Technical Report: VERITAS VxFS

Backup and Recovery Using a NetApp Filer in a SAN Environment

Toby Creek | 10/1/2003 | TR 3281

TECHNICAL REPORT

Network Appliance, a pioneer and industry leader in data storage technology, helps organizations understand and meet complex technical challenges with advanced storage solutions and global data management strategies.

Abstract

This document details the implementation of file system freeze/thaw functionality with VERITAS File System (VxFS) with NetApp filer Snapshot™ copies to perform backup and recovery using iSCSI and Fibre Channel filers.

Network Appliance Inc.

Table of Contents

1) Purpose and Scope

This document covers the techniques for utilizing Snapshot technology available in Network Appliance™ storage systems with the VERITAS File System (VxFS) product. Specifically, this report covers the following issues:

Backing up a VxFS file system using Snapshot

Restoring a VxFS file system from a Snapshot backup

Connecting Snapshot copy-backed writable VxFS LUNs to a host

2) Requirements and Assumptions

For the methods and procedures in this document to be useful to the reader, several assumptions are made:

The reader has at least basic UNIX® administration skills and has access to the administrative login for the server.

An ANSI-C compiler is available on the UNIX server. The open-source GNU C-compiler is recommended on platforms where the vendor does not supply a compiler.

The reader has at least basic Network Appliance administration skills and has administrative access to the filer via the command-line interface.

The filer has the necessary licenses necessary to perform the activities outlined in this document.

The target system has the required block-level and network protocol interconnects to perform the activities outlined in this document.

In the examples in this report, all administrative commands are performed at the server or filer console for clarity. Web-based management tools can also be used.

This report and the code in the appendix were written for the Sun™ Solaris™ operating environment, but are applicable to other UNIX variants. Minor modifications to the source code provided may be required.

3) Best Practices

3.1) Snapshot and Storage Design

Recommendations for designing the storage configuration are presented in this section to help you utilize NetApp Snapshot most effectively. These recommendations are designed to prevent configuration issues from impacting data integrity or the ability to take or restore Snapshot images.

All VxFS-formatted LUNs in a given filer volume should be frozen during the Snapshot backup– Snapshot backups occur at the volume level on the filer. All LUNs within the filer volume are included in the Snapshot copy and should be consistent at the time the Snapshot copy is taken. All file systems using those LUNs should be frozen for the Snapshot creation. By doing this, any LUNs that are restored from Snapshot will be consistent.

If multiple hosts have LUNs in a single filer volume, prefix the Snapshot name with the host name – Snapshot can be synchronized with only one host at a time. Using the host name as the prefix

will allow the administrator to quickly identify which Snapshot copies are consistent for a given host. In these environments, only the single LUN SnapRestore® backup should be used.

Additional recommendations are presented in the *Network Appliance SAN System Administrator's Guide*. A link to this document can be found in the references section of this paper.

3.2) Archive to Tape Considerations

When using Snapshots as the source for archiving to tape, some conventional backup-to-tape solutions are no longer relevant. This section will present a few high-level recommendations when designing a three-tier backup solution using Snapshots.

Integrate the freeze/snap/thaw process into the prebackup facilities of the backup package –All enterprise-level backup packages have the ability to call other scripts to prepare the system for backup. The scripts presented in this paper can easily be integrated with leading packages to handle the creation of a Snapshot image before the backup-to-tape operation begins.

Allow the backup software to be the scheduler for Snapshot – All enterprise-level backup software packages have error notification and reporting functions built-in. By utilizing the backup software as the Snapshot scheduler, rather than another facility such as "cron," the reporting function can be accessed from a single console. The reporting capabilities will often be much more robust as well, with features such as notification to a network management console or pager.

If using NDMP, make sure that a consistent Snapshot copy is used as the source for the backup –During NDMP backup operations, the filer will create a Snapshot copy to use as a static backup image unless a specific Snapshot path is specified in the backup. Since the filer automatically creates the Snapshot backup, no file system synchronization is performed, and the file system being backed up may not be consistent. Specifying a Snapshot path, such as /vol/vol1/.snapshot/dbhost.0 will allow the administrator to select a consistent Snapshot copy for backup.

Additional recommendations may be made by the backup software vendor and Network Appliance. Consult the relevant documentation when designing a backup solution.

