumes 27GB. At this point, the available space and snap reserve space are completely consumed. At this point of time `TestVol` reports that the volume is full and no Snapshot copies can be created. If `snapmirror update` fails if triggered using CLI or using SnapDrive for Windows. This will make sure that even if LUN1 is completely overwritten, the `TestVol` volume has enough space for the `Testsnap.1` Snapshot copy, which holds point in time data at the time of creation, and consumes 2GB of fractional reserve space. Though there is 27GB free fractional reserve space, Data ONTAP will not allow creation of new Snapshot copies, as `Testsnap.1` Snapshot copy is hanging on the `TestVol` volume.

The best practice to follow when performing storage space sizing with fractional reserve 100% and space reserve 20% and LUN reservation enabled is to consider rate of change on LUNs, snap reserve space, and available space on the volume. If LUN size is greater than available space on the volume and rate of change on LUNs is greater than 50%, there will be lower number of Snapshot copies retained on volume in

SnapMirror environment. In this case you can retain a few Snapshot copies to make sure that the available space and snap reserve space on the volume are free to hold user created and SnapMirror Snapshot copies.

7.2 LUN SIZE SMALLER THAN AVAILABLE SPACE WITH FRACTIONAL RESERVE SPACE SET TO ZERO

This section describes how to size storage space on the source and destination SnapMirror volumes when setting fractional reserve and space reserve space to zero. When the LUN size is smaller than the available space, you can retain more Snapshot copies on the volume than if the LUN size is greater than 50% of available space, as explained in section 7.

By changing this setting, Data ONTAP does not reserve fractional space that is 1x LUN size after creating the first Snapshot copy on the volume. However, you must size storage well in advance to retain more Snapshot copies on the volume without the LUNs going offline, or encountering write failures. If volumes run out of space under these situations, the workaround is described below.

Table 15 and Table 16 show the space guarantee settings and logical view of storage on both source PrimaryClusterA and destination SecondaryClusterB systems.

Table 15) Default space guarantee settings.

Space Guarantee	Volume
LUN Reservation	ON
Space Reservation	0% of Volume Size
Fractional Reservation	0% of LUNs Size

Table 16) Logical view of storage.

PrimaryClusterA/SecondaryClusterB	
Aggregate1 Size	120GB
Volume1 Size	100B
Space Reserve	0GB
Fractional Reserve	0GB
Available Space	80GB
LUN1 Size	20GB

There are various scenarios explained below when rate of change on LUNs is 25%, 50%, and 75%, and the number of Snapshot copies that can be retained on the SnapMirror volumes before the volume runs out of space, causing the LUN to go offline, or causing write failures.

Assume the rate of change on LUN is 25% or 50% or 75% and LUN1 with 20GB is created on a Volume1 of 100GB size on source PrimaryClusterA system as shown in Table 16.

- If the rate of change on the LUN is 25%, which is 5GB for 15 times during any time frame (day, month, or year), you can retain 15 Snapshot copies in the `TestVol` volume in PrimaryClusterA.
- If rate of change on the LUN is 50% for seven times during any time frame (day, month, or year), that is 10GB, you can retain seven Snapshot copies in the `TestVol` volume in PrimaryClusterA.
- If the rate of change on the LUN is 75% at 5 times during any time frame (day, month, or year), that is 15GB in this scenario; then you can retain five Snapshot copies in the `TestVol` volume in PrimaryClusterA.

- If SnapDrive for Windows is driving SnapMirror update, there will be an extra Snapshot copy in source PrimaryClusterA and Destination SecondaryClusterB system, which has to be considered when retaining Snapshot copies in the `TestVol` SnapMirror volume.
- If there is a situation where you retain more Snapshot copies than defined above and if rate of change on LUN changes by more than 25% or 50% or 75% by application, if storage space is not increased before a new Snapshot copy is created, the LUN may go offline or cause write failures as no free space available in the `TestVol` volume in PrimaryClusterA. This situation is caused due to improper proper storage sizing.

7.3 LUN SIZE GREATER THAN AVAILABLE SPACE WITH FRACTIONAL RESERVE SPACE SET TO ZERO

This section describes how to size storage space on the source and destination SnapMirror volumes when you set fractional reserve and space reserve space to 0%.

When LUN size is greater than 50% of available space, you can only retain fewer user created SnapMirror or SnapDrive created Snapshot copies on the volume when compared with LUN size smaller than 50% of available space, as explained in section 7. However, you must size storage well in advanced to retain the number of Snapshot copies on the `TestVol` volume without the LUNs going offline.

Table 17 and Table 18 show the space guarantee setting and logical view of storage on both source PrimaryClusterA and destination SecondaryClusterB systems.

Table 17) Default space guarantee.

