Redbooks Paper

Alex Osuna
Patrick Március Médice Bisi
Sven Schaffranneck

N series Snapshot: A Technical Discussion

Introduction and overview

This IBM® Redpaper discusses the N series Snapshot™ feature. Snapshot is a standard feature of the Data ONTAP® operating system. It enables online backups to be maintained, thus providing near-instantaneous access to previous versions of data without requiring complete, separate copies or resorting to offline backups. Snapshots can be scheduled by an administrator, and you can keep up to 255 Snapshots online at any one time. The paper provides UNIX® and Windows® examples of creating Snapshots.

Snapshot technology makes extremely efficient use of storage by storing only block-level changes between each successive Snapshot. In this way, it can be simply thought of as being similar to any modern source code control system that maintains only the changes made to the original source code in order to minimize space and maximize recoverability. This analogy actually extends further, and we explore it later in this document.

Because the Snapshot process is automatic and virtually instantaneous, backups are significantly faster and simpler. Snapshots can also be coordinated with outside applications to ensure highly consistent data states (as viewed from the application) prior to performing Snapshot and other backup procedures. For example, flushing data from a production database prior to Snapshot creation is a generally recognized best practice.

The primary focus of this paper is on the algorithms and data structures that Write Anywhere File Layout (WAFL®) uses to implement Snapshots, which are read-only clones of the active file system. WAFL uses a technique to minimize the disk space that Snapshots consume.

This paper also describes how WAFL uses Snapshots to eliminate the need for file system consistency checking after an unclean shutdown, and gives examples of how to set up a Snapshot using the FilerView®, Data ONTAP CLI, SnapDrive® for Microsoft® Windows and SnapDrive for Linux and UNIX.

The WAFL primary distinguishing characteristic is Snapshots, which are read-only copies of the entire file system. WAFL creates and deletes Snapshots automatically at prescheduled times, and it keeps up to 255 Snapshots online at once to provide easy access to old versions of files.

Figure 1 on page 3 gives an overview of Snapshot features.

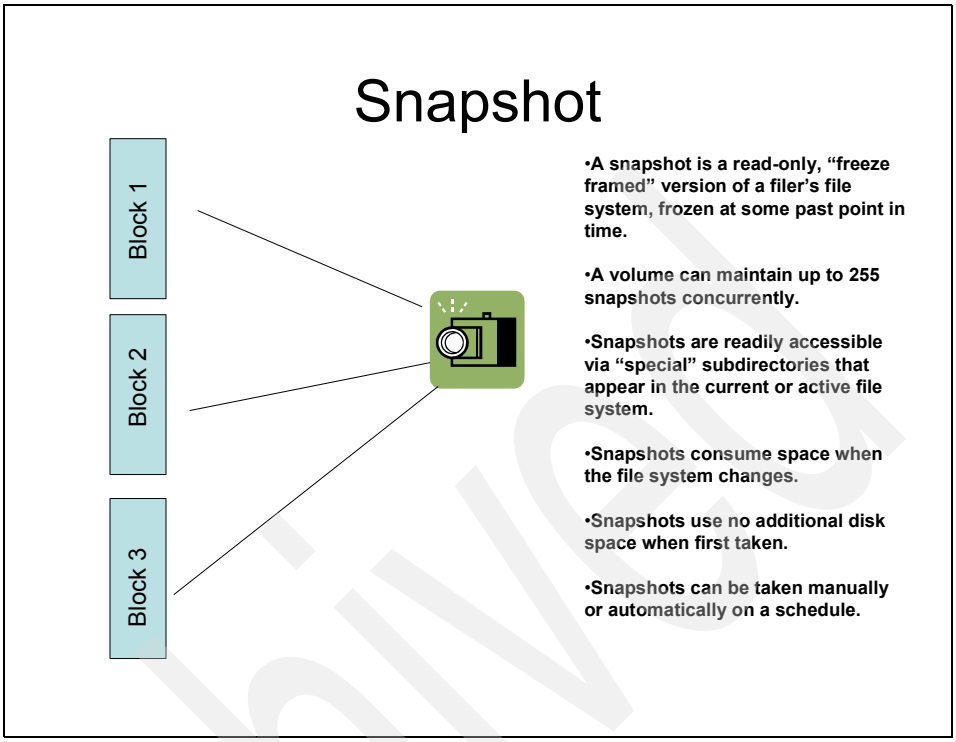

Figure 1 Snapshot features

Snapshots use a technique to avoid duplicating disk blocks that are the same in a Snapshot as in the active file system. Only when blocks in the active file system are modified or removed do Snapshots containing those blocks begin to consume disk space.

Users can access Snapshots through NFS to recover files that they have accidentally changed or removed, and system administrators can use Snapshots to create backups safely from a running system. In addition, WAFL uses Snapshots internally so that it can restart quickly even after an unclean system shutdown.

High level Snapshot process

In this section we provide an overview of the Snapshot process:

1. Snapshots are taken from active data on the file system, as shown in Figure 2 on page 4.

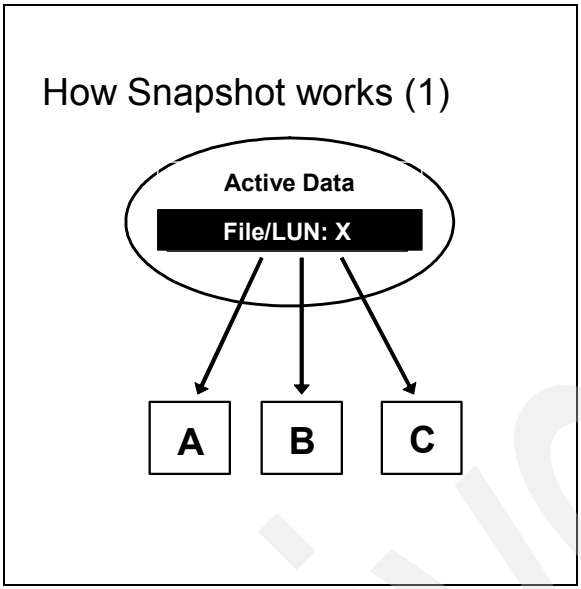

Figure 2 Step 1 - Initial Snapshot is taken on active data

2. When an initial Snapshot is taken, no initial data is copied. Instead, pointers are created to the original blocks for recording the point-in-time state of these blocks, as shown in Figure 3 on page 5.