4) Using Snapshots with the VERITAS File System

4.1) Overview

The process of creating Snapshot backups in the SAN environment differs from the NAS environment in one very fundamental way: in the SAN environment, the filer does not control the state of the file system. For this reason, Snapshot must be initiated from the host after the appropriate operations have been performed to ensure that a consistent file system image is obtained in the Snapshot backup. These operations are commonly referred to as "freeze" and "thaw." The freeze operation flushes any dirty buffers in the file system cache, and then suspends new activity on the file system until the thaw operation is performed.

If freeze and thaw are not performed on the file system, there are several failure scenarios that are possible with file systems in general and VxFS in particular. In the first, minimal log replay will be required, in which changes from the journal are applied to the file system. This is generally not a time-consuming operation. In the second, a full file system sanity check, performed by the fsck process, will be required to return the file system to a usable state. This check can take from minutes to hours depending on the size of the file system, during which time the file system cannot be mounted or

otherwise accessed. Data may be lost during the fsck process. The final possibility is that the file system is completely unusable and all data is lost.

VERITAS provides an application programming interface (API) in VxFS via the UNIX “ioctl” to flush and freeze a file system prior to taking a Snapshot copy. This call takes a timeout as an argument. If the file system is not thawed manually, it will automatically be thawed when the timer expires. The Snapshot on the filer must occur before the file system thaws.

This paper makes use of a program called “vxfreeze” that is written to use the VERITAS API. It takes the file system mount point and an optional timer as arguments. The C program source can be found in the appendix. The vxfreeze program prints the process ID of a background process that issues the `freeze` command then returns an exit code of 0 upon successful completion. This background process can be signaled with the `kill` command to thaw the file system before the timer expires, if desired. If the background process is not signaled, it will terminate automatically when the thaw timer expires.

The simplest way to automate the freeze/snap/thaw process is to make use of a scripting language available on the host to call “vxfreeze” to perform the freeze and rsh on the filer to manage the Snapshot backups. The script presented uses the Korn shell. PERL, C-shell, and many other languages are also suitable to the task.

4.2) Software Installation

The steps required to prepare the UNIX server are outlined below.

1. Compile the source for vxfreeze into an executable and install the executable on the server. Compilation will require the VxFS file system packages to be installed since the VERITAS header files are needed. The compilation command will look something similar to the following:

   ```
   # gcc –I /opt/VRTSvxfs/include –o vxfreeze vxfreeze.c
   ```

2. Install the Korn shell script called nasnapvx.ksh into a directory on the server and edit the script to reflect the environment in which Snapshot will be used. This script is provided in the appendix of this paper.

4.3) Creating Snapshot Copies of VxFS File Systems

To take a Snapshot copy of a VxFS file system, execute the script from the command line as in the following example:

```
# ksh nasnapvx.ksh
```

The script will output basic status messages as it performs operations on the filer and the file systems involved in the Snapshot backup.

4.4) Restoring VxFS File Systems Using SnapRestore

Snapshot provides a very efficient and time-conserving way to restore file systems. Restoring a LUN that contains a VxFS file system is easily accomplished. The steps to restore a pool are detailed below:

1. Unmount the file system to be restored. If a volume-level SnapRestore process is used, all file systems with LUNs in the filer volume being restored must be unmounted.

   ```
   # umount /u01
   ```

2. Use the appropriate SnapRestore command on the filer. To restore a single LUN:

```
filer> lun offline /vol/vol1/vxfs0.lun
filer> snap restore -t file -s snap.0 /vol/vol1/vxfs0.lun
filer> lun online /vol/vol1/vxfs0.lun
```

To restore a filer volume and all of its LUNs:

```
filer> snap restore -t vol -s snap.0 vol1

WARNING! This will revert the volume to a previous snapshot.
All modifications to the volume after the snapshot will be
irrevocably lost.

Volume vol1 will be made restricted briefly before coming back
online.

Are you sure you want to do this? y

You have selected volume vol1, snapshot snap.0

Proceed with revert? y
Volume vol1: revert successful.
```

3. Remount the restored file systems once the restore has completed.

```
# mount /u01
```

4.5) Mounting Writable Snapshot Software-Backed LUNs

The mounting of writable Snapshot software-backed LUNs can be used to restore individual files or to allow the backup-to-tape process to occur on a second host to offload the process.