Space Guarantee	Volume
LUN Reservation	ON
Space Reservation	0% of Volume Size
Fractional Reservation	0% of LUNs Size

Table 18) Logical view of storage.

PrimaryClusterA/SecondaryClusterB	
Aggregate1 Size	120GB
Volume1 Size	100B
Space Reserve	0GB
Fractional Reserve	0GB
Available Space	50GB
LUN1 Size	50GB

There are various scenarios explained below when rate of change on LUNs is 25%, 50%, and 75%, the number of Snapshot copies that can be retained on the SnapMirror volumes before the volume runs out of space causing the LUN to go offline. It also describes the precautionary steps to follow in a SnapMirror environment.

Suppose the rate of change on LUN is 25% or 50% or 75%. Suppose LUN1 with 50GB is created on a `TestVol` volume of 100GB size on source PrimaryClusterA system as shown in the table above.

- If rate of change on the LUN is 25% and occurs thrice during any time frame (day, month, or year), that is 12.5GB; you can retain only three Snapshot copies on `TestVol` SnapMirror volume on PrimaryClusterA system.
- If rate of change on the LUN is 50% and occurs once during any time frame (day, month, or year), that is 25GB; you can retain one Snapshot copy in the `TestVol` volume in PrimaryClusterA.

- If rate of change on the LUN is 75% and occurs once during any time frame (day, month, or year), that is 37.5GB; you cannot retain any Snapshot copy in the `TestVol` volume in PrimaryClusterA, not even SnapMirror Snapshot copies.
- If SnapDrive for Windows is driving SnapMirror update, there is an extra Snapshot copy in source PrimaryClusterA and Destination SecondaryClusterB system, which has to be considered when retaining the Snapshot copy in the `TestVol` SnapMirror volume.
- To retain more Snapshot copies than defined above and if rate of change on a LUN is more than 25% or 50% or 75%, if the storage space is not increased before new Snapshot copy consumes space, then LUN might go offline or cause write failure as no free space is available on the `TestVol` volume in PrimaryClusterA system. This situation is caused due to improper storage sizing.

7.4 LUN SIZE SMALLER OR GREATER THAN AVAILABLE SPACE WITH FRACTIONAL RESERVE SPACE SET TO ZERO WITH FLEXCLONE

This section decribes how to size storage space when FlexClones volumes are used in the source or destination SnapMirror systems. FlexClone locks the Snapshot copy created. The `snap autodelete` command cannot delete these locked Snapshot copies, even if the `commitment` option it set to `disrupt`. Before deciding on the number of FlexClone volumes to be created in a SnapMirror environment, you must plan storage space; otherwise, you will end up in with LUN offline.

If your LUN size is greater or smaller than available space on the volume, you will only be able to retain fewer Snapshot copies to retain as discussed in section 7. In this situation if any retained Snapshot copies (SnapMirror or SnapDrive created are locked due to FlexClone volumes) and if rate of change on the LUN changes on the volume and aggregate with no free space available LUN offline mode will follow.

To grow the volume when there is no free space available on the aggregate, you must destroy FlexClone volumes or split the FlexClone volume on the source or destination SnapMirror volumes. This situation occurs due to poor storage sizing.

7.5 BEST PRACTICE GUIDELINES DURING VOLUME FULL SITUATION

This section explains various strategies to be followed during when the volume runs out of space when proper storage sizing is not planned. This sections describes the implication and obligation when SnapMirror Snapshot copies are deleted in a SnapMirror environment. Before you set these options, `enable snap autodelete` and `volume autosize`.

This section describes scenarios with various `snap autodelete` options to be considered when the volume runs out of space.

SCENARIO 1: THE TRY_FIRST VOLUME OPTIONS WITH THE SNAP_DELETE OR VOLUME_GROW AND SNAP AUTODELETE OPTIONS ENABLED ON LUNS SMALLER THAN THE AVAILABLE VOLUME SPACE

Suppose the LUN sizes are smaller than the available space and if rate of change on the LUN changes 25%, 50%, or 75% as explained in section 7, there are various strategies to be followed to tune the `snap autodelete` options.

If the LUN size is smaller than the available space, you can retain more Snapshot copies than if the LUN size is greater than 50% of available space. If you retain more Snapshot copies and if the rate of change on LUN is 25%, 50%, or 75%, the Snapshot copies will start consuming space. If the volume space is full, the LUN might go to offline mode or writes will fail.

To resolve this problem, either increase the volume space or delete the older Snapshot copies.

To grow the volume, you must consider rate of change on LUN. For example, if 20GB LUN is created on volume of 100GB and If rate of change is 25%, you can retain 15 Snapshot copies; 50% you can retain 7 Snapshot copies; 75% you can retain 5 Snapshot copies, as explained in section 7. However, If you retain more Snapshot copies and rate of change on LUN is 25%, 50%, or 75%, you must specify the maximum size volume in the `vol autosize` option using the `-m` option and specify the increments using the `-I` option.

