Technical report: Oracle 10*g* R2 and IBM System Storage N series with SnapMirror

Disaster recovery planning

Document NS3455-0

October 12, 2007

Table of contents

Abstract **3**
Executive summary **3**
Introduction **3**
Purpose and scope **4**
Understanding DR language **5**
Business continuity plan (BCP) 5
DR plan (DRP) 5
Recovery time objective (RTO) 5
Recovery point objective (RPO) 5
Disaster tolerance 5
High availability (HA) 5
Archive logging 6
Media recovery 6
Instance or crash recovery 6
Application-coordinated Snapshot copy 6
SnapMirror: A quick overview **6**
Requirements and assumptions **8**
General assumptions 8
Environment assumptions 8
Security and access issues 9
Network connectivity 9
Mount the IBM N series storage system's root volume to the database server 9
Enable `rsh` access from the database server 10
Required permissions on the volumes to be used for the database 10
Mount and change ownership of the file system on the mount point 10
Network and storage infrastructure **11**
Configuration details **12**
Oracle Database layout 12
Database setup 12
Configure SnapMirror 13
Database recovery using SnapMirror Async **14**
Database recovery using SnapMirror Async and Sync **21**
Conclusion **25**
Stored procedure script 26
Application-coordinated snapshot script (do_snap) 27
Application-Coordinated snapshot script (begin_bkup) 27
Application-Coordinated snapshot script (end_bkup) 27
Trademarks and special notices **28**

Abstract

As 24x7 enterprise operations are growing, a protracted interruption in the ability to access data disrupts business operations. A well-planned, rehearsed, and tested disaster recovery solution can save time and money for an organization by offering smaller recovery windows with no or acceptable data loss. IBM System Storage N series offers an array of proven, low-cost, and simplified data protection and disaster recovery options for enterprise data.

Executive summary

As the need for uninterrupted availability of enterprise data is growing, more businesses are striving for 24x7 system availability and can't afford any downtime. In this era of continuous information availability, the complete and rapid recovery from a disaster is not nice to have, but is a necessity.

Whether you're a business, a government agency, a healthcare organization, or an educational institution, you must ensure that you're prepared for when, not if, disaster strikes. A protracted interruption in your organization's ability to access data will disrupt business operations. This can lead to the loss of customers and revenue, a drop in share price, or possible noncompliance fines for failing to protect and/or provide information promptly when required.

A well-planned, rehearsed, and tested disaster recovery (DR) solution can save time and money for your organization by offering small recovery windows with no or acceptable data loss. IBM® System Storage™ N series offers an array of proven, low-cost, and simplified data protection and recovery solutions for your organization's data.

Introduction

As recent world events have proven, disaster can strike anywhere and anytime. The question you need to answer is how well prepared your organization is to recover in the event of a disaster. Though most business houses, even the small ones, are meticulous when it comes to thinking ahead and devising future strategies to take the company forward, a common mistake is ignoring the possibility of a disaster crippling the organization. Critical data loss could well be the fatal prescription for an organization that is otherwise doing well.

Survey after survey shows they're in the majority: over 50% of companies make no effort whatsoever to prevent avoidable disasters or to put into place strategies for recovering from outage events that can't be avoided. Of those companies that do plan, fewer than 50% actually test the strategy they develop, which is like having no strategy at all.

From time to time different groups, including professional accounting organizations, universities, and the US government, study the results of disasters and downtime on business and employment. The results are never good. Among the most-published results are:

- 93% of companies that suffer a significant data loss are out of business within five years (US Bureau of Labor).
- 43% of US businesses never reopen after a disaster, and 29% close within two years (University of Wisconsin).
- 30% of computer users say they spend the equivalent of one week per year reconstructing lost data (3M Corporation).

All these statistics suggest that organizations must plan up-front for data recovery. The DR plan is a corporate survival kit in the eventuality of disaster. The continued operations of an organization depend on management's awareness of potential disasters, the ability to develop a DR plan to minimize disruptions of critical functions, and the capability to recover and restore vital business operations successfully and quickly.

A DR plan is a comprehensive statement of consistent actions to be taken before, during, and after a disaster. The plan should be documented and tested to ensure the continuity of operations and availability of critical resources in the event of a disaster. The primary objectives of the plan are to protect the organization in the event that all or part of its IT services are rendered unusable, minimize the disruption to critical business operations, ensure some level of organizational stability, and provide an orderly recovery after a disaster.

Incidents like 9/11 and Hurricane Katrina give businesses a chance to see their DR plan in action. While some companies pass with flying colors, the plans of others are exposed as incomplete, unrealistic, and technologically flawed. Those companies with untested or poorly tested plans will eventually discover that they aren't as protected as they thought they were.

There are several approaches to protect data and maintain data availability in the face of hardware, software, or even site failures. Backups provide a way to recover lost data from an archival medium (tape or disk). Redundant hardware technologies also mitigate the damage caused by hardware failures. Data replication or mirroring provides a third mechanism to ensure data availability and minimize downtime. The IBM System Storage N series with SnapMirror® provides a fast and flexible enterprise solution for data replication over local area (LAN), wide area (WAN), and (FC) networks. If a disaster strikes at a source site, the replicated mission-critical data can immediately be made available at a remote site, ensuring uninterrupted operation and data availability.

Purpose and scope

The scope of this document is limited to recovery of an Oracle10*g* R2 single-instance database. The two DR solutions covered in this document are:

- DR using SnapMirror Async
- DR using a mix of SnapMirror Async and Sync

Using SnapMirror, it is now possible to recover from a disaster at a remote physical location. If critical data is mirrored to a different physical location, a serious disaster no longer means prolonged data unavailability. The mirrored data can be made available to clients across the network until the damage caused by the disaster is repaired. Recovery may include recovery from corruption, natural disaster at the source site, accidental deletion, sabotage, etc. SnapMirror also allows application server layer information to be replicated to the DR site. In the event of disaster, once the DR site is operational, all applications can be switched over to the servers at the DR site and all application traffic can be directed to these servers for as long as necessary to recover the source site. Once the source site is recovered, SnapMirror can be used to transfer the data efficiently back to the primary site. After the production site takes over normal application operation again, SnapMirror transfers to the DR site can resume without requiring a second complete data transfer.

Understanding DR language

Business continuity plan (BCP)

The BCP describes processes and procedures an organization puts in place to ensure that essential business processes continue during and after a disaster. Normally it takes into account the protection of the whole organization, including buildings, IT infrastructure, employees, and all other resources. The main objective of BCP is to prevent interruption of mission-critical services and to reestablish full functions as swiftly and smoothly as possible.

DR plan (DRP)

The DRP is a subset of the BCP and focuses solely on the protection and recovery of the mission-critical data and the IT infrastructure. It details step-by-step procedures and processes for the IT staff to follow during and after the disaster. It describes the recovery point and time objectives for each application.

Recovery time objective (RTO)

The RTO indicates the time spent in bringing the application up and resuming the operation after the disaster. It is also known as acceptable downtime after the disaster. The unit of measure for RTO is time, with values ranging from seconds to days or weeks. The lower the application's RTO value, the greater the organization's dependence on that particular application, and, consequently, the higher the recovery priority after the disaster.

Recovery point objective (RPO)

The RPO defines data currency. The unit of measure for the RPO is also time, with values ranging from seconds to days or weeks. It denotes how current the data should be after the recovery. In another words, it defines acceptable data loss from the point the disaster starts. The lower an application's RPO value, the greater the organization's dependence on that particular process, and, consequently, the higher the priority when recovering the systems after the disaster.

Disaster tolerance

Greater awareness of the need for DR is prompting application architects to build disaster readiness into the business systems they design. Disaster tolerance is a term used to signify a system with some ability to withstand major disruption. Several technologies are used to provide disaster tolerance, including hardware redundancy, data replication, server clustering, and remote data centers.

High availability (HA)

HA is an architecture that maximizes the data availability. It is a subcategory of DR. The ultimate disaster-tolerant system is classed as a high-availability (HA) system. The HA systems are designed to eliminate application downtime by using redundant hardware and networking components and specialized application and operating system software. HA systems can seamlessly route around failures in the computing infrastructure without affecting end-user access to data. The resilience of this system is often measured in terminology borrowed from the telecommunications industry. For example, a configuration that offers 99.999% availability (also known as five nines), can have only 5 minutes of planned or unplanned downtime in any a given year.