The procedure for mounting writable Snapshot software-backed LUNs is as follows:

1. Create the writable Snapshot software-backed LUN on the filer and map it to the host.

```
filer> lun create -b /vol/vol1/.snapshot/snap.0/vxfs0.lun
/vol/vol1/vxfssnap0.lun
filer> lun map /vol/vol1/vxfssnap0.lun vxfshost
lun map: auto-assigned vxfshost=1
```

2. Create the device files if none exist already.

```
# devfsadm
```

3. Mount the file system on the host.

```
# mount /dev/dsk/c1t2d1s6 /mnt
```

4. When the LUN is no longer needed, it can be unmounted and destroyed.

```
# umount /mnt
filer> lun destroy -f /vol/vol1/vxfssnap0.lun
```

This example is specific to Solaris, but the process is similar on other UNIX variants; only the individual operating system commands may differ.

5) Conclusions

A Network Appliance filer offers the UNIX administrator using VxFS compelling advantages in terms of backup and recovery. Use of Snapshot, combined with conventional backup-to-tape techniques, can dramatically optimize the server backup operation. Retaining a number of online Snapshot copies allows the system administrator to restore file systems without the need to restore from tape in many circumstances. Backup and recovery performance is dramatically improved over conventional local disk and SAN configurations, improving mean-time-to-recovery (MTTR) intervals.

6) Caveats

This paper is not intended to be a definitive implementation guide. There are many factors that may not be addressed in this document. Expertise may be required to solve logistical problems when the system is designed and built. NetApp has not tested this procedure with all of the combinations of hardware and software options available on UNIX variants. There may be significant differences in your configuration that will alter the procedures necessary to accomplish the objectives outlined in this paper. If you find that any of these procedures do not work in your environment, please contact the author immediately.

7) References

VERITAS file system documentation
http://www.veritas.com/van/products/filesystem.html

Network Appliance SAN System Administrator's Guide

http://now.netapp.com/NOW/knowledge/docs/ontap/rel641/html/ontap/san_sag/index.htm

8) Appendices

8.1) vxfreeze.c Source Code

```
/*
 * vxfreeze.c, version 1.0 - August 26, 2003
 *
 * This sample code is provided AS IS, with no support or
 * warranties of any kind, including but not limited to
 * warranties of merchantability or fitness of any kind,
 * expressed or implied.
 *
 * If this code does not work in your environment, please
 * notify the author: toby.creek@netapp.com
 *
 * Compile with:
 *   (g)cc -I /opt/VRTSvxfs/include -o vxfreeze vxfreeze.c
 *
```

```
 * Written on Solaris, but should compile on other platforms
 * with minor modifications
 *
 * Revision History
 * 1.0 - Initial release
 */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <sys/fs/vx_ioctl.h>
#include <sys/mnttab.h>
#include <errno.h>
#include <fcntl.h>
#include <signal.h>
#include <string.h>

/* Global Variables (mostly for the signal handlers) */
int vxfs_fd;

/* Signal Handlers */
void success() {
  exit(0);
}

void failure() {
  perror ("ERROR: VX_FREEZE ioctl failed!");
  exit(1);
}

void thaw() {
  if ( ioctl(vxfs_fd,VX_THAW,NULL) ) {
    perror("ERROR: Filesystem thaw failed");
    close(vxfs_fd);
    exit(1);
  } else {
    exit(0);
  }
}

/* Usage function */
void usage(char *command) {
  fprintf(stderr, "USAGE: %s [ -t timeout ]
filesystem_mount_point\n",command);
  exit(1);
}

/* main */
int main(int argc, char **argv) {

  long timeout = 30;
```

```
  uid_t iam = geteuid();
  pid_t gppid = getpid();
  int opt, status;
  char *mntpt;
  struct mnttab mnttab_entry, mnttab_filter;
  FILE *mnttabfile;

  fclose(stdin);

  /* Parse the args */
  while ((opt=getopt(argc,argv,"t:"))!=EOF) {
    switch(opt) {
      case 't':
        timeout = atol(optarg);
        break;
      case '?':
        usage(argv[0]);
        break;
    }
  }
  mntpt=argv[argc-1];

  /* Make sure they entered enough arguments, otherwise print usage */
  if ( argc < 2 ) {
    usage(argv[0]);
  }

  /* Check the timeout */
  if ( ( timeout < 10 ) || ( timeout > 60 ) ) {
    fprintf (stderr, "ERROR: Timeout must be between 10 and 60 seconds
(default 30).\n");
    exit(1);
  }

  /* Verify that the mount point is an absolute path */
  if ( strncmp(mntpt,"/",1) ) {
    fprintf (stderr, "ERROR: Filesystem mount point must begin with /.\n");
    exit(1);
  }

  /* Make sure that we are root */
  if ( iam != 0 ) {
    fprintf (stderr, "ERROR: %s must be run setuid/as root.\n",argv[0]);
    exit(1);
  }

  /* Get the mnttab entry */
  mnttabfile=fopen(MNTTAB,"r");
  if ( !mnttabfile ) {
    perror("ERROR: Unable to open the MNTTAB");
    exit(1);
  }
  mnttab_filter.mnt_special = NULL;
  mnttab_filter.mnt_mountp = mntpt;
```

```
  mnttab_filter.mnt_fstype = "vxfs";
  mnttab_filter.mnt_mntopts = NULL;
  mnttab_filter.mnt_time = NULL;
  if ( getmntany(mnttabfile, &mnttab_entry, &mnttab_filter) ) {
    fprintf (stderr, "ERROR: filesystem %s not found in MNTTAB or is not
VXFS.\n", mntpt);
    exit(1);
  }
  fclose(mnttabfile);

  /* Child does the work, Parent waits for signal to return status */
  if ( fork() == 0 ) {

    /* Child, setup the signal handlers */
    signal(SIGHUP, thaw);
    signal(SIGINT, thaw);
    signal(SIGTERM, thaw);

    /* Freeze filesystem, signal parent to return status */
    vxfs_fd = open(mntpt,O_RDONLY);
    if ( vxfs_fd < 0 ) {
      perror("ERROR: Failed to open device file");
      exit(1);
    }
    if ( ioctl(vxfs_fd,VX_FREEZE,timeout) ) {
      perror("ERROR: Filesystem freeze failed");
      close(vxfs_fd);
      sigsend(P_PID, gppid, SIGUSR2);
      exit(1);
    } else {
      sigsend(P_PID, gppid, SIGUSR1);
      /* Print our pid, then wait for later thaw signal */
      printf("%i\n", (int)getpid());
      fclose(stdout);
      sleep(timeout);
      close(vxfs_fd);
      exit(0);
    }
  } else {
    /* Parent */
    fclose(stdout);
    /* Setup to receive success/failure signal from child */
    signal(SIGUSR1, success);
    signal(SIGUSR2, failure);
    wait(&status);
    sleep(timeout);
    exit(0);
  }
}
```