Check whether you have free space available on the aggregate where the volume is been created as shown in Table 19; otherwise, the LUN may go offline, or you may encounter write failure.

Based on the maximum numbers of Snapshot copies to be retained, and the rate of change on LUN, you can plan the maximum volume size and its incremental growth.

The `vol autosize` setting (Table 19) can be referred on a LUN of 20GB size with different rate of change created on the 100GB volume with 80GB available space as shown in section 7.

Note: When the volume space at the primary system has been exceeded, either manually grow the volume at the secondary system using `vol size` command or create SecondaryClusterB volume with size to the potential volume size PrimaryClusterA volume can be grown.

Table 19) vol autosize options.

LUN Size (GB)	Rate of Change (%)	Volume Size (GB	Volume Available Size (GB)	Aggregate Size (GB)	Aggregate Available Free Space (GB)	Maximum Number of Snapshot Copies Retained	Potential Number of Snapshot Copies Retained	Vol Autosize Options
20	25	100	80	120	20	15	19	Vol autosize TestVol –m 120g –i 5g
20	50	100	80	120	20	7	11	Vol autosize TestVol –m 120g –i 10g
20	75	100	80	120	20	5	7	Vol autosize TestVol –m 120g –i 15g

If the aggregate does not have free space available due to poor storage sizing, write operations might fail on LUN if there are more Snapshot copies retained than available space on the volume as defined in Table 19. Delete Snapshot copies to release space so that LUN offline or write failure situations never occur. When you configure these options, make sure that you do not delete SnapMirror base Snapshot copies or SDW rolling Snapshot copies if driving SnapMirror update using SnapDrive for Windows.

In Windows environment where SnapDrive for Windows application is involved in `snapmirror update`, set the order so that the rolling Snapshot copy is the last to be deleted if there is no available free space. To do this, in the `defer_delete` option, specify the prefix that sets the last Snapshot copy to be deleted as the rolling Snapshot copy. In the `delete_order` specify `oldest_first,` which makes sure that Snapshot copies deleted have an older time stamp, since rolling Snapshot copies are always recycled and have the latest time stamp (see Table 20**)**. In a SnapMirror environment where there is no SnapDrive for Windows application involved, specify the prefix option as SecondaryClusterB, which is SnapMirror base Snapshot copy, and the `defer_delete` option should be set as `prefix`. In the `delete_order` specify `oldest_first`, since SnapMirror recycles and always is the latest one.

The `target_free_space` option should be defined considering rate of change on LUN. If LUN changes by 25%, 50%, or 75% by setting accordingly `snap autodelete` will stop deleting Snapshot copies after the available space is reached. For example, if the LUN size is 20GB and available size in the volume is 80GB and rate of change on the LUN is 25% as discussed in section 9.2, `snap autodelete target_free_space` is set to 25% and `snap autodelete` will stop deleting the Snapshot copy after available space on the volume is reached at 75GB. This makes sure it will delete one Snapshot copy if volume full condition occurs.

To select the `snap autodelete commitment` option as either `try` or `disrupt`, consider the available free space on the volume and number of Snapshot copies to be retained.

For instance, if there is no available space and if `snap autodelete` is enabled with commitment options set to `try`, SnapMirror Snapshot copies cannot be deleted; however, it can delete SnapDrive or user created Snapshot copies to release space. This makes sure that `snapmirror update` does not fail for the

next update. If the `commitment` is set to `disrupt`, it can delete SnapMirror Snapshot copies, following which you must perform a fresh SnapMirror initialization on the destination.

In LUN size is smaller than available space situation, you will never encounter situation volume running out of space since you can retain more Snapshot copies wherein you can delete user created Snapshot copies first. This makes sure SnapMirror Snapshot copies can still be retained.

Table 20) Storage sizing information.

	State	Commitment	Trigger	target_free_space	delete_order	Defer_delete	Prefix
PrimaryClusterB> snap autodelete TestVol	on	try	volume	5%, 10%, or 15%	oldest_first	prefix	@snapmir@
PrimaryClusterB> snap autodelete TestVol	on	disrupt	volume	5%, 10%, or 15%	oldest_first	prefix	User Created

SCENARIO 2: VOLUME OPTION TRY_FIRST WITH SNAP_DELETE OR VOLUME_GROW AND THE SNAP AUTODELETE OPTION ON LUN GREATER THAN 50% OF THE AVAILABLE VOLUME SPACE

When the LUN size is greater than 50% of the available space, and rate of change on the LUN is 25%, 50%, or 75% as explained in section 7, there are various implications caused in SnapMirror environment. This section helps you decide different strategies to tune `snap autodelete` options in SnapMirror environment to make sure LUN offline mode or write failures never occur.