Archive logging

Archive logging is a database feature that enables retention of the transaction logs. The retained transaction logs are called archive logs. Using archive and active logs, a database roll-forward recovery is possible to any point in time before the failure occurred, rather than only to the point in time of a full backup. The archived logs can be moved off line and still be used for roll-forward recovery.

Media recovery

Media recovery is a user-initiated data recovery. It can be used to recover out-of-date or damaged datafiles, SPFile, or control files. The recovery is performed by applying archive logs followed by active logs. The various scenarios that require media recovery are:

- Restore datafiles from backup
- Restore control files from backup
- Datafiles are taken offline without OFFLINE NORMAL option
- Datafiles are out of date with the corresponding control file
- A database can't be opened if datafiles need media recovery.

Instance or crash recovery

Instance crash recovery process is a special form of recovery, which happens when a database instance is started for the first time after a crash (or SHUTDOWN ABORT). The crash recovery uses only online transaction logs and the goal is to bring the datafiles to a transaction-consistent state, preserving all committed changes up to the point when the instance failed.

Application-coordinated Snapshot copy

An application-coordinated snapshot copy is a snapshot copy created via IBM N series with Snapshot™ technology after putting a database in hot backup mode; it guarantees the database consistency. That means a database recovery is guaranteed from an application-coordinated snapshot copy.

SnapMirror: A quick overview

In order to protect your organization's data from disaster and ensure quick and smooth recovery, your data needs to be replicated to one or more other physical locations. IBM N series with SnapMirror technology allows data replication between two IBM N series storage systems. The IBM N series storage system from which data is transferred is referred to as the SnapMirror source, and the IBM N series storage system to which the data is transferred is referred to as the SnapMirror destination. The SnapMirror source and destination can be miles apart, provided that both IBM System Storage N series can communicate with each other across a network.

IBM N series with SnapMirror technology supports Volume SnapMirror (VSM) as well as qtree SnapMirror (QSM). SnapMirror source volumes and qtrees are writable data objects, but the SnapMirror destination volumes and qtrees are read-only, usually on a separate storage system. In the case of a disaster where the source volumes or qtrees go down, the replicated data at the destination volumes or qtrees can be made available by making the volumes or qtrees writable. The SnapMirror configuration details are maintained in a configuration file called snapmirror.conf, which resides on the destination IBM N series storage system. This file, along with information entered via the snapmirror.access option or the snapmirror.allow file, is used to establish relationships between specified source volumes or qtrees and the destination volume or qtree where the mirrored data is kept.

IBM System Storage N series with Data ONTAP® supports SnapMirror Async as well as SnapMirror Sync. In the SnapMirror Async mode, SnapMirror performs incremental, block-based replication as per the frequency defined in `snapmirror.conf` file. Write requests to the SnapMirror source are acknowledged as soon as they are written to its non-volatile random access memory (NVRAM) and are not delayed until the SnapMirror destination has received and/or processed the request. Performance impact on the SnapMirror source filer is minimal, as long as the system is configured with sufficient central processing unit (CPU) and disk I/O resources. The first and most important step in Async mode involves the creation of a one-time baseline transfer of the entire data set from the SnapMirror source system to the SnapMirror destination system. This is required before incremental updates can be performed. After the baseline transfer is complete, scheduled or manually triggered updates can occur. Each update transfers only the new and changed blocks from the source to the destination storage system. Because asynchronous replication is periodic, SnapMirror is able to consolidate writes and conserve network bandwidth, thereby minimizing the impact on write throughput and write latency.

IBM N series with Data ONTAP supports SnapMirror Sync to meet the very high availability requirements for certain environments where all data changes written to a production site must be replicated to a remote site synchronously. SnapMirror in Synchronous or Sync mode is a mode of replication that sends updates from the source volumes or qtree to the destination volumes or qtree as they occur, rather than according to a predetermined schedule. This guarantees that data written on the source system is protected on the destination even if the entire source system goes down. With synchronous mode, each time a transaction attempts to write data to disk, the data is sent to both the source and destination storage systems in parallel. It is not until both IBM N series storage systems have committed the data associated with the write operation to NVRAM that the system acknowledges that the transaction is complete. In other words, the application that initiated the write operation must wait until it receives the acknowledgement from both the source and destination storage systems before it can continue.

SnapMirror Semi-Sync mode is a variation of SnapMirror Sync mode and it provides a middle ground that keeps the source and destination file systems more closely synchronized than the asynchronous mode, but with less impact on application performance. Configuration of Semi-Synch mode is nearly identical to the configuration of Sync mode, with the exception being the addition of an option that specifies how many writes, seconds, or ops can be outstanding (unacknowledged by the destination system) before the source system delays acknowledging write operations from clients. Internally, Semi-Sync mode works identically to Sync mode in most cases. The only difference lies in how quickly client writes are acknowledged; the replication methods used are the same.

Note: The Sync and Semi-Sync modes are supported only with VSM.

For more details on SnapMirror deployment and Implementation, refer to the various SnapMirror deployment and implementation guides.

Requirements and assumptions

General assumptions

In order to take maximum advantage of the procedures described in this document, it is assumed that readers of this document are familiar with the following:

- Commands and operations of Data ONTAP and IBM System Storage N series
- Administration and operation of Oracle 10*g* R2 instance and database
- UNIX® system administration commands
- IBM N series with SnapMirror technology
- DR concept.

The IBM N series storage systems used to produce this document are loaded with Data ONTAP 7.1 and are licensed for network file sharing (NFS), FC protocol, iSCSI protocol, SnapMirror, and SnapMirror Sync products.

It is also assumed that the UNIX hosts used for the production database have Oracle 10*g* R2 software installed and a single database instance created.

In order to produce this document we used NFS protocol. If you are using a storage area network (SAN) environment, then make sure that database hosts have the following products installed and configured:

- Appropriate host bus adaptor (HBA)
- SanSurfer Utility (installed if HBA used is from Qlogic)
- IBM N series Host Attach/Support Kit.

Check IBM N series compatibility and configuration guides for IBM N series FC and iSCSI to find out about supported HBAs.

Environment assumptions

This document covers DR solutions offered by IBM N series for Oracle 10*g* R2 single database instance using SnapMirror Async as well as Sync technology. The scripts and process steps contained in this document may require significant modifications to run under your version of UNIX. The sample scripts in this document assume the following:

- The IBM N series storage system used for the SnapMirror source is `‘ntapsrc’`
- The IBM N series storage system used as the SnapMirror destination is `‘ntapdst’`
- The aggregate on the IBM N series storage systems used for database storage is `‘dbaggr’`
- The flexible volume used to store database data is `‘dbdata’`
- The flexible volume used to store database transaction logs is `‘dblogs’`
- The flexible volumes `dbdata` and `dblogs` reside in aggregate `dbaggr`
- At the database host, the mount points used are `/mnt/dbdata` and `/mnt/dblogs`
- Oracle Home resides on a Linux® (Red Hat® Enterprise Linux [RHEL 4]) database host.

It is also assumed that the database host used for accessing the database at the DR site has a similar setup to the database host on the primary site and has all required privileges to access the volumes on the SnapMirror destination IBM N series storage system.

Security and access issues

You need to make sure that the IBM System Storage N series with FlexVol™ volumes to be used for the database's data and transaction logs have their security style set appropriately. If the database host used is a UNIX host, then the security style must be set to 'UNIX.' The security style can be changed by executing the following command on the IBM N series storage system:

```
qtree security <volume name> unix
```

For example, to change the security style for a volume named `dbdata`, you would execute the following command on the IBM N series storage system:

```
qtree security /vol/dbdata UNIX
```

Network connectivity

You also need to make sure that the database hosts and IBM N series storage systems can communicate with each other through the network. This is done by making appropriate entries to the `/etc/hosts` files on the IBM N series storage system as well as on the database host systems.