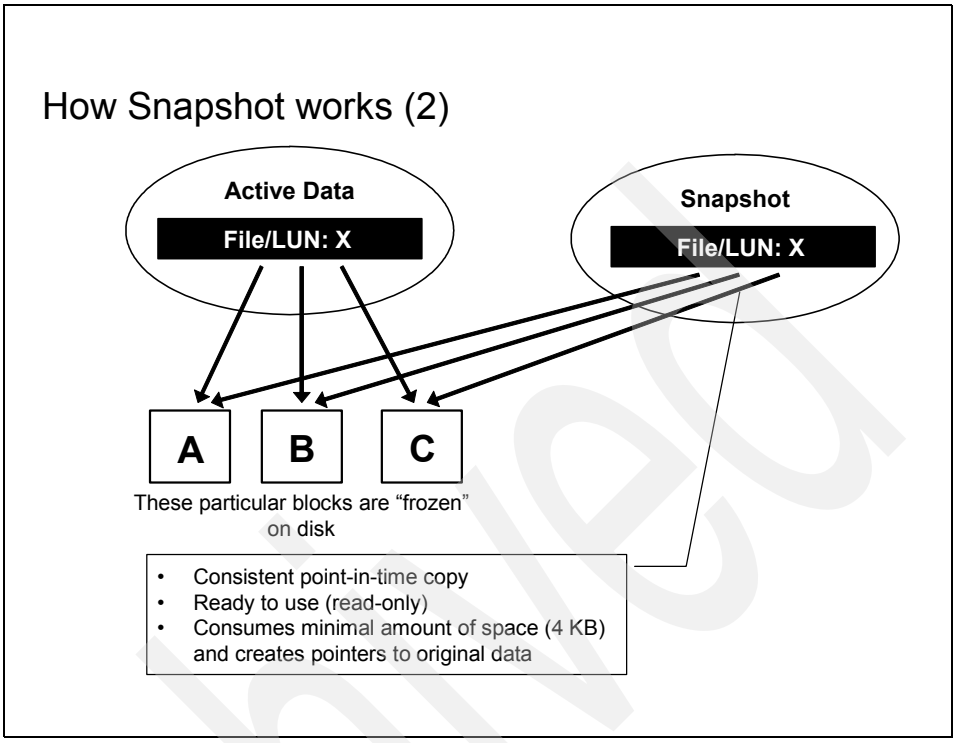

Figure 3 Step 2 - Pointers are created

3. When a request to Block C occurs, the original Block C1 is frozen in order to maintain a point-in-time copy. The modified block C2 is written to another location on disk and becomes the active block, as shown in Figure 4 on page 6.

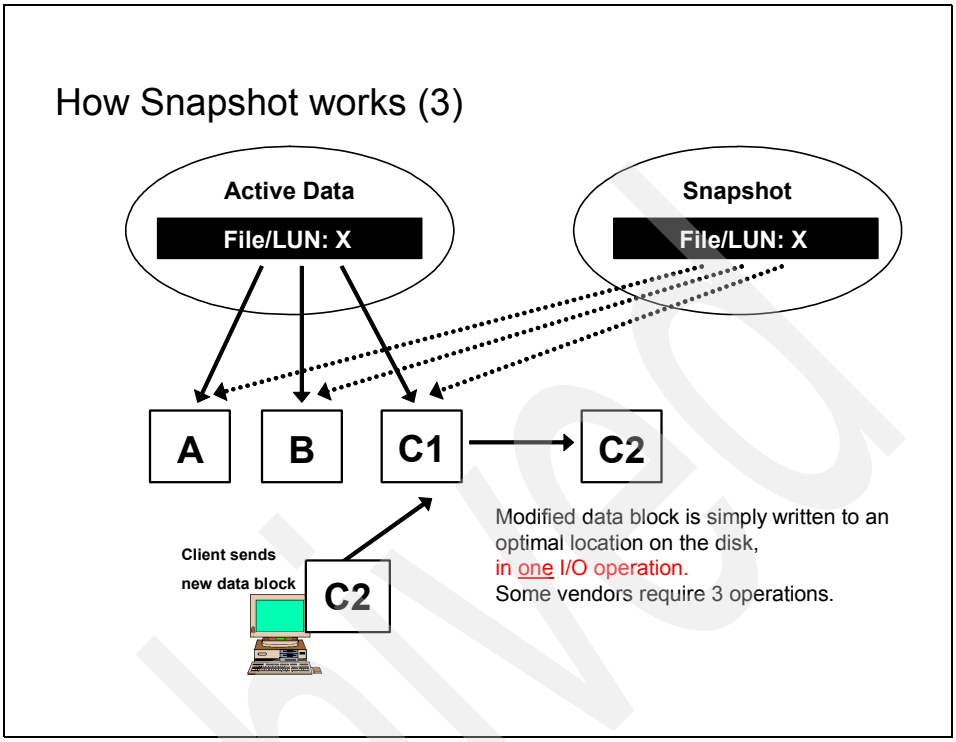

Figure 4 Step 3 - Modified data block written to disk in one operation

4. The final result is that the Snapshot now consumes 4 K + C1 of space. Active pointers for the point-in-time snapshot are unmodified blocks A, B, and point-in-time copy C1, as shown in Figure 5 on page 7.