If the LUN size is greater than 50% of available space, you can retain fewer Snapshot copies than if LUN size is smaller than available space. If you retain more Snapshot copies and rate of change on LUN occurs, Snapshot copies start consuming space, and if available space is full on the volume in this situation, LUN might go to offline mode or write failure will occur.

To resolve this problem you need to either increase the volume space or delete Snapshot copies.

Refer to the `vol autosize` settings below for a LUN of 50GB size with different rate of change created on the 100GB volume with 50GB available space as shown in section 9.3. Table 21 shows the number of Snapshot copies that can be retained and the potential number of Snapshot copies that you can retain in future. In case of rate of change on the LUN is greater than 50%, then you can retain only one Snapshot copy, and in 75% then you cannot retain any Snapshot copies, which is created by SnapMirror or SnapDrive.

Note: When volume at the Primary system is full, manually increase volume at the Secondary system using the `vol size` command. Alternatively, you can create SecondaryClusterB volume with size to the potential volume size that will be grown in PrimaryClusterA.

Table 21) Snapshot copies retained.

LUN Size (GB)	Rate of Change (%)	Volume Size (GB	Volume Available Size (GB)	Aggregate Size (GB)	Aggregate Available Free Space (GB)	Maximum Number of Snapshot Copies Retained	Potential Number of Snapshot Copies Retained	Vol Autosize Options
50	25	100	50	120	20	3	5	Vol autosize TestVol –m 120g –I 13g
50	50	100	50	120	20	1	2	Vol autosize TestVol –m 120g –i 20g
50	75	100	50	120	20	0	0	Vol autosize TestVol –m 120g –i 20g

If there is a situation where the aggregate does not have free space available due to poor storage, the option `try_first` set to `volume_grow` does not solve any problem. The only option is to delete Snapshot copies to free up space on the volume by configuring the snap autodelete options to delete Snapshot copies. When you configure these options, you need to consider the following conditions in a SnapMirror environment.

If snap autodelete option commitment is set to try and `target_free_space` is set to 15% or 25% or 40% according to the rate of change on the LUN as shown in Table 22, you have different obligations when volume is running out of space due to LUN overwrite in a SnapMirror environment.

Table 22) Storage sizing information.

	State	Commitment	Trigger	target_free_space	delete_order	Defer_delete	Prefix
PrimaryClusterA> snap autodelete TestVol	on	Try	volume	15%	oldest_first	prefix	@snapmir@
PrimaryClusterA> snap autodelete TestVol	on	Disrupt	volume	25%	oldest_first	prefix	testsnap
PrimaryClusterA> snap autodelete TestVol	on	Disrupt	volume	40%	oldest_first	prefix	@snapmir@

1. If rate of change on the LUN is 25%, note the following:

- You can retain maximum of five Snapshot copies on the source SnapMirror volume as shown in Table 21. These Snapshot copies can be created using SnapDrive or SnapMirror.
- If you are using SnapDrive for Windows for driving SnapMirror updates, it always retains two rolling Snapshot copies, and SnapMirror always retains one Snapshot copy on the source, which is recycled after each SnapMirror update.
- The time stamp of these rolling Snapshot copies for the latest rolling Snapshot copy and SnapMirror Snapshot copy is the same. This makes sure both Snapshot copies point to the same data blocks on the volume and thus consume the same data if LUN rate of change is 25%. The second rolling Snapshot copy will consume space as it will be recycled in the next SnapMirror update. These rolling Snapshot copies should be considered when sizing your storage in a SnapMirror environment; in this case, it should be included among the five Snapshot copies that can be retained. If SnapDrive for Windows is not used, then only one SnapMirror Snapshot copy is included among the maximum of five Snapshot copies that can be retained.
- When there are more than five Snapshot copies in `TestVol` volume inPrimaryClusterA, and rate of change on the LUN is 25% (that is, 12.5GB change occurs more than five times), then without `snap autodelete` enabled on `TestVol` volume you will run out of space and LUN will go offline.
- To resolve this problem, configure `snap autodelete` options as shown in Table 22 in a SnapMirror environment. The `snap autodelete` option in the `Testvol` should be turned on, which gets triggered when `TestVol` volume is 98% full. This condition occurs when you have more than five Snapshot copies on the TestVol volume and rate of change on the LUN is 25%, as discussed in section 7.
- When the `snap autodelete commitment` option is set to `try` it cannot delete the SnapMirror Snapshot copy, which is the base Snapshot copy for the next SnapMirror update. It is a rare condition if LUN is overwritten with rate of change of 25% for five times. If this condition is met, then you need to set the `commitment` option to `disturb`, which will allow deleting SnapMirror Snapshot copies to prevent the LUN from hanging. If base SnapMirror Snapshot copy is deleted, you need to perform a fresh SnapMirror initialization.
- When the `snap autodelete trigger` option is set to `volume` the `snap autodelete` is triggered when volume space is 98% full. When `target_freespace` option set to 15% causes `snap autodelete` to delete Snapshot copies. It stops deleting Snapshot copies when the volume space is 85% full (that is, 85GB). In this scenario if rate of change is 25% on the LUN, which is 12.5GB, it deletes only one Snapshot copy due to creation of extra Snapshot copy which consumes space.