- Add the following line to the database host's `/etc/hosts` file if it doesn't already exist:

  ```
  <IBMN storage system IP>  <IBMN storage system name>
  ```

  For example, to add an entry to the `/etc/hosts` file on the database server for the IBM N series storage system named `ntapsrc` that has IP address `10.32.70.134`, you would add the following line:

  ```
  10.32.70.134 ntapsrc
  ```

- Add the following line to the `/etc/hosts` file on an IBM N series storage system if it doesn't already exist:

  ```
  <database server IP>  <database server name>
  ```

  For example, to add an entry to the `/etc/hosts` file on the IBM N series storage system for a database server named `dbhost1` that has IP address `172.32.70.43`, you would add the following line:

  ```
  172.32.70.43 dbhost1
  ```

Mount the IBM N series storage system's root volume to the database server

In order to mount the IBM N series storage system's `root` volume to the database host and make necessary changes to some configuration files, the user `root` on the database sever must have access to the root volume `/vol/vol0.` For example, to grant access on a root volume named `/vol/vol0` on an IBM N series storage system to the user `root` on the database host system (`dbhost1`), you would execute the following command on the storage systems:

```
exportfs –p rw=dbhost1,root=dbhost1,anon=0 /vol/vol0
exportfs –a
```

Enable `rsh` access from the database server

If you intend to use `'rsh'` commands from your database host, then add the IP address of the database host to the `/etc/hosts.equiv` file on the IBM N series storage system. The entry should look similar to the following:

```
<hostsrc IP>
```

Required permissions on the volumes to be used for the database

Before you create a database on the FlexVol volumes on the IBM N series storage system, you need to mount them to a database host system. In order to mount FlexVol volumes, the user `root` on the database host must have access to them. Execute the following commands on the IBM N series storage systems to grant access on FlexVol volumes:

```
exportfs -p rw=<db host name>,root=<db host name>,anon=0 <volume name>
```

For example, to grant access on the FlexVol named `dbdata` to the user `root` on the database server named `dbhost1`, you would execute the following command:

```
exportfs -p rw=172.17.38.112,root=172.17.38.112,anon=0 /vol/dbdata
exportfs –a
```

Grant permission on all FlexVol volumes to be used for the database using the above command.

Mount and change ownership of the file system on the mount point

The FlexVol volumes to be used for the database need to be mounted on the database host system by executing the following command:

```
mount -o rw,bg,hard,nointr,rsize=32768,wsize=32768,tcp,vers=3,timeo=600 <IBMN
storage system name>:<volume name> <mount point>
```

For example, to mount a volume named `dbdata` on a mount point named `/mnt/dbdata`, you would execute the following command on the database server:

```
mount -o rw,bg,hard,nointr,rsize=32768,wsize=32768,tcp,vers=3,timeo=600
<ntapsrc>:/vol/dbdata /mnt/dbdata
```

To install Oracle and create a database successfully, you need to make sure that the file system on the mount points is owned by the user `'oracle'` on the database server. Change the ownership of the mounted volumes to the user `oracle` by executing the following command at the database server:

```
chown –R oracle:dba <mount point>
```

For example, to change the ownership of a file system on the mount point named `/mnt/dbdata`, you would execute the following command on the database server:

```
chown –R oracle:dba /mnt/dbdata
```

Network and storage infrastructure

The following network diagram shows a very simple and basic architecture for the Oracle database DR scenario using an IBM N series storage system at the back end and that was used to produces this document.

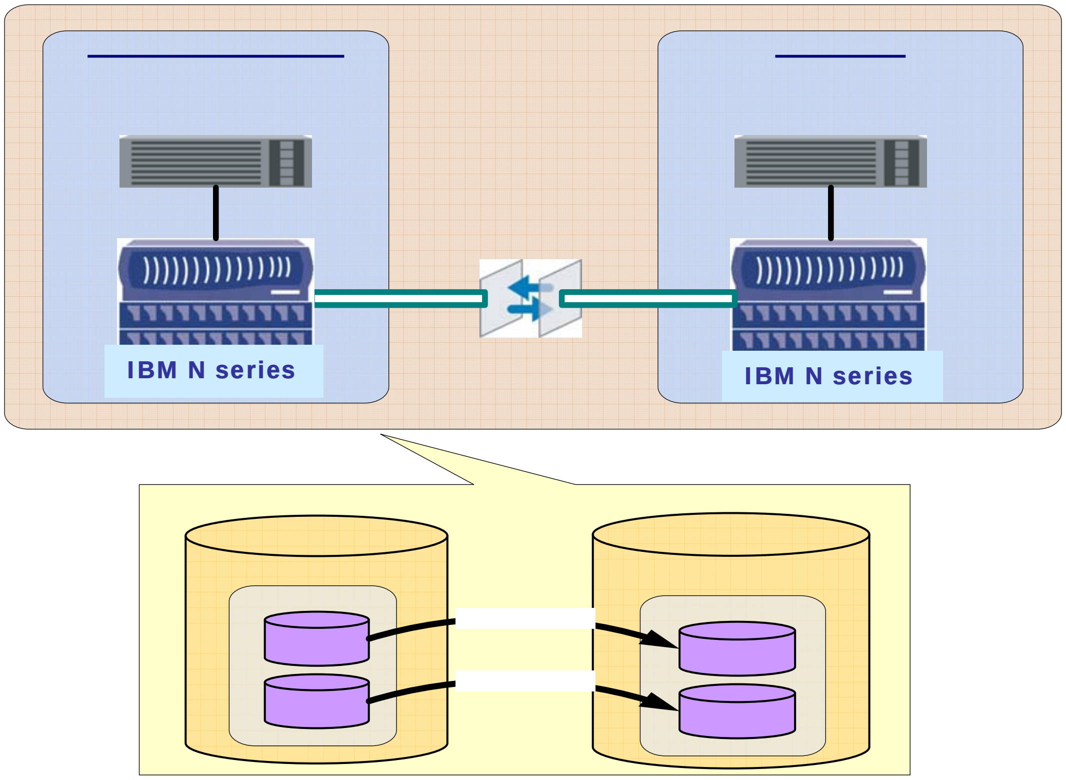

Figure 1. Network and storage infrastructure.

In this architecture, the Oracle Home resides on the database server. The datafiles, redo logfiles, and archive logfiles for the database reside on the FlexVol volumes on an IBM N series storage system. The volumes used for the database on the IBM N series storage system at the primary data center are replicated to a secondary location using SnapMirror. The secondary location is referred to as the DR site through out this document.

Primar

Data

Configuration details

In order to produce this document we used Oracle Database 10*g* R2, Enterprise Edition. The database host was running 32-bit Red Hat Enterprise Linux Kernel 2.6 on a Sun Fire V20Z dual process machine. For database storage, we used IBM N series. The environment details are:

- Database – Oracle 10*g* R2, single database instance (orcl)
- Database archive log – Enabled
- Oracle Home – Local on database Host (`/home/oracle/ora10g/product/10.2.0/db_1`)
- IBM N series storage system volumes used for the database data and transaction logs are `dbdata` and `dblogs`, respectively
- Soft links created to follow the OFA directory structure:

```
    ln -s /mnt/dbdata /home/oracle/ora10g/dbdata
    ln -s /mnt/dblogs /home/oracle/ora10g/dblogs
```

Oracle Database layout

Control files

```
/home/oracle/ora10g/dbdata/orcl/control01.ctl
/home/oracle/ora10g/dbdata/orcl/control02.ctl
/home/oracle/ora10g/dblogs/orcl/control03.ctl
```

Redo logs

```
/home/oracle/ora10g/dblogs/orcl/redo_01.log
/home/oracle/ora10g/dblogs/orcl/redo_02.log
/home/oracle/ora10g/dblogs/orcl/redo_03.log
```

Datafiles

```
/home/oracle/ora10g/dbdata/orcl/*.dbf
```

It is recommended that a control file copy should be stored on every disk drive that stores members of online redo log groups, if the online redo log is multiplexed. By storing control files in these locations, you minimize the risk that all control files and all groups of the online redo log will be lost in a single disk failure. For example, we have placed one control file named home/oracle/ora10g/dblogs/orcl/control03.ctl on the same volume where database redo logfiles reside.