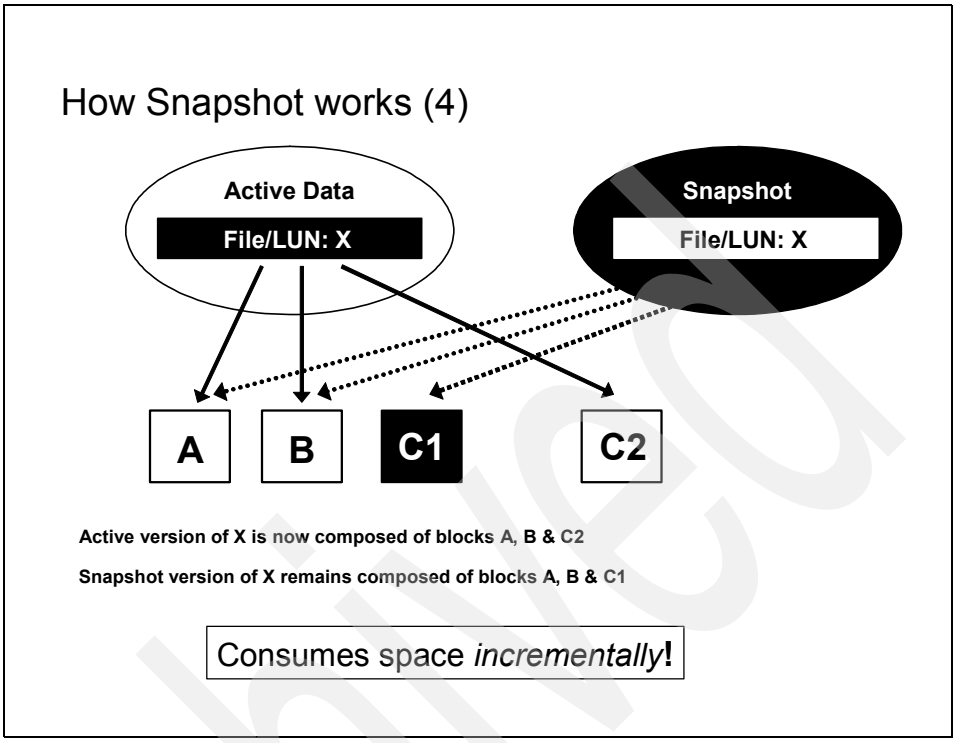

Figure 5 Step 4 - Active pointers

Understanding Snapshots in detail

A small percentage of the drive's available space is used to store file system-related data, and can be considered as overhead. A file system splits the remaining space into small, consistently-sized segments. In the UNIX world, these segments are known as *inodes*.

Understanding that the WAFL file system is a “tree of blocks” rooted by the root inode is the key to understanding Snapshots. To create a virtual copy of this tree of blocks, WAFL simply duplicates the root inode. Figure 6 on page 8 illustrates how this works. WAFL creates a Snapshot by duplicating the root inode that describes the inode file, and avoids changing blocks in the Snapshot by writing new data to new locations on disk.

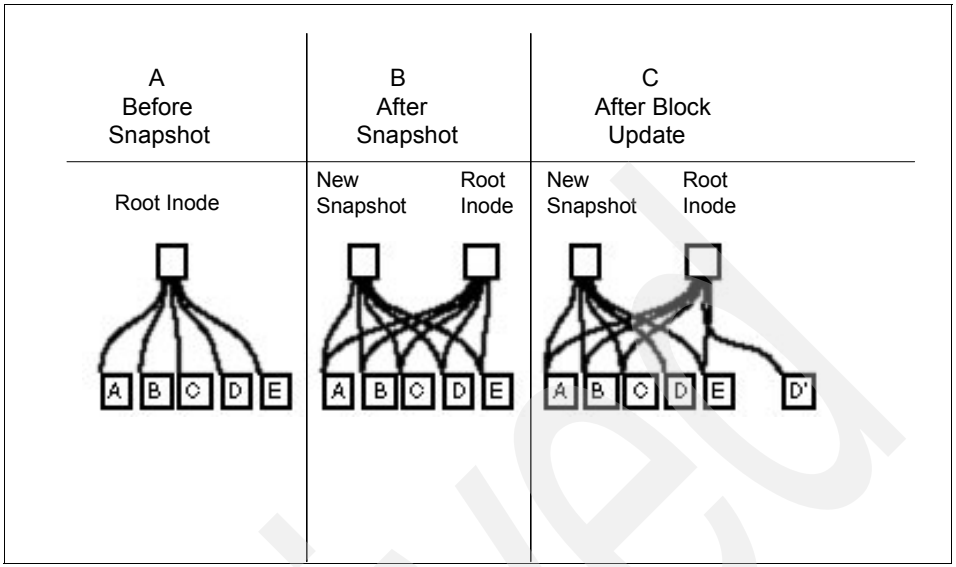

Figure 6 WAFL duplicates root inode - avoids changing blocks by writing new data to new locations on disk

Column B in Figure 6 shows WAFL creating a new Snapshot by making a duplicate copy of the root inode. This duplicate inode becomes the root of a tree of blocks representing the Snapshot, just as the root inode represents the active file system. When the Snapshot inode is created, it points to exactly the same disk blocks as the root inode, so a brand-new Snapshot consumes no disk space except for the Snapshot inode itself.

Column C in Figure 6 shows what happens when a user modifies data block D. WAFL writes the new data to block D' on disk, and changes the active file system to point to the new block. The Snapshot still references the original block D, which is unmodified on disk.Over time, as files in the active file system are modified or deleted, the Snapshot references more and more blocks that are no longer used in the active file system. The rate at which files change determines how long Snapshots can be kept online before they consume an unacceptable amount of disk space.