- The `snap autodelete prefix` option set to `testsnap,` which is the SnapDrive created Snapshot copy naming convention with the `defer_delete` option set to `prefix,` deletes the `testsnap` Snapshot copy at the end, if the volume is getting full. In this scenario if you only have the `Testsnap` Snapshot copies and SDW rolling Snapshot copies, it first deletes rolling Snapshot copies and then Testsnap Snapshot copies and finally the SnapMirror Snapshot copy. By using this setting, you can prevent deleting testsnap Snapshot copies, which are used as a backup purpose; however, the rolling Snapshot copies can be deleted, which are recycled.

2. If rate of change is 50% before executing `snap autodelete`, consider the following:

- You can retain a maximum of two Snapshot copies on the source SnapMirror volume, as shown in Table 21. SnapDrive or SnapMirror can create these Snapshot copies.
- If you use SnapDrive for Windows for for `snapmirror update`, it always retains two rolling Snapshot copies, and SnapMirror always retains one Snapshot copy on the source, which is recycled after each SnapMirror update.
- The time stamp of these rolling Snapshot copies (the latest rolling Snapshot copy and SnapMirror Snapshot copy) is the same. This makes sure both Snapshot copies point to the same data blocks on the volume and thus consume same data if LUN rate of change is 50%. The second rolling Snapshot copy will consume space since it will be recycled in the next SnapMirror update. Hence, consider these rolling Snapshot copies when sizing your storage in a SnapMirror environment.
- If rate of change on the LUN is 50% (that is, 25GB), you can retain only two Snapshot copies on `TestVol` (see Table 21). However, when SnapDrive for Windows is involved, you can retain two rolling Snapshot copies and the SnapMirror base Snapshot copy. If SnapDrive for Windows is not used, you can retain one SnapMirror Snapshot copy on the `TestVol`. This is has to be considered for storage sizing.
- Suppose LUN is already 30GB full if LUN starts to overwrite by 50%, which is 25GB SnapMirror and older rolling Snapshot copy and latest SnapMirror Snapshot copy starts consuming 25GB each from the `Testvol` volume in PrimaryClusterA. Now available space left on Testvol is 20GB. Assume SnapDrive has created the `Testsnap` Snapshot copy and the LUN is overwritten by 50% again, then the `Testsnap` Snapshot copies start consuming space. When volume is 98% full, then `snap autodelete` is triggered to delete Snapshot copies to free up space. Table 22 shows `snap autodelete` configuration options so that the LUN does not go offline and write failures do not occur.
- If the `snap autodelete commitment` option is set to `disturb`, it allows deleting SnapMirror, user created, and rolling Snapshot copies, if the volume is running of space. The `snap autodelete` deletes the SnapMirror created Snapshot copy at the end though; `snap autodelete` option `defer_delete` is set to `prefix` and prefix is set to `@snapmir@` (Table 22). Using this setting `snap autodelete` deletes the rolling Snapshot copies and the user created Snapshot copies to prevent LUN write failures and offline mode.
- SnapMirror Snapshot copies cannot be deleted if the `snap autodelete commitment` is set to `try`. If volume is around 98%, the `snap autodelete` command deletes the SnapDrive created Snapshot copies and SDW rolling Snapshot copies if it exists. However, if it still cannot free up space, and if the SnapMirror Snapshot copies start to consume space when LUN rate of change is 50%, write failures on LUN occur unless the volume size is increased or the SnapMirror Snapshot copy is deleted. The only solution left is to delete SnapMirror Snapshot copy, in which case you to must perform fresh SnapMirror initialization.
- Suppose during SnapMirror update period, SnapMirror locks the SnapMirror and SDW rolling Snapshot copies. If the user creates a `testsnap1` Snapshot copy and if the rate of change on the LUN is 50% (that is, around 25GB), the `testsnap1` SDW rolling Snapshot copy and the SnapMirror Snapshot copies start to consume space. When the volume is 98% full, the `snapmirror update` fails, and SnapMirror and SDW rolling Snapshot copies are deleted. If the writes are continuous, the LUN will go offline as there is no space in the volume. You need to perform a fresh SnapMirror initialize operation.
- In the above scenarios if due to poor storage sizing there is no free space available in the aggregate and if the volume space cannot be increased, delete SnapMirror Snapshot copy. In this case, perform a fresh SnapMirror initialization.