Database setup

As described in section 6, it is assumed that you already have Oracle 10***g*** R2 software installed and a single database instance created, and that the database has its data and transaction logs on IBM N series storage system volumes. In order to complete the scenario discussed in this document, you need to complete the following steps:

1. Log in to the database and create a table by executing the following command:

   ```
   create table scott.tab1(col1 number not null, col2 character(240));
   ```

2. Create a database stored procedure by executing the script `demobld.sql`. The script can be found in Appendix A.

   ```
   sql>@demobld.sql
   ```

The procedure `demobld` inserts rows in the table `scott.tab1` and generates the database load.

Configure SnapMirror

To protect your organization's data using IBM N series with SnapMirror technology, you need to complete the following very simple SnapMirror configuration process steps:

1. Find the SnapMirror source and destination IBM N series storage systems. The source storage system already has an Oracle 10*g* R2 database on it. For example, we used an N series storage system named `ntapsrc` with a Oracle 10*g* R2 database on it, as the SnapMirror source and another N series storage system named `ntapdst` that is on the DR site as the SnapMirror destination.
2. Identify the volumes on the source IBM N series storage system that need to be replicated to one or more other IBM N series storage systems using SnapMirror.
3. Define a schedule for each SnapMirror relationship in the `/etc/snapmirror.conf` file, which resides on the SnapMirror destination's IBM N series storage system. The `/etc/snapmirror.conf` file controls where data is replicated from and how often the mirror is updated. The entry in the `/etc/snapmirror.conf` file should be in the following format:

   ```
   <source IBMN storage system>:<vol name | qtree path>  <destination IBMN storage
   system>:<vol name | qtree path>  '-' schedule
   ```

   Where the schedule is defined by [minute] [hour] [days of month] [days of week]

   A snippet from the `/etc/snapmirror.conf` file:

   ```
   ntapsrc:dbdata ntapdst:dbdata – 0,30 9-17 * 1-5
   ntapsrc:dblogs ntapdst:dblogs – 0,30 * * 1-5
   ```

   In the above snippet, the volume named dbdata on the IBM N series storage system named ntapsrc is replicated to the volume named dbdata on the IBM N series storage system ntapdst. The argument dash '-' means that data is replicated at the fastest rate possible. The IBM N series storage system ntapdst updates the volume dbdata at 30-minute intervals between 9 am and 5 pm, Monday through Friday. The asterisk (*) in the example means that the mirror is updated on every day of the month.

4. The SnapMirror feature needs to be enabled SnapMirror on the source as well as on the destination IBM N series storage system by runnng the following command:

   ```
   options snapmirror on
   ```

5. In order to complete a one-time baseline transfer, each SnapMirror relationship needs to be initialized by executing the following command from the SnapMirror destination IBM N series storage system:

   ```
   snapmirror initialize –S <source IBMN storage system>:volume <destination IBMN
   storage system>:volume
   ```

   For example, to initialize the baseline database transfer for the volume named dbdata, you would execute the following command on the SnapMirror destination IBM N series storage system:

   ```
   snapmirror initialize –S ntapsrc:dbdata ntapdst:dbdata
   ```

   After the baseline transfer, the SnapMirror updates can be triggered based on a predefined schedule in the /etc/snapmirror.conf file. It is also possible to trigger a SnapMirror update manually by executing the following command from the destination storage system:

   ```
   snapmirror update <volume name>
   ```

   For example, you would execute the following command on the SnapMirror destination IBM N series storage system to start the manual SnapMirror update for a volume named dbdata:

   ```
   snapmirror update dbdata
   ```

Database recovery using SnapMirror Async

SnapMirror Async transfers the changed data blocks to the SnapMirror destination based on snapshot copy comparison. On each SnapMirror update, a new snapshot copy is created at the SnapMirror source IBM N series storage system and it is compared against the snapshot copy from the previous SnapMirror update to determine the changes. After changes are determined, the changed data blocks are transferred to the destination IBM N series storage system and the old snapshot copy gets deleted. The SnapMirror update and application-coordinated snapshot schedule is summarized in the following, Table 1.

Flexible Volume	SnapMirror	App. Coordinated Snapshot
dbdata	**2 hrs**	**30 min**
dblogs	**30 min**	**10 min**

Table 1. SnapMirror and snapshot update schedule.

The snippet from the `/etc/snapmirror.conf` file looks somewhat similar to the following:

```
ntapsrc:dbdata ntapdst:dbdata – 0 0-22/2 * 1-5
ntapsrc:dblogs ntapdst:dblogs – 0,30 * * 1-5
```

In order to simulate disaster on the primary site and perform recovery on the DR site, you would need to complete the following steps:

1. Complete the SnapMirror base data transfer and run the load

a. Initialize the SnapMirror relationship for each volume used for the database and complete one-time baseline transfer by executing the following command on the destination
IBM N series storage system:

```
snapmirror initialize –S <source IBMN storage system>:<volume name>
<destination IBMN storage system>:<volume name>
```

For example, to initialize the baseline transfer for a volume named **dbdata**, you would execute the following command on the destination IBM N series storage system:

```
snapmirror initialize –S ntapsrc:dbdata ntapdst:dbdata
```

b. Monitor the SnapMirror status by executing the following command on the destination storage system:

```
snapmirror status
```

Let us assume our base line transfer started at 12:00 p.m. and completed at 12:25 p.m.

c. After creating a table and stored procedure as described in section 9, you need to connect to the database and start running the load by executing the following commands on the database server:

```
Connect scott/tiger
Execute scott.demobld(25000000);
```

d. Copy the `do_snap`, `begin_bkup`, and `end_bkup` scripts to a work directory on the database server and start creating application-coordinated snapshot copies by executing the `do_snap` script on the database server:

```
./do_snap
```

The above command can be executed as a cron job from the database server.

2. Simulate the disaster

Let us assume that the disaster strikes at 3:05 p.m. In order to simulate the disaster, we made the SnapMirror source IBM N series storage system unavailable by executing the following command:

```
Halt
```

At this point, the system state on the primary site will look somewhat similar to Figure 2.

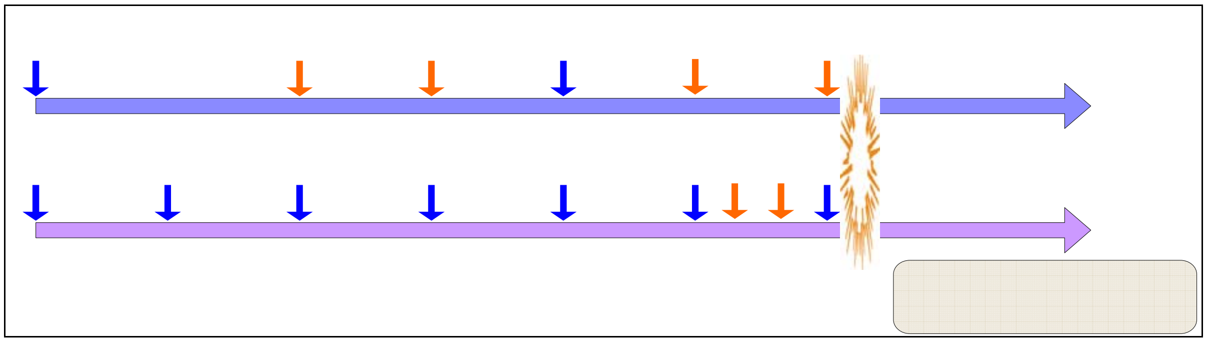

Figure 2. Database state at the primary site at the time disaster struck.

After a disaster, the DR site will be in SnapMirror snapshot consistent state. That means the volumes at the DR site will have the same data, including application-coordinated snapshot copies, as the corresponding source volume had at the time of successful SnapMirror update. For our test scenario, `dbdata` and `dblogs` are the SnapMirror destination volumes corresponding to the SnapMirror source volumes used for the database; therefore, these two volumes will be in the following states:

- `dbdata` is consistent with the source volume at the time of the SnapMirror update at 2.00pm
- `dblogs` is consistent with the source volume at the time of the SnapMirror update at 3.00pm

3. Recovery

After the initial transfer, the destination volumes are available to clients, but in a read-only state. The status of a destination will show that it is **snapmirrored.** A disaster makes the source unavailable; to use the destination volumes for writing as well as reading, you would need to end the SnapMirror relationship for each volume used for the database by executing the following command on the destination IBM N series storage system:

```
snapmirror break <volume name>
```

SM1

For example, to make a volume named `dbdata` writable, you would execute the following command on the IBM N series storage system:

```
snapmirror break dbdata
```

This command changes the destination volume's status from **snapmirrored** to **broken-off**, thus making it writable.