WAFL's Snapshots duplicate the root inode instead of copying the entire inode file. This reduces considerable disk I/O and saves significant disk space.

By duplicating just the root inode, WAFL creates Snapshots very quickly and with very little disk I/O. Snapshot performance is important because WAFL creates a Snapshot every few seconds to allow quick recovery after unclean system shutdowns.

Figure 7 shows the transition from column A to column B in Figure 6 on page 8 in more detail. When a disk block is modified, and its contents are written to a new location, the block's parent must be modified to reflect the new location. The parent's parent, in turn, must also be written to a new location, and so on up to the root of the tree.

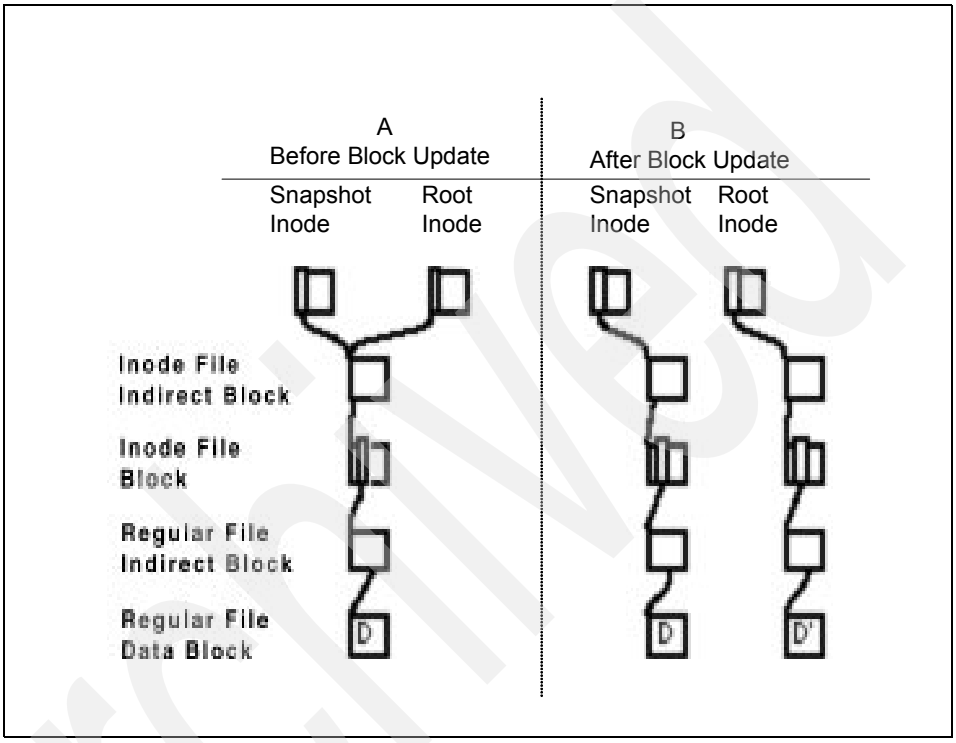

Figure 7 Updating the pointers in the block's ancestors, requiring them to be written to new locations

WAFL would be very inefficient if it wrote this many blocks for each NFS write request. Instead, WAFL gathers up many hundreds of NFS requests before scheduling a write episode. During a write episode, WAFL allocates disk space for all the corrupted data in the cache and schedules the required disk I/O. As a result, commonly modified blocks, such as indirect blocks and blocks in the inode file, are written only once per write episode instead of once per NFS request.

Snapshot data structures and algorithms

The Snapshot data structures and algorithms are unique to the N series storage systems, and they are built upon WAFL and its design. Snapshot utilizes the WAFL design characteristics so that operational overhead is kept to a minimum.

The block-map file

Most file systems keep track of free blocks by using a bitmap with one bit per disk block. If the bit is set, then the block is in use. However, this technique does not work for WAFL, because many Snapshots can reference a block at the same time.

Instead, the WAFL block-map file contains a 32-bit entry for each 4 KB disk block. Bit 0 is set if the active file system references the block, bit 1 is set if the first Snapshot references the block, and so on. A block is in use if any of the bits in its block-map entry are set.

Figure 8 on page 11 shows the life cycle of a typical block-map entry. At time t1, the block-map entry is completely clear, indicating that the block is available. At time t2, WAFL allocates the block and stores file data in it.

When Snapshots are created, at times t3 and t4, WAFL copies the active file system bit into the bit indicating membership in the Snapshot. The block is deleted from the active file system at time t5. This can occur either because the file containing the block is removed, or because the contents of the block are updated and the new contents are written to a new location on disk.

The block cannot be reused, however, until no Snapshot references it. In Figure 8 on page 11, this occurs at time t8 after both Snapshots that reference the block have been removed.

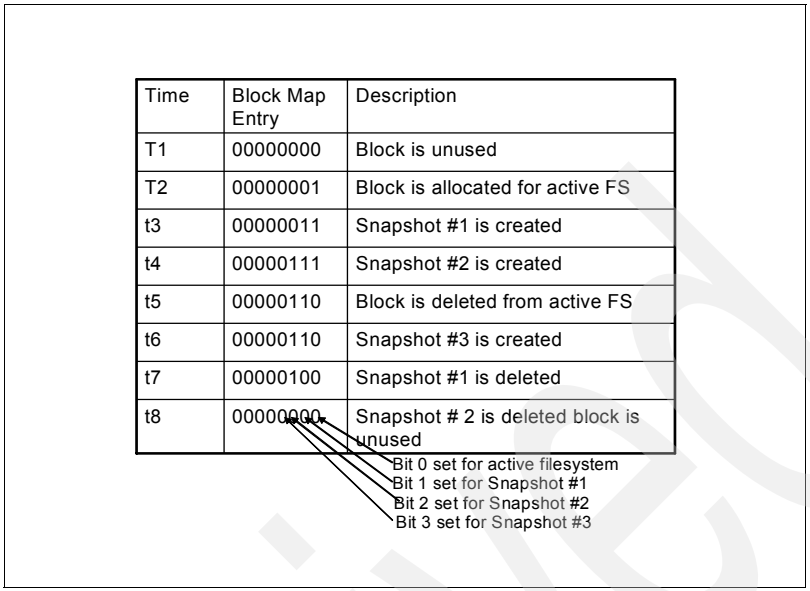

Figure 8 The life cycle of a block-map file entry

Creating a Snapshot

The challenge in writing a Snapshot to disk is to avoid locking out incoming NFS requests. The problem is that new NFS requests may need to change cached data that is part of the Snapshot that must remain unchanged until it reaches disk.