3. If rate of change is 75% before using `snap autodelete`, note:

- You cannot retain any Snapshot copy on the source SnapMirror volume as shown in Table 21.
- If you are using SnapDrive for Windows to drive the SnapMirror update, it always retains two rolling Snapshot copies, and SnapMirror always retains one Snapshot copy on the source, which is recycled after each SnapMirror update.
- The time stamp of these rolling Snapshot copies, the latest rolling Snapshot copy, and the SnapMirror Snapshot copy is same, which makes sure both Snapshot copies point to the same data blocks on the volume and thus consume same data if LUN rate of change is 75%. The second rolling Snapshot copy will be consuming space as it is recycled during the next SnapMirror update. Consider these rolling Snapshot copies when sizing your storage in a SnapMirror environment.
- If rate of change on the LUN is 75% (that is, 40GB), you cannot retain any Snapshot copy on the `TestVol` volume (Table 21). However, in SnapDrive for Windows, you end up retaining two rolling Snapshot copies and one SnapMirror Snapshot copy.
- The `snap autodelete` option commitment is set to `disturb`, which allows deleting SnapMirror Snapshot copy if volume is running out of space. If rate of change on the LUN that is 75% before next SnapMirror update, at some point when volume is 98% full due to Snapshot copies (SnapMirror or SDW rolling) starts consuming space, `snap autodelete` deletes the first SnapDrive and SDW rolling Snapshot copies. If it still cannot free up space, then it deletes SnapMirror Snapshot copy. This makes sure that the LUN does not go offline or encounter write failure. The disadvantage is you need to perform fresh SnapMirror initialization.
- If the `snap autodelete commitment` set to `try`, then SnapMirror Snapshot copy cannot be deleted when rate of change on the LUN is 75%. When the volume is around 98% full, it will trigger `snap autodelete` to delete Snapshot copies to release space. However, this setting does not allow deleting the SnapMirror created Snapshot copies and will start to consuming space. This causes the LUN to hang, and causes write failures, until the volume is increased or if the SnapMirror Snapshot copy is deleted.
- If the above scenario occurs during `snapmirror update` period, SnapMirror locks SnapMirror and SDW rolling Snapshot copies if used. If at this point the LUN rate of change is 75%, which is around 40GB when volume is 98% full, then `SnapMirror update` fails, and SnapMirror and SDW rolling Snapshot copies will be deleted and LUN will go to offline mode due to lack of space. You must perform a fresh SnapMirror initialization.
- In all above scenarios if due to poor storage sizing there is no free space available in the aggregate, and if the volume cannot be grown and only option left is to delete SnapMirror Snapshot copy in which case you need to perform fresh SnapMirror initialization.

SCENARIO 3: THE TRY_FIRST WITH SNAP_DELETE OR VOLUME_GROW OPTION AND SNAP AUTODELETE OPTION ON LUNS SMALLER OR GREATER THAN 50% AVAILABLE VOLUME SPACE WITH FLEXCLONE VOLUMES

The `snap autodelete` option with `commitment` set to `disturb` cannot delete Snapshot copies locked by FlexClone, so you need to consider number of Snapshot copies that are locked by FlexClone for storage sizing in SnapMirror environment.

If the LUN size is smaller or greater than 50% of available space with snap autodelete options, as explained above, shows the number of Snapshot copies you can retain on the TestVol SnapMirror volume if LUN rate of change is 25% or 50% or 75%.

During storage sizing you need to consider how many Snapshot copies will be locked by FlexClone in the source or destination SnapMirror volume.

When FlexClone volumes are created on the source or destination system in case of LUN size smaller or greater than 50%of available space. Table 23 shows maximum number of Snapshot copies that can be locked by FlexClone on volume of 100GB that can grow to 120GB on aggregate of 120GB size. If you create more FlexClone volumes by locking Snapshot copies, more than the number mentioned below in SnapMirror environment, then LUN might go offline.

Table 23) Maximum number of Snapshot copies locked by FlexClone.

LUN Size (GB) Smaller than Available Space	Rate of Change (%)	Maximum Number of Snapshot Copies Retained	Maximum Number of Snapshot Copies Locked by FlexClone
20	25	19	18
20	50	11	10
20	75	7	6
50	25	5	4
50	50	2	1
50	75	0	0

If you have created FlexClone volumes on the destination volume, make sure you do not delete the parent Snapshot copy in the source volume. However, if there no space in the source volume, `snap autodelete` deletes the Snapshot copy at the source. If `snapmirror update` is started, it fails, as it cannot delete the locked Snapshot copy in the destination (see section 7).