SM1

SM2

00 p.m.

30 p.m.

After a disaster, the destination volumes will be in SnapMirror snapshot consistency state with their corresponding source volumes used for the database's data and will have the same data, including application-coordinated snapshot copies, as the corresponding source had at the time of successful SnapMirror update. The database transaction logs volume is **sync snapmirrored**, and therefore is in NVLOG or CP consistency state based on the configuration. In this type of configuration, there are four ways of database recovery after the disaster:

- Recovery from the SnapMirror update points of the `dbdata` and `dblogs` volumes.
- Recovery from an application-coordinated snapshot copy of the `dbdata` volume and SnapMirror update point of the `dblogs` volume.
- Recovery from the SnapMirror updated point of the `dbdata` volume and an application-coordinated snapshot copy of the `dblogs` volume.
- Recovery from the application-coordinated snapshot copies of the `dbdata` and `dblogs` volumes.

a. Recovery from the SnapMirror update point

In this scenario of database recovery, the RPO (recovery point objective) is determined by the time from the last successful SnapMirror update of the transaction logs volume. The data after the last SnapMirror update of the transaction logs volume will be lost. The RTO (recovery time objective) will vary based on the transaction logs that need to be applied to recover the database. For example, in our DR scenario, the disaster struck at 3:05 p.m. and the last SnapMirror update for the `dblogs` volume occurred at 3:00 p.m. therefore, the RPO will be 5 minutes. Only 5 minutes worth of data needs to be reconstructed.

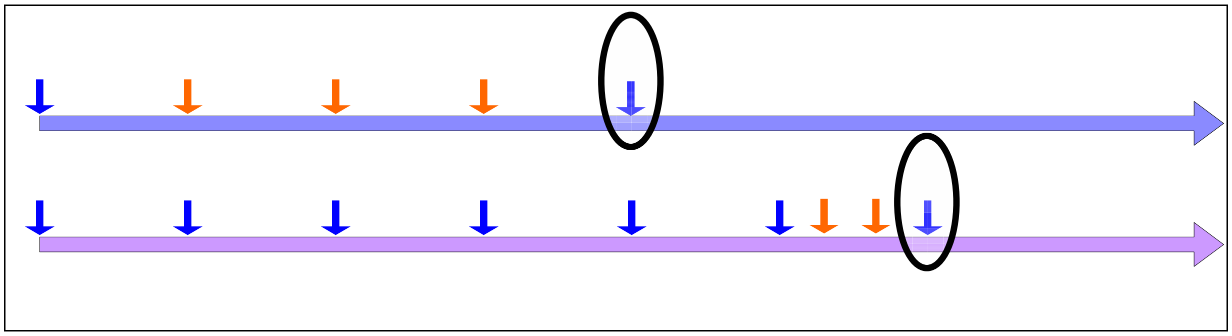
Figure 3. Recovery from a SnapMirror points.

In order to perform recovery from the SnapMirror update points for database data and transaction volumes, you would need to complete the following steps:

i. At the DR site, mount the volumes (used as the SnapMirror destination for the database volumes on the primary site) to a database server by running the following command:

```
mount -o rw,bg,hard,nointr,rsize=32768,wsize=32768,tcp,vers=3,timeo=600
<IBMN storage System Name>:<volume name> <mount point>
```

For example, to mount a volume named `dbdata` on a mount point named `/mnt/dbdata`, you would execute the following command on the database server:

```
mount -o
rw,bg,hard,nointr,rsize=32768,wsize=32768,tcp,vers=3,timeo=600
ntapdst:/vol/dblogs /mnt/dblogs
```

ii. The control file that resides on the transaction logs volume will have the latest database information, so you would copy that control file from the dblogs volume to the dbdata volume by executing the following commands on the database server:

```
cp /mnt/dblogs/orcl/control03.ctl
/home/oracle/ora10g/dbdata/orcl/control01.ctl
cp /mnt/dblogs/orcl/control03.ctl
/home/oracle/ora10g/dbdata/orcl/control02.ctl
```

iii. Start the database instance and recover the database by executing the following commands on the database server:

```
sqlplus / as sysdba
sql>startup mount
sql>recover database
sql>alter database open
```

After completing the above steps, the database recovery from the SnapMirror point of the database volumes will be complete.

b. Recovery from an application-coordinated Snapshot copy of the data volume

This type of recovery may become necessary in rare cases, such as when database recovery is not possible from the SnapMirror update point. In this type of situation, the database can be recovered using application-coordinated snapshot copies. The RPO will be the time from the last successful SnapMirror update of the `dblogs` volume. The RTO will vary based on the transaction logs that need to be applied to recover the database.

For example, in our test scenario, the RPO will be 5 minutes. Data after the last SnapMirror update point of the `dblogs` volume will be lost.

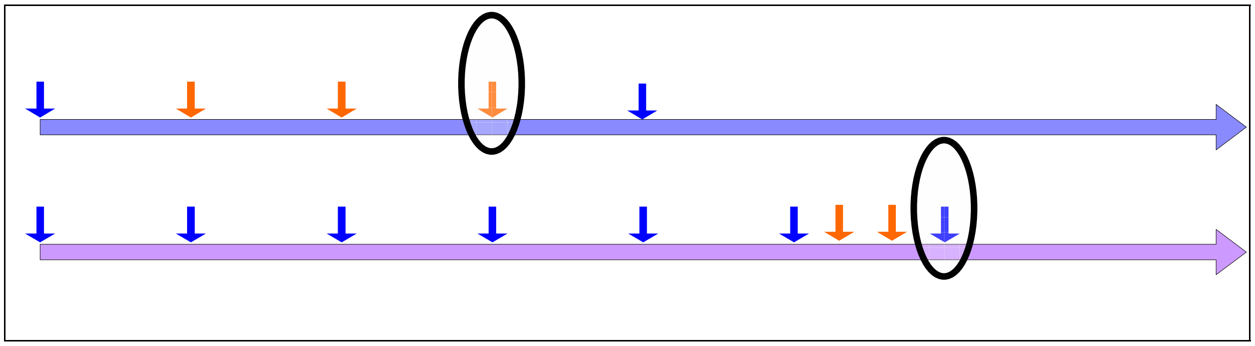

Figure 4. Recovery from an application-coordinated snapshot copy of the data volume.

To complete database recovery from the application-coordinated snapshot copy of the data volume and the SnapMirror update point of the transaction logs volume of the database, you would need to complete the following steps:

i. First you need to snap restore the volumes that hold data for the database from the application-coordinated snapshot copies by executing the following command on the IBM N series storage system:

```
snap restore -s <snapshot name> <volume name>
```

For example, to restore a volume named `dbdata` from an application-coordinated snapshot copy named `s3`, you would execute the following command on the

IBM N series storage system:

```
snap restore -s s3 dbdata
```

ii. Mount the IBM N series storage system volumes that hold the database's data and transaction logs to a database host by executing the following command:

```
mount -o rw,bg,hard,nointr,rsize=32768,wsize=32768, tcp,vers=3,timeo=600
<IBMN storage system name>:<volume name> <mount point>
```

For example, to mount a volume named `dbdata` on a mount point named `/mnt/dbdata`, you would execute the following command on the database server:

```
mount -o rw,bg,hard,nointr,rsize=32768,wsize=32768,tcp,vers=3,timeo=600
ntapdst:/vol/dbdata /mnt/dbdata
```

iii. The control file that resides on the transaction logs volume will have the latest database information, so copy that control file from the dblogs volume to the dbdata volume by executing the following commands on the database server:

```
cp /mnt/dblogs/orcl/control03.ctl
home/oracle/ora10g/dbdata/orcl/control01.ctl
cp /mnt/dblogs/orcl/control03.ctl
/home/oracle/ora10g/dbdata/orcl/control02.ctl
```

iv. Start the database instance and perform database recovery by executing the following commands on the database server:

```
sqlplus / as sysdba
sql>startup mount
sql>recover database
sql>alter database open
```

After completing the above steps, the database recovery from an application-coordinated snapshot copy of the database data volume and the SnapMirror point of the transaction logs volumes will be complete.

c. Recovery from an application-coordinated snapshot copy of the transaction logs volume

This type of recovery becomes useful in some cases, such as when recovery is desired to a point in time before the SnapMirror update of the transaction logs volume, or when the recovery is not possible from the SnapMirror update point of the transaction logs volume. In such cases, the application-coordinated snapshot copy can be used to recsver the database. The RPO (recovery point objective) is determined from the time the snapshot copy was created to the time the disaster started and the RTO (recovery time objective) will vary based on the transaction logs that need to be applied for the database recovery.