An easy way to create a Snapshot would be to suspend NFS processing, write the Snapshot, and then resume NFS processing. However, writing a Snapshot can take more than a second, which is too long for an NFS server to stop responding. (Remember that WAFL creates a consistency point Snapshot at least every 10 seconds, so performance is critical.)

The WAFL technique for keeping Snapshot data self-consistent is to mark all the corrupted data in the cache as IN_SNAPSHOT. The rule during Snapshot creation is that data marked IN_SNAPSHOT must not be modified, and data not marked IN_SNAPSHOT must not be flushed to disk. NFS requests can read all file system data, and they can modify data that is not IN_SNAPSHOT, but processing for requests that need to modify IN_SNAPSHOT data must be deferred.

To avoid locking out NFS requests, WAFL must flush IN_SNAPSHOT data as quickly as possible. To do this, WAFL performs the following steps:

1. It allocates disk space for all files with IN_SNAPSHOT blocks. WAFL caches inode data in two places: in a special cache of in-core inodes, and in disk buffers belonging to the inode file. When it finishes write allocating a file, WAFL copies the newly updated inode information from the inode cache into the appropriate inode file disk buffer, and clears the IN_SNAPSHOT bit on the in-core inode.

   When this step is complete, no inodes for regular files are marked IN_SNAPSHOT, and most NFS operations can continue without blocking. Fortunately, this step can be done very quickly because it requires no disk I/O.

2. It updates the block-map file. For each block-map entry, WAFL copies the bit for the active file system to the bit for the new Snapshot.

3. It writes all IN_SNAPSHOT disk buffers in cache to their newly-allocated locations on disk. As soon as a particular buffer is flushed, WAFL restarts any NFS requests waiting to modify it.

4. It duplicates the root inode to create an inode that represents the new Snapshot, and turn the root inode's IN_SNAPSHOT bit off. The new Snapshot inode must not reach disk until after all other blocks in the Snapshot have been written.

   If this rule were not followed, an unexpected system shutdown could leave the Snapshot in an inconsistent state.

After the new Snapshot inode has been written, no more IN_SNAPSHOT data exists in cache, and any NFS requests that are still suspended can be processed. Under normal loads, WAFL performs these four steps in less than a second. Step 1 can generally be done in just a few hundredths of a second, and after WAFL completes it, very few NFS operations need to be delayed.

Deleting a Snapshot

Deleting a Snapshot is a trivial task. WAFL simply zeros the root inode representing the Snapshot and clears the bit representing the Snapshot in each block-map entry.

When creating Snapshots from LUNs, the task can be accomplished by using SnapDrive software from the host and running the command from the Data ONTAP Command Line Interface (CLI), or running the command from the FilerView.

SnapDrive can be installed on Microsoft Windows servers and on UNIX servers (such as IBM-AIX, HP-UX, Red Hat Linux and Solaris™).

Creating a Snapshot from the FilerView management interface

FilerView is the Web management interface for the N series storage system. To access FilerView, using a Web browser go the URL:

```
http://Hostname/na_admin/
```

Follow these steps to create a Snapshot using the FilerView:

1. On the FilerView window, expand Volumes, then expand Snapshots. Click **Manage** to view the list of Snapshots that exist on the Filer (see Figure 9).

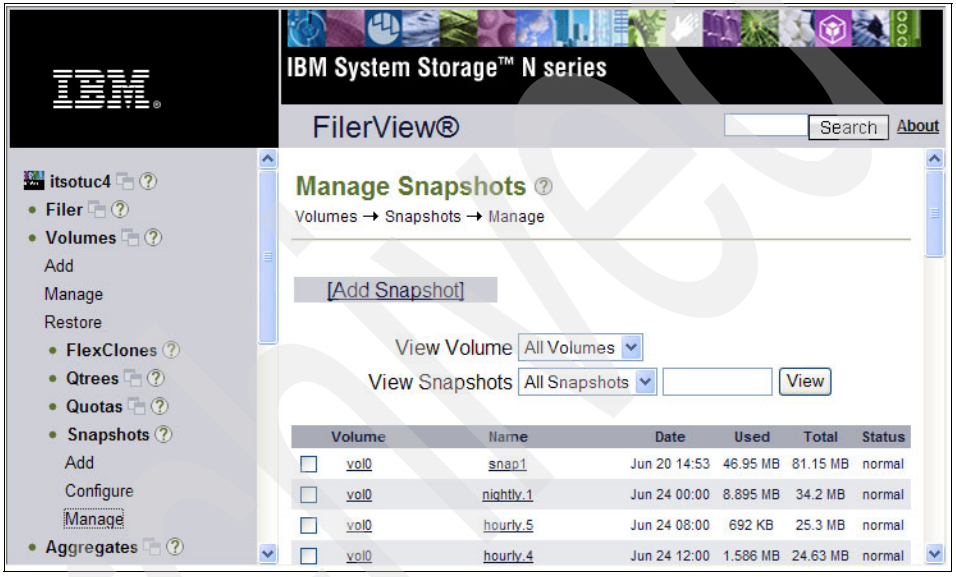

Figure 9 List of Snapshots

2. At the FilerView window, click **Add** to add a new Snapshot, as illustrated in Figure 10 on page 14. Select the Volume that you need to take the Snapshot, provide a name for the Snapshot file, and click **Add**.