1. FlexClone volume created on the destination volume:

   If LUN size is smaller than available space and rate of change is 75%, which is 15GB (Table 23), you can retain only eight Snapshot copies on the source or destination volume if you have created a FlexClone volume using oldest Snapshot copy in the destination volume. An extra Snapshot copy created violates the maximum number of Snapshot copies, at the same time LUN on the source volume is overwritten by 75% when volume is 98% full, `snap autodelete` is triggered. It deletes the oldest Snapshot copy on the source free space for the new Snapshot copy. When SnapMirror update has started, it has to delete the oldest Snapshot copy on the destination. That is normal behavior; however, SnapMirror cannot delete the Snapshot copy as FlexClone hard locks it and the SnapMirror operation fails.

2. FlexClone volume created on the source volume:

   If LUN size is greater than 50% of available space and rate of change is 75%, which is 40GB, as shown in Table 23, you cannot retain any Snapshot copy on the source or destination volume. If user has created Snapshot copies and at LUN on the source volume is overwritten by 75%, when volume is 98% full `snap autodelete` is triggered to delete Snapshot copies. However, it cannot delete the Snapshot copy, which is hard locked by FlexClone. Hence it deletes the SnapMirror Snapshot copy to prevent the LUN from going offline or cause write errors. If there is a SnapMirror update progress and LUN is overwritten by 75% and when volume is 98% full, `snap autodelete` is triggered, which stops SnapMirror replication and deletes SnapMirror Snapshot copies, which prevents the LUN from going offline. To prevent the volume from running out of space, delete the SnapMirror Snapshot copies and perform a fresh initialization of the SnapMirror relationship.

8 SNAPMIRROR NETWORK SIZING SPECIFICATION

This section lists the requirements to determine SnapMirror network sizing for synchronous or asynchronous replication. It also describes how to collect initial data to design the design network link for carrying data modes between sites.

Table 24 lists the nominal bit rates and bandwidths for commonly used link types. Based on these link rates you can determine which replication mode is feasible. You also must consider the TCP/IP overhead and storage usage on the effective bandwidth.

Table 24) Bit rates and bandwidths.

Link Type	Nominal Bit Rate (MB/s)	MB/s	MB/m	GB/h
T1	1.5	0.18	10.8	0.63
T3	45	5.6	336	19.6
100Mb	100	12.5	750	43.9
OC3	155	19.3	1158	67.85

8.1 SNAPMIRROR IN SYNCHRONOUS MODE

This section explains how to gather information that helps to size your network and decide whether synchronous SnapMirror replication mode can be used.

The synchronous SnapMirror replication mode synchronously replicates writes from the source NVLOG RAM to the destination NVLOG RAM. After the data transfer is completed, an acknowledgement is sent from the destination NVLOG RAM. This is known as NVLOG forwarding. At this point, data is not yet written to the disk. Once the destination NVRAM is half-full or 10 seconds to previous Consistency Point (CP) time, Data ONTAP creates a tetris that computes RAID parity information and data blocks to be written to the disk. It also forwards the same tetris data to a destination system until the CP data is written to the destination; this is known as CP forwarding. If there is a delay in NVLOG or CP forwarding due to network or storage error synchronous replication, the synchronous mode falls back to the asynchronous mode. As there are two writes, one during NVLOG forwarding and another during CP forwarding, you must consider 2X of data written in synchronous replication mode.

Before deciding on the link requirement, gather the following data:

- Identify the LUNs to be replicated to the destination.
- Identify the amount of data written during the peak time in a day by examining the LUN stats output. This provides an estimate of the average data written, which helps size the network.

  Table 25 shows the average data written measured during the peak time.

```
PrimaryClusterB*> lun stats -i 1 -o /vol/space/lun1
Read Write  Other QFull   Read  Write Average   Queue     Partner  Lun
Ops   Ops   Ops             kB     kB Latency     Length   Ops     kB
0   392   0    0    9328      261.01  100.09   0    0 /vol/TestVol/lun1
0   459   0    0    0 112644  186.71   96.00   0    0 /vol/ TestVol /lun1
0   398   0    0    0 108288  258.44  101.06   0    0 /vol TestVol /lun1
0   467   0    0    0 111874  214.38   94.01   0    0 /vol/ TestVol /lun1
0   417   0    0    0 107264  227.64   99.09   0    0 /vol/TestVol /lun1
0   416   0    0    0 111816  238.22   97.00   0    0 /vol/ TestVol /lun1
0   361   0    0    0  87560  191.20  103.07   0    0 /vol/ TestVol /lun1
0   418   0    0    0 111232  315.53   96.05   0    0 /vol/ TestVol /lun1
```

Table 25) LUN stats output.

LUNs	Time Taken (Hours)	Data Written (KB)	Rate of Change (MB)
/vol/TestVol/Lun1	9AM–10AM	4322	4
/vol/TestVol/Lun1	10AM–11AM	5399	5
/vol/TestVol/Lun1	5PM–6PM	5899	5.8
/vol/TestVol/Lun1	6PM–7PM	6020	6

The above example (Table 25) shows that the application is writing at an average rate of 6MB/s to the LUN.