For example, in our DR scenario, the disaster stuck at 3:05 p.m. and an application-coordinated snapshot copy named `'S2'` from which we want to restore the `dblogs` volume was created at 2:50 p.m. therefore, the RPO will be15 minutes. The work done after the application-coordinated snapshot point will be lost.

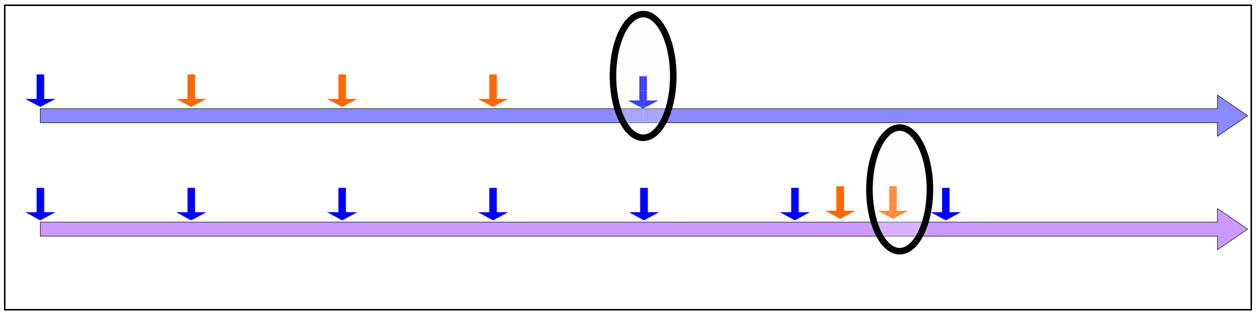

Figure 5. Recovery from application-coordinated Snapshot copy of the transaction logs volume.

In order to perform database recovery using the SnapMirror point of the database's data volume and an application-coordinated Snapshot copy of the transaction logs volume, you would need to complete the following steps:

i. Restore the transaction logs volume from an application-coordinated snapshot copy by executing the following statement on the IBM N series storage system:

```
snap restore -s <snapshot name> <volume name>
```

For example, to restore a volume named dblogs from an application-coordinated snapshot copy named 's2', you would execute the following command on the IBM N series storage system:

```
snap restore -s s2 dblogs
```

ii. After restoring the database transaction logs volume, mount the data and transaction logs volumes used for the database to a database host by executing the following command:

```
mount -o rw,bg,hard,nointr,rsize=32768,wsize=32768,tcp,vers=3,timeo=600
<IBMN storage system name>:<volume name> <mount point name>
```

For example, to mount a volume named dbdata on a mount point named /mnt/dbdata, you would execute the following command on the database server:

```
mount -o rw,bg,hard,nointr,rsize=32768,wsize=32768,tcp,vers=3,timeo=600
ntapdst:/vol/dbdata /mnt/dbdata
```

iii. The control file that resides on the transaction logs volume will have the latest database information, so copy that control file from the `dblogs` volume to the `dbdata` volume by executing the following commands on the database server:

```
cp /mnt/dblogs/orcl/control03.ctl
/home/oracle/ora10g/dbdata/orcl/control01.ctl
cp /mnt/dblogs/orcl/control03.ctl
/home/oracle/ora10g/dbdata/orcl/control02.ctl
```

iv. Start the database instance and perform database recovery by executing the following commands on the database server:

```
sqlplus / as sysdba
sql>startup mount
sql>recover database
sql>alter database open
```

After completing these steps, database recovery from an application-coordinated snapshot copy of the transaction logs volume and the SnapMirror point of the data volumes is complete.

d. Recovery from application-coordinated snapshot copies of the data and transaction logs volume

This case is variation of the previous option. The only difference is that the data volume for the database is also restored from an application-coordinated snapshot copy. The RPO is determined from the time that the snapshot copy of the transaction logs volume was created to the time the disaster started. The RTO will vary based on the transaction logs that need to be applied for the database recovery.

For example, in our DR scenario, the disaster started at 3:05 p.m. and the application-coordinated snapshot copy from which we want to restore the transaction logs volume was created at 2:50 p.m. Therefore, the RPO value will be15 minutes.

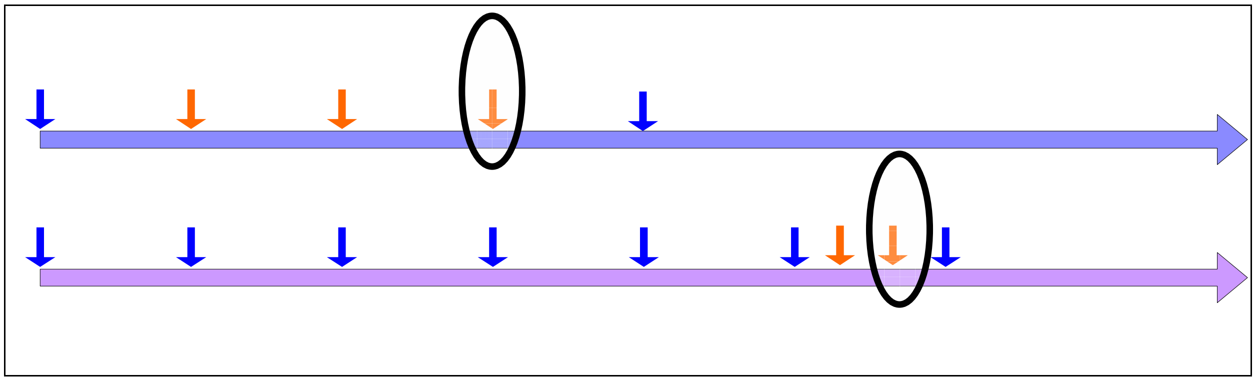

Figure 6. Recovery from the application-coordinated Snapshot copies of the dbdata and dblogs volumes.

In order to perform recovery from application-coordinated snapshot copies of database data and transaction logs volumes, you would need to complete the following steps:

i. Restore the database data and transaction logs volume from application-coordinated snapshot copies by running the following command on the IBM N series storage system:

```
snap restore -s <snapshot name> <volume name>
```

For example, to restore a volume named `dblogs` from an application-coordinated snapshot copy named `'s2'`, run the following command on the IBM N series storage system:

```
snap restore -s s2 dblogs
```

Restore all the data as well as the transaction logs volumes using the above command.

ii. After restoring the data and transaction logs volumes, you need to mount them to a database host by executing the following command:

```
mount -o rw,bg,hard,nointr,rsize=32768,wsize=32768,tcp,vers=3,timeo=600
<IBMN storage system name>:<volume name> <mount point name>
```

For example, to mount a volume named `dbdata` on mount point `/mnt/dbdata` , you would execute the following command on the database server:

```
mount -o rw,bg,hard,nointr,rsize=32768,wsize=32768,tcp,vers=3,timeo=600
ntapdst:/vol/dbdata /mnt/dbdata
```

iii. The control file that resides on the transaction logs volume will have the latest database information, so copy that control file from the `dblogs` volume to the `dbdata` volume by executing the following commands on the database server:

SM1

S1

```
cp /mnt/dblogs/orcl/control03.ctl
/home/oracle/ora10g/dbdata/orcl/control01.ctl
cp /mnt/dblogs/orcl/control03.ctl
/home/oracle/ora10g/dbdata/orcl/control02.ctl
```

iv. Start the database instance and perform database recovery by running these commands:

```
sqlplus / as sysdba
sql>startup mount
sql>recover database
sql>alter database open
```

After completing the above steps, the database recovery from an application-coordinated snapshot copy of the data as well as the transaction logs volume will be complete.