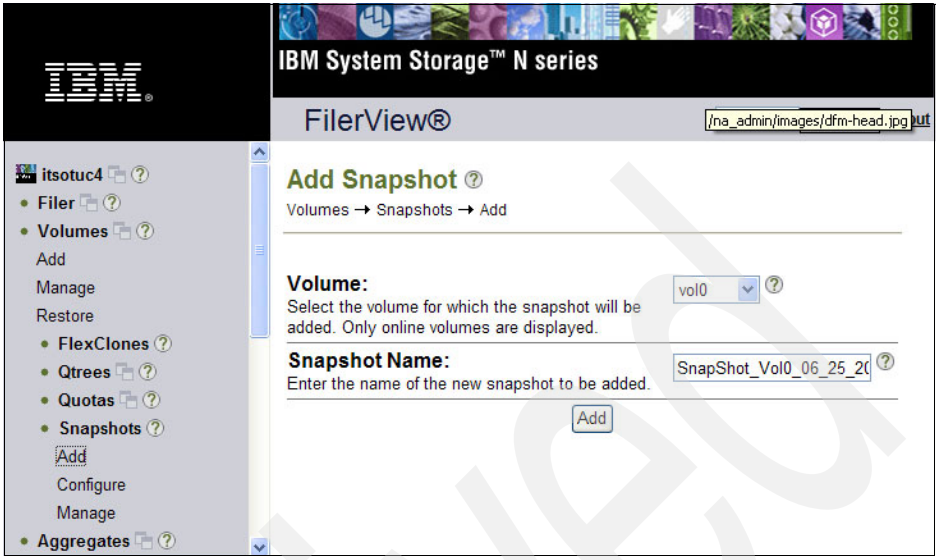

Figure 10 Snapshot creation

3. A Success message is shown, indicating that the Snapshot was successfully created (see Figure 11).

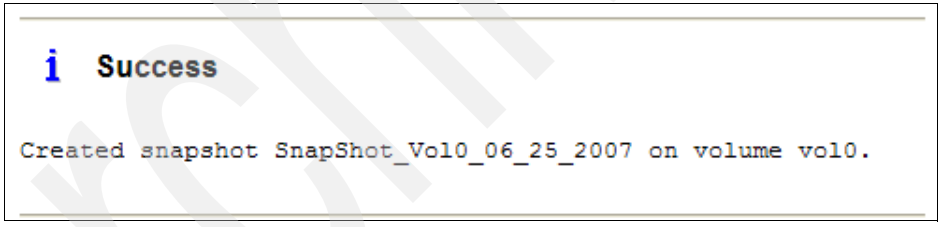

Figure 11 Success creation message

Creating a Snapshot from SnapDrive for UNIX and Linux

One of the better data protection solutions provided by IBM System Storage N series is the combination of Snapshot and SnapDrive. This section explains how these two features work together to create a high availability solution, allowing you to create, modify, delete or connect a Snapshot with just one command. SnapDrive for UNIX includes AIX® support.

You use the **`snapdrive snap create`** command to create Snapshot copies. As previously mentioned, they are point-in-time, read-only images of data on storage system volumes. The snap create operation ensures that you have backed up your LUNs files and directory trees. You can use the Snapshot copy you create to restore your data if you encounter corruption or other problems.

Note: To ensure that a Snapshot copy is "application-consistent," you usually need to stop (or perform whatever steps are required to quiesce) the application before taking the Snapshot copy.

Follow these guidelines when you enter commands that create Snapshot copies:

- You can keep a maximum of 255 Snapshot copies per storage system volume. This limit is set by the storage system. The total number can vary depending on whether other tools use these Snapshot copies.

  When the number of Snapshot copies has reached the maximum limit, the **snapshot create** operation fails. You must delete some of the old Snapshot copies before you can use SnapDrive for Linux and UNIX to take any more.
- SnapDrive for UNIX does not support Snapshot copies that it does not create. For example, it does not support Snapshot copies that are created from the storage system console, because such a practice can lead to inconsistencies within the file system.
- You cannot use SnapDrive for UNIX to create Snapshot copies of the following:
  - Root disk groups. The snap create operation fails when you try to take a Snapshot copy of a root disk group for a logical volume manager.
  - Boot device or swap device. SnapDrive for UNIX does not take a Snapshot copy of a system boot device or a system swap device.

Note: For more information about requirements, hints and commands, we highly recommend referring to *IBM System Storage N series SnapDrive for UNIX Installation and Administration Guide*, which is available at the following site:

`http://www-1.ibm.com/support/docview.wss?uid=ssg1S7001600&aid=1`

To create a Snapshot copy from the Linux or UNIX server using SnapDrive, follow these steps:

1. Create the Snapshot with the **`snapdrive snap create`** command; see Example 1.

Example 1 Creating the Snapshot of the Lotus® Domino® database and transactional log

```
[root@domino1 /]# snapdrive snap create -lun db_SdDg log_SdDg -snapname snap1
Successfully created snapshot snap1 on 2 filer volumes:
        itsotuc3:/vol/vol_DominoDB
        itsotuc3:/vol/vol_DominoLog

        snapshot snap1 contains:
        disk group db_SdDg containing host volumes
                db_SdHv (filesystem: /notesdata/db)
        disk group log_SdDg containing host volumes
                log_SdHv (filesystem: /notesdata/log)
[root@domino1 /]#
```

Where:

`-snapname` This is the short name of the Snapshot (in our case, it is `snap1`).

Note: Unless you specify otherwise, SnapDrive assumes that all entities that you specify on a given **`snap create`** command line are related; that is, the validity of updates to one entity may depend on updates to the other entities specified.