- To choose a network link type, consider the following:
  - 2X (NVLOG and CP) forwarding, which is 12MB/s link bandwidth
  - The maximum delay of 2ms that synchronous replication can support
  - The distance between the two sites; if distance is more, the delay exceeds 2ms

If you are using T1 link type, theoretically it can transfer data at the rate of 0.18MB/s bandwidth (see Table 24). However, it is required to transfer 12MB/s (NVLOG and CP) data. In this case, SnapMirror synchronous replication will fall back to asynchronous mode, since NVLOG and CP forwarding will be delayed over 2ms.

To implement synchronous SnapMirror replication, you require at least 100MB link type to provide a potential bandwidth of 12.5MB/s.

8.2 SNAPMIRROR IN ASYNCHRONOUS MODE

This section describes how to gather initial information to help size the network link requirements for SnapMirror asynchronous mode. It also helps you design scheduling policy for different data sets that use common network link bandwidth between source and destination SnapMirror systems.

The asynchronous replication mode replicates only changed data blocks in the form of Snapshot copy. This is achieved by performing a full replication of the source volume by creating a SnapMirror based Snapshot copy and transferring the base Snapshot copy to the destination SnapMirror volume.

SnapMirror is not application aware when it performs asynchronous replication when triggered by SnapMirror scheduler or SnapMirror update operations. NetApp recommends driving asynchronous replication through SnapManager applications, which are available for Oracle, Exchange, VMware, SQL Server, and other applications that are integrated with SnapMirror to perform restartable copies at the destination when disaster strikes; however, if SnapManager is not utilized, you can refer to section 8.1 to perform asynchronous replication.

Before deciding the link requirement, gather the following data:

- Identify the LUNs to be replicated to the destination.
- Identify the rate of change of data on the LUN during the peak time in a day. To do this, create an initial Snapshot copy on the volume where LUN is created, and then create subsequent Snapshot copies at various peak times during the day. Compare the Snapshot copies created during peak times and perform snap delta with Snapshot copies.
- Check that you have space available on the volume for the Snapshot copies; otherwise the LUN might go offline.
- The snap delta output is shown below, which is used to find the change data blocks by subtracting from the last Snapshot copy to the next subsequent Snapshot copies.

```
PrimaryClusterB*> snap delta space

From Snapshot     To                    KB changed  Time
--------------- --------------------  ----------- ------------ ------------
```

```
Test.6          Active File System
Test.5          Test.6                1802341
Test.4          Test.5                1450712
Test.3          Test.4                2295020
Test.2          Test.3                518224
Test.1          Test.2                1036372
```

Table 26) Snapshot copy details.

Volume Name	Snapshot Copy Name	Time Created	Commands
TestVol	Test.1	9AM	snap create –V TestVol Test.1
TestVol	Test.2	10AM	snap create –V TestVol Test.2
TestVol	Test3	11AM	Snap create –V TestVol Test.3
TestVol	Test.4	5PM	snap create –V TestVol Test.4
TestVol	Test.5	6PM	snap create –V TestVol Test.5
TestVol	Test.6	7PM	Snap delta TestVol Test.3 Test.6

Table 27) Rate of change of Snapshot copies.

Snapshot Name	Time Created (Hours)	Rate of Change (KB)	Rate of Change (MB)
Test.1 and Test.2	9AM–10AM	1036372–518224	506
Test.2 and Test.3	10Am–11AM	518224–1295020	758
Test.3 and Test.4	5PM–6PM	2295020–1450712	824
Test.5 and Test.6	6PM–7PM	1450712–1802341	343

When performing asynchronous replication, you must design how often it is required to schedule replication and determine the rate of change on the LUN between these schedules. Based on this data, perform the network sizing.

Table 26 lists the Snapshot copies created at peak times during the day. Table 27 shows the rate of change for data blocks between Snapshot copies. From the above tables, you can conclude that around 300–850MB of data is required to replicate data from the source to destination system.

For a better a recovery time objective (RTO), the replication should be performed frequently during the day. For example, by replicating every 30 minutes in a day as opposed to 4 hours, in a disaster scenario, you can increase your chances of retaining the most current data.

The following example describes how to choose a network link that provides the best recovery point objective (RPO) and RTO.

From Table 27, the maximum rate of change is shown to be 850MB.

The T1 link can transfer data at the rate of 10.8MB/minute or 0.6 GB/hour (see Table 24). Hence, to replicate 850MB of data, it will take 1 hour and 18 minutes. The T3 link type can transfer data at 336MB/minute or 19.6GB/hour. Hence, to replicate 850MB, it will take 2 minutes and 6 seconds. Therefore, for better RPO, is recommended to use the T3 link type.