Database recovery using SnapMirror Async and Sync

The database recovery strategy may include a mix of SnapMirror Async, Sync, and Semi-Sync modes of data replication. To simulate a DR strategy using mixed modes of SnapMirror, we configured the data volume replication using SnapMirror Async and the transaction logs volume replication using SnapMirror Sync mode. The SnapMirror update and application-coordinated snapshot schedule has been summarized in following table.

Flexible Volume	SnapMirror	App. Coordinated Snapshot
Dbdata	**2 hrs**	**30 min**
Dblogs	**Sync**	**-**

Table 2. SnapMirror and snapshot update schedule (B).

To simulate a disaster at the primary site and perform recovery at the DR site that has replicated data using SnapMirror Async and Sync modes, complete the following steps.

For this configuration, the entries in the `/etc/snapmirror.conf` file look similar to the following:

```
ntapsrc:dbdata ntapdst:dbdata – 0 0-22/2 * 1-5
ntapsrc:dblogs ntapdst:dblogs – sync
```

The argument `‘sync’` in the above entries controls whether a volume is **sync snapmirrored** with its source or Async. If the argument is left out of the `conf` file, the volume will be **async snapmirrored**.

To simulate a disaster on the primary site and perform recovery on the DR site, you would need to complete the following steps:

4. Complete the SnapMirror base data transfer and run the load

   a. Initialize the SnapMirror relationship for each volume used for the database and complete one-time baseline transfer by executing the following command on the IBM N series storage system:

   ```
   snapmirror initialize –S <source IBMN storage system>:<volume name>
   <destination IBMN storage system>:<volume name>
   ```

   b. For example, to initialize the baseline transfer for the volume named dbdata, you would execute the following command on the IBM N series storage system:

   `snapmirror initialize –S ntapdst:dbdata ntapdst:dbdata`Monitor the SnapMirror status by running the following command on the destination storage system:

```
snapmirror status
```

Let us assume our baseline transfer started at 12.00 p.m. and completed at 12.25 p.m.

c. Connect to the database and run the load by executing the following commands on the database server:

```
connect scott/tiger
execute scott.demobld(25000000);
```

d. Copy the `do_snap`, `begin_bkup`, and `end_bkup` scripts to a work directory on the database server and start taking coordinated snapshot copies by executing the `do_snap` script on the database server.

```
./do_snap
```

The above command can be executed as a `cron` job from the database server.

5. Simulate the disaster

Let us assume disaster strikes at 3:05 p.m. In order to simulate the disaster, we made the SnapMirror source IBM N series storage system unavailable by executing the following command:

```
halt
```

At this point, the system state will look somewhat similar to the one shown in the Figure 7.

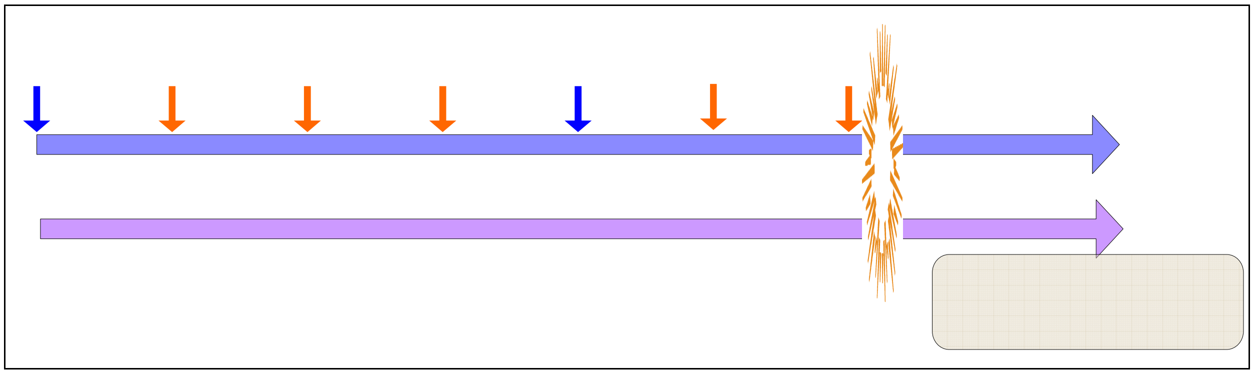

Figure 7. Database state at the time of disaster at the primary site.

After a disaster, the SnapMirror destination volumes corresponding to the database data will be in SnapMirror snapshot consistent state. That means the data volumes at the DR site will have the same data, including application-coordinated snapshot copies, as the corresponding source volume had at the time of successful SnapMirror update. The transaction logs volume is **sync snapmirrored**; therefore, the corresponding volume at the DR site will have all the changes up to the point the disaster started. For our test scenario, the data volume named `dbdata` is **async snapmirrored** and the transaction logs volume named `dblogs` is **sync snapmirrored**; therefore, these two volumes will be in the following states:

- `dbdata` – consistent with the source volume at the time of SnapMirror update at 2.00 p.m.
- `dblogs` – consistent with the source volume at the time disaster started at 3.05 p.m.

6. Recovery

After the baseline transfer, the destination volumes are available to clients, but in a read-only state. The status of a destination will show that it is **snapmirrored**. A disaster makes the source unavailable; to use the destination volumes for writing as well as reading, you would need to end the SnapMirror relationship by executing the following command on the destination IBM N series storage system:

```
snapmirror break <volume name>
```

For example, to make a volume named `dbdata` writable, you would execute the following command on the IBM N series storage system:

```
snapmirror break dbdata
```

This command changes the destination volume's status from **snapmirrored** to **broken-off**, thus making it writable.

After a disaster, in this type of configuration, there are two possible options for database recovery:

- Recovery from the SnapMirror update point of the `data` volumes.
- Recovery from an application-coordinated snapshot copy of the data volumes.

a. Recovery from the SnapMirror update point of data volumes

The database's transaction logs volume is **sync snapmirrored**; as a result, all the changed data up to the point the disaster started is replicated to the destination volume. Therefore, after database recovery, there shouldn't be any data loss; the RPO (recovery point objective) will be zero. The RTO (recovery time objective) will vary based on the transaction logs that need to be applied for recovery.

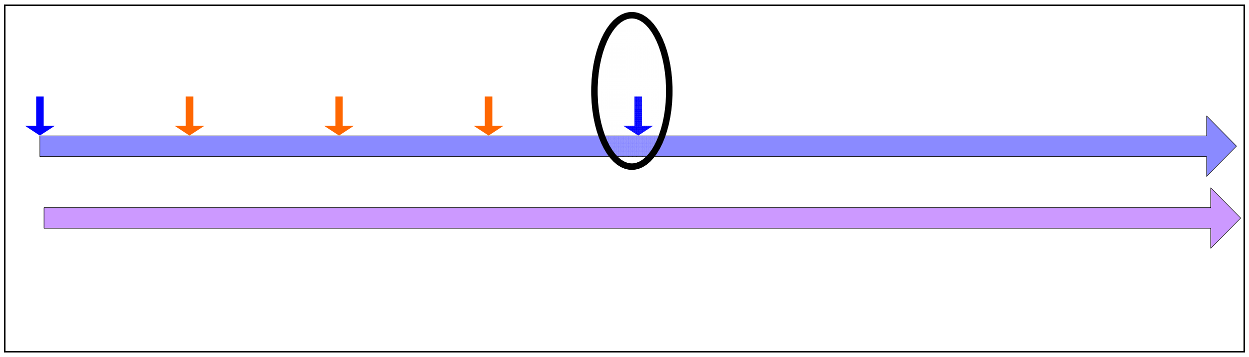
Figure 8. Recovery from the SnapMirror point of the dbdata volume.