8.2) nasnapvx.ksh Source Code

```
#!/bin/ksh

#** Configuration section **#
```

```
# Path to the vxfreeze executable
VXFREEZE=/usr/local/bin/vxfreeze

# Filer name or IP address
FILER="opaka"

# Filer volumes containing filesystem LUNs
VOLUMES="vol1"

# The filesystem(s) to freeze (in volumes above)
FILESYSTEMS="/vxfs1 /vxfs2"

# Snapshot prefix
SNAPPREFIX="vxfssnap"

# Number of snapshots to retain
SNAPSAVE=5

# Thaw timeout, in seconds
TIMEOUT=15



#** End configuration section **#

# Variable initialization
PIDS=""
export PIDS

# The freeze function
freeze()
{
      # Freeze the filesystems and record the PIDs
      for FILESYSTEM in $FILESYSTEMS; do
            PID=`$VXFREEZE -t $TIMEOUT $FILESYSTEM`
            RET=$?
            PIDS="$PIDS $PID"
            if [ "$RET" != "0" ]; then
                  thaw
                  exit $RET
            fi
      done
      return
}

# The thaw function
thaw()
{
      kill $PIDS
      return
}

# The snapshot rotation function
```

```
rotate()
{
      for VOLUME in $VOLUMES; do
            SNAPNO=$SNAPSAVE
            rsh $FILER "snap delete $VOLUME ${SNAPPREFIX}.${SNAPNO}"
            while [ $SNAPNO != 0 ]; do
                  SNAPLESSONE=`expr $SNAPNO - 1`
                  rsh $FILER "snap rename $VOLUME
${SNAPPREFIX}.${SNAPLESSONE} ${SNAPPREFIX}.${SNAPNO}"
                  SNAPNO=`expr $SNAPNO - 1`
            done
      done
      return
}

## MAIN

# Perform snapshot housekeeping
rotate

# Freeze the filesystems
freeze

# Take the snapshot
for VOLUME in $VOLUMES; do
      rsh $FILER "snap create ${VOLUME} ${SNAPPREFIX}.0"
done

# Thaw the filesystems
thaw

exit 0
```

Network Appliance, Inc.
495 East Java Drive
Sunnyvale, CA 94089
www.netapp.com