When storage entities have "dependent writes" in this way, SnapDrive takes steps to create a Snapshot copy that is crash-consistent for all storage entities as a group.

Example 2 illustrates the creation of a Snapshot from a normal LUN without using the LVM feature of Linux or UNIX.

Example 2 Creating a Snapshot from a single LUN

```
[root@domino1 ~]# snapdrive snap create -lun itsotuc3:/vol/vol_DominoDB/testlun
-snapname snapshot_vol0_06_25_2007
Successfully created snapshot snapshot_vol0_06_25_2007 on
itsotuc3:/vol/vol_DominoDB

        snapshot snapshot_vol0_06_25_2007 contains:
        raw LUN: itsotuc3:/vol/vol_DominoDB/testlun
```

```
[root@domino1 ~]#
```

Where:

`lun`	This switch follows the long name of a LUN from which the Snapshot should be created.
`snapname`	This is the short name of the Snapshot (in our case, it is `snapshot_vol0_06_25_2007`).

2. Validate the created Snapshot. Use the **`snapdrive snap list`** command for a list of all Snapshots on the specified file system or volume (see Example 3).

Example 3 List all Snapshots

```
[root@domino1 /]# snapdrive snap list -vg db_SdDg log_SdDg

snap name host date snapped
-------------------------------------------------------------------------
itsotuc3:/vol/vol_DominoDB:snap1  domino1 Jun  8 16:24 db_SdDg log_SdDg
itsotuc3:/vol/vol_DominoLog:snap1 domino1 Jun  8 16:24 db_SdDg log_SdDg
[root@domino1 /]#
```

Where:

-vg	This option means list all Snapshots of the specified volume groups.

Note: As shown in Example 3, the Snapshot group is displayed at the end of every row. At the time when the Snapshot is created, SnapDrive stops all I/O operations on `db_SdDg` and `log_SdDg`.

Creating a Snapshot from SnapDrive for Microsoft Windows

SnapDrive for Microsoft Windows is an optional feature provided by N series for Microsoft Windows servers that can improve the management of storage resources from the host operating system. After it is installed, SnapDrive eases the creation and management of Snapshots from the host.

To create a Snapshot from SnapDrive for Microsoft Windows, follow these steps:

1. Open Computer Management; SnapDrive will be available from the MMC.
2. Expand SnapDrive, expand Disks and expand the LUN that you wish to create a Snapshot of. Note that despite the fact that you are selecting a LUN, the actual Snapshot is volume-based.
3. Right-click **Snapshot** and select **Create Snapshot** (see Figure 12).

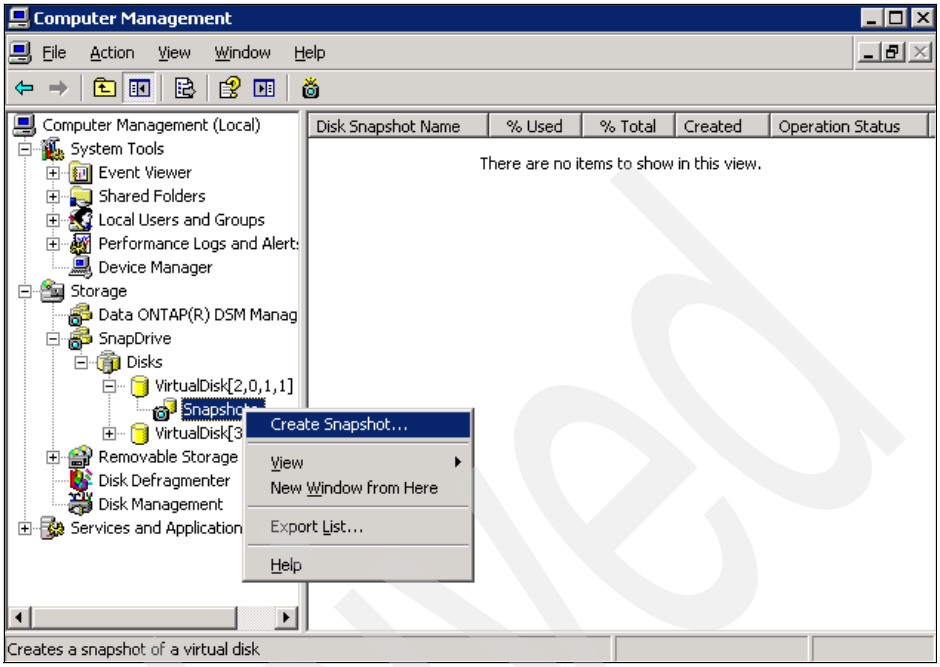

Figure 12 Computer Management with SnapDrive MMC

4. Type in a name for the Snapshot, then click **OK** (see Figure 13).

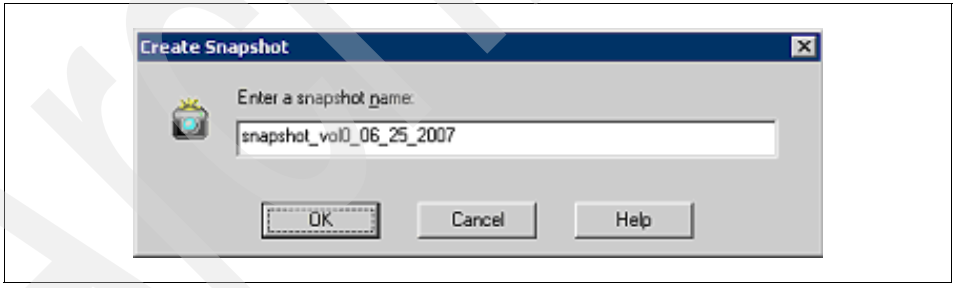

Figure 13 Snapshot name

5. The Snapshot will be created and the information will be shown on the SnapDrive MMC detail window (see Figure 14 on page 19).