To perform recovery from the SnapMirror update point of the volumes used for holding the database's data, you would complete the following steps on the database server:

i. You need to mount each volume used for the database to a database host by running the following command:

```
mount -o rw,bg,hard,nointr,rsize=32768,wsize=32768,tcp,vers=3,timeo=600
<IBMN storage system name>:<volume name> <mount point name>
```

For example, to mount a volume named `dbdata` that resides on an IBM N series storage system named `ntapdst` and that needs to be mounted on a mount point named `/mnt/dbdata`, you would execute the following command on the database server:

```
mount -o rw,bg,hard,nointr,rsize=32768,wsize=32768,tcp,vers=3,timeo=600
ntapdst:/vol/dbdata /mnt/dbdata
```

ii. The control file that resides on the transaction logs volume will have the latest database information, so copy that control file from the `dblogs` volume to the `dbdata` volume by executing the following commands on the database server:

```
cp /mnt/dblogs/orcl/control03.ctl
/home/oracle/ora10g/dbdata/orcl/control01.ctl
cp /mnt/dblogs/orcl/control03.ctl
/home/oracle/ora10g/dbdata/orcl/control02.ctl
```

iii. Start the database instance and perform database recovery by executing the following commands on the database server:

```
sqlplus / as sysdba
sql>startup mount
sql>recover database
sql>alter database open
```

After completing the above steps, the database recovery will be complete.

b. Recovery from an application-coordinated snapshot of the dbdata volume

This type of recovery may become necessary in rare cases, such as when database recovery is not possible from the SnapMirror update point. In this kind of situation, the database can be recovered using application-coordinated snapshot copies of the data volume.

In this configuration, the database transaction logs volume is **sync snapmirrored**; as a result, all the changed data up to the point the disaster started is replicated to the destination volume. Therefore, after database recovery, there shouldn't be any data loss; RPO (recovery point objective) will be zero. The RTO (recovery time objective) will vary based on the transaction logs that need to be applied for recovery.

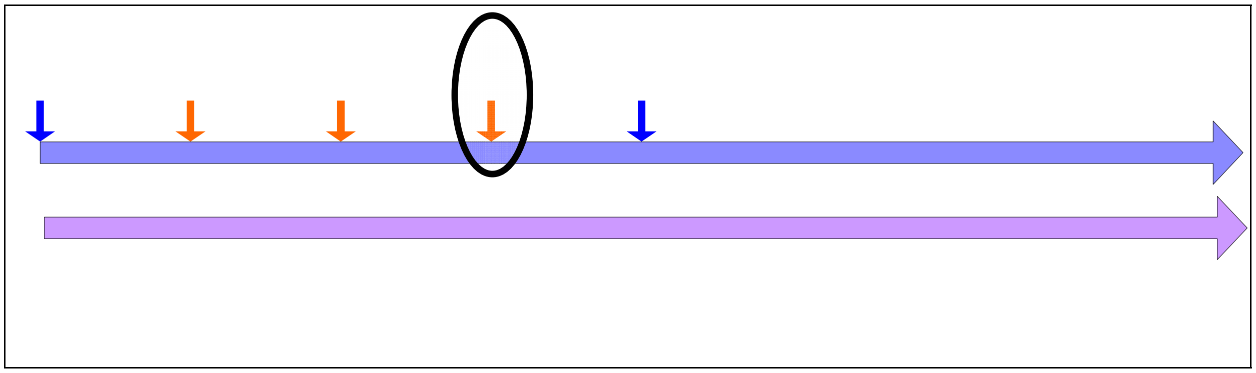

Figure 9) Recovery from an application-coordinated snapshot copy of the dbdata volume.

To perform database recovery using an application-coordinated snapshot copy of the volume holding the database's data, you would need to complete the following steps:

i. First you need to restore the volumes that hold data for the database from the application-coordinated snapshot copy by executing the following command on the IBM N series storage system:

```
snap restore -s <snapshot name> <volume name>
```

For example, to restore a volume named `dbdata` from an application-coordinated snapshot copy named `s3`, you would execute the following command on the IBM N series storage system:

```
snap restore -s s3 dbdata
```

ii. Mount the IBM N series storage system volumes that hold the database's data and transaction logs to a database host by executing the following command:

```
mount -o rw,bg,hard,nointr,rsize=32768,wsize=32768, tcp,vers=3,timeo=600
<IBMN storage system name>:<volume name> <mount point>
```

For example, to mount a volume named `dbdata` that resides on an IBM N series storage system named `ntapdst` and that needs to be mounted on a mount point named `/mnt/dbdata`, you would execute the following command on the database server:

```
mount -o rw,bg,hard,nointr,rsize=32768,wsize=32768,tcp,vers=3,timeo=600
ntapdst:/vol/dbdata /mnt/dbdata
```

iii. The control file that resides on the transaction logs volume will have the latest database information, so copy that control file from the `dblogs` volume to the `dbdata` volume by executing the following commands on the database server:

```
cp /mnt/dblogs/orcl/control03.ctl
/home/oracle/ora10g/dbdata/orcl/control01.ctl
cp /mnt/dblogs/orcl/control03.ctl
/home/oracle/ora10g/dbdata/orcl/control02.ctl
```

iv. Start the database instance and perform database recovery by executing the following commands on the database server:

```
sqlplus / as sysdba
sql>startup mount
sql>recover database
sql>alter database open
```

After completing the above steps, the database recovery will be complete.

Conclusion

IBM N series with SnapMirror is a proven data replication technology that offers great return on investment. It provides flexible and robust methods to keep the recovery objectives controlled. The SnapMirror technology can be easily integrated with other IBM N series technologies, such as IBM System Storage N series with Metro Cluster and IBM System Storage N series with SyncMirror®, to further improve recovery objectives and ensure high availability.

Appendix A: Scripts

Stored procedure script

```
CREATE OR REPLACE PROCEDURE demobld( rcount IN number DEFAULT 10) IS
--
-- Purpose: Generate database load by inserint specified number of rows in the
--          table
--
-- MODIFICATION HISTORY
-- Person      Date    Comments
-- ---------   ------  -------------------------------------------

BEGIN
   for i in 1..rcount  loop
      insert into scott.tab1 values (i,
'ABDFADFAFAFAFASFDASDFFFFFFFFFFFFFFFFFFFFFFFFFFF');
      if mod(i,1000)=0 then
         commit;
      end if;
   end loop;
END; -- Procedure
/
```

Application-coordinated snapshot script (do_snap)

```
--------------------------------------------------------------------------------
--  Script Name: do_snap
--      Purpose: This script connects to the database and takes application –
--              Coordinated Snapshots
--        Usage:do_snap <IBMN storage system name> <vol name> <Snapshot name>
--
--------------------------------------------------------------------------------
if [ $# = 0 ]   then
   echo " USAGE: do_snap <IBMN storage system name> <vol name>  <snapshot name>"
   exit 1
fi
if [ $# = 1 ]  then
    echo "USAGE: do_snap <IBMN storage system name> <vol name> <snapshot name>"
    exit 1
fi
if [ $# = 2 ]  then
    echo " USAGE: do_snap <IBMN storage system name> <vol name> <snapshot name>"
    exit 1
fi
rsh $1 snap delete $2 $3
#------------------------------------------------------------------------------
# Put the database in hot backup mode
#------------------------------------------------------------------------------
sqlplus system/oracle @begin_bkup.sql
#-------------------------------------------------------------------------------
# Create snapshot
#-------------------------------------------------------------------------------
rsh $1 snap create $2 $3
#-------------------------------------------------------------------------------
# Take the Tablespaces/datafiles out of hot backup mode
#-------------------------------------------------------------------------------
sqlplus system/oracle @end_bkup.sql
rsh $1 snap list $2
echo ""
```

Application-Coordinated snapshot script (begin_bkup)

```
-------------------------------------------------------------------------
-- Script Name: begin_bkup
-- Script Purpose: This script put the database in hot backup mode
-------------------------------------------------------------------------
alter database begin backup;
EXIT;
```

Application-Coordinated snapshot script (end_bkup)

```
-------------------------------------------------------------------------
-- Script Name: end_bkup
-- Script Purpose: This script bring the database out of hot backup mode
-------------------------------------------------------------------------
alter database end backup;
EXIT;
```

Trademarks and special notices