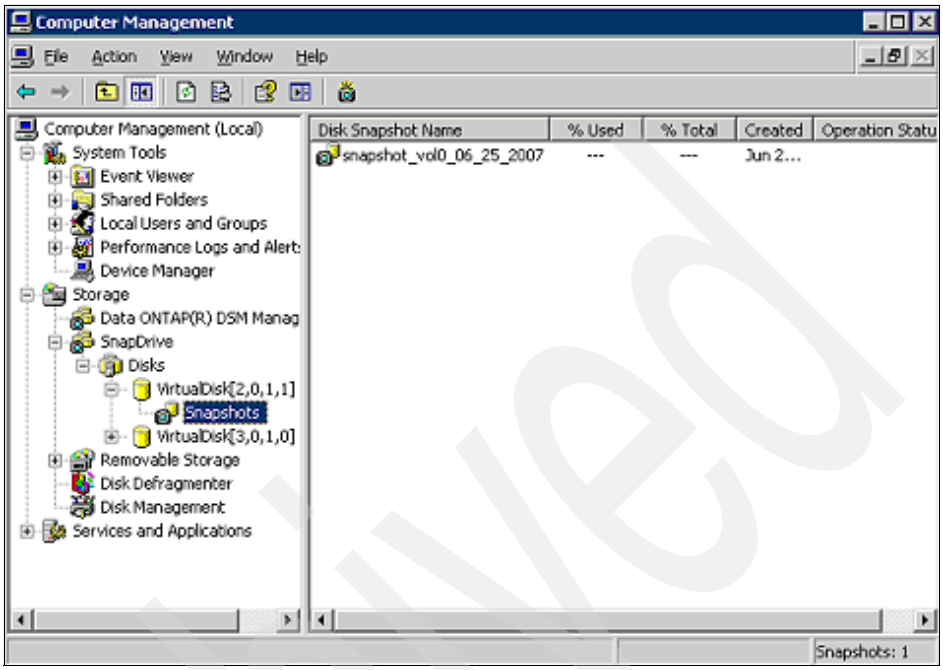

Figure 14 Snapshot created

Creating a Snapshot from Data ONTAP CLI

Another way to create Snapshots from a volume is from the Data ONTAP command line interface, using the **`snap create`** command.

Note: To view brief help information about the command, type **`snap help create`** at the command line interface:

```
itsotuc3> snap help create
snap create [-A | -V] <vol-name> <snapshot-name>
itsotuc3>
```

1. Logon to the N series Data ONTAP command line interface using Telnet, SSH or the serial console.
2. Use the **`snap create`** command to create a Snapshot of your target volume (see Example 4 on page 20).

Example 4 Snapshot creation on the Data ONTAP CLI

```
itsotuc3> snap create vol_DominoDB snapshot_vol0_06_25_2007
itsotuc3>
```

Where:

`vol_DominoDB` This is our volume name.

`snapshot_vol0_06_25_2007` This is the name of the Snapshot.

3. Verify the created Snapshot, using the **`snap list`** command (see Example 5). This step is not mandatory, but it is useful.

> **Note:** If you enter **`snap list`** without any options, all existing Snapshots are returned.

Example 5 List the newly created Snapshot

```
itsotuc3> snap list vol_DominoDB
Volume vol_DominoDB
working...

  %/used       %/total  date          name
----------  ----------  ------------  --------
0% ( 0%)    0% ( 0%)  Jun 26 15:44  snapshot_vol0_06_25_2007
itsotuc3>
```

Where:

`vol_DominoDB` This is the name of the volume on which we created the Snapshot.

The team that wrote this Redpaper

This Redpaper was produced by specialists working at the International Technical Support Organization, San Jose Center.

Alex Osuna is a Project Leader at the International Technical Support Organization, San Jose Center. He has more than 28 years of experience in the IT industry, and has spent 19 years specializing in the storage area. Alex holds 10 certifications from IBM, Microsoft and Red Hat. Before joining the ITSO, he worked as a System Engineer for Tivoli®.

Patrick Március Médice Bisi is an IT Specialist at IBM Global Services in IBM Brazil. He has more than 10 years of experience in the IT industry. He holds a degree in Business Administration from SEDES/UVV and holds several certifications from Microsoft, including MCSE: Messaging and MCSA: Messaging. His areas of expertise include Microsoft Windows operating system, Microsoft Exchange servers and Active Directory® directory services.

Sven Schaffranneck is an IT Project Leader at Netzlink Informationstechnik GmbH, an IBM Business Partner from Germany. He has more than 10 years of experience in several areas of the IT industry. He is responsible for a high availability hosting environment including customer IT outsourcing, Microsoft Exchange and RIM BES infrastructure. Sven's current focus is on presales and postsales work on Microsoft Windows, Active Directory, Exchange and IBM system storage installations. Sven holds a Dipl.-Ing. degree in Computer Sciences from the University of Applied Science in Braunschweig/Wolfenbüttel, Germany.

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

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

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

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

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

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

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

Send us your comments in one of the following ways:

- Use the online **Contact us** review redbook form found at:
  `ibm.com/redbooks`
- Send your comments in an email to:
  `redbook@us.ibm.com`
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099, 2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks

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

AIX®	Lotus®	System Storage™
Domino®	Redbooks®	Tivoli®
IBM®	Redbooks (logo) ®	

The following terms are trademarks of other companies:

Snapshot, WAFL, SnapDrive, FilerView, Data ONTAP, and the Network Appliance logo are trademarks or registered trademarks of Network Appliance, Inc. in the U.S. and other countries.

Snapshot, Data ONTAP, WAFL, and NetApp logo are trademarks or registered trademarks of NetApp Corporation or its subsidiaries in the United States, other countries, or both.

Solaris, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

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

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