HP-UX JFS Freeze/Thaw

File System Backup and Restore Using a NetApp Filer in a SAN Environment

by Toby Creek, Network Appliance, Inc.

February, 2004 | TR 3308

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.

Table of Contents

Abstract

This document details the implementation of file system backup and restore using HP-UX's Journaling File System (JFS) with NetApp filer Snapshot™ technology to perform backup and recovery using iSCSI and Fibre Channel filers.

1. Purpose and Scope

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

- Backing up a JFS file system using Snapshot technology
- Restoring a JFS file system from a Snapshot backup
- Connecting writable Snapshot 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 HP-UX administration skills and has access to the administrative login for the server.
- An ANSI-C compiler is available on the HP-UX server. The open-source GNU C compiler is recommended on platforms where the vendor does not supply a compiler. Precompiled versions of GCC are available at the location listed in the references section.

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

- The filer and host have the necessary licenses 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 appendices were written for the HP-UX operating environment, but are applicable to other UNIX® variants. Minor modifications to the source code provided may be required.

3. Best Practices

3.1. Storage Design Considerations

To help users utilize NetApp Snapshot technology most effectively, this section presents recommendations for designing the storage configuration. These recommendations are designed to prevent configuration issues from impacting data integrity or the ability to create or restore a Snapshot backup.

All JFS file systems on logical volumes in LVM volume group LUNs must be frozen during Snapshot creation. Snapshot backups occur at the volume level on the filer. All LUNs within the filer volume are included in the Snapshot record image. All file systems on logical volumes in those volume group LUNs should be made consistent for the Snapshot creation. If this procedure is followed, any file systems that are restored from the Snapshot backup will be guaranteed to be recoverable. If the LVM volume group spans multiple filer volumes or filers, the Snapshot image for each filer volume must be created while the file systems are frozen.

If SnapRestore is used to restore LVM LUNs, all LUNs in the volume group must be restored at the same time. When a Snapshot image is created, all LUNs in the volume group must exist in the Snapshot image. The LVM volume group is the most granular object that can be restored from a Snapshot backup. When a volume group is restored, all LUNs (and consequently, all logical volumes and file systems) in the volume group are restored from the same Snapshot backup. The restore process will require the volume group to be deactivated for the restore, so all file systems on logical volumes within the volume group must be unmounted and unavailable during the restore. If new LUNs have been added to the volume group since the Snapshot image was created, they must be removed when the Snapshot backup is restored to ensure that the volume group configuration is returned to the configuration at the time the Snapshot image was created.

If multiple hosts have LUNs in a single filer volume, prefix the Snapshot record name with the host name. The Snapshot process is generally synchronized with only one host at a time. Using the host name as the prefix will allow the administrator to quickly identify which Snapshot image is 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 a Snapshot backup 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 for designing a three-tier backup solution using Snapshot technology.

Integrate the Snapshot process into the prebackup facilities of the backup package. All enterprise-level backup packages have the ability to call 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 the Snapshot creation. All enterprise-level backup software packages have error notification and reporting functions built in. If you utilize 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 record is used as the source for the backup. During NDMP backup operations, the filer will create a Snapshot image to use as the source unless a specific Snapshot path is specified in the backup. Since the filer automatically creates the Snapshot image, no file system synchronization is performed, and the file system being backed up may not be consistent. Specifying a Snapshot path, for example, `/vol/vol1/.snapshot/dbhost.0`, will allow the administrator to select a consistent Snapshot image to backup.

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

3.3. HP-UX Logical Volume Manager Considerations

Special care must be taken when JFS file systems reside in HP-UX Logical Volume Manager (LVM) volumes to maintain the correct volume group configuration.

The LUNs that make up an LVM volume group must be treated as a single unit. When a Snapshot process is triggered, all LUNs in that LVM volume group must exist in the Snapshot image. This will require that all file systems on logical volumes in the volume group need to be frozen while the Snapshot creation takes place. When a logical volume within the volume group is restored, all LUNs (and consequently, all volumes and file systems) in the volume group must be restored from the same Snapshot backup. Restoring the volume group will require the volume group to be deactivated for the restore, so all file systems within the volume group must be unmounted and unavailable during the restore. If new LUNs have been added to the volume group since the Snapshot image was created, they must be removed when the Snapshot backup is restored to ensure that the volume group configuration is returned to the same configuration that existed at the time the Snapshot image was created.

4. Using Snapshot Technology with the JFS File System

4.1. Overview

The Snapshot process in the SAN environment differs from that of 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, the Snapshot process 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 image. 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, several failure scenarios are possible with file systems in general and JFS 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.

Hewlett-Packard provides an application programming interface (API) in JFS via the UNIX "ioctl" system call to flush and freeze a file system prior to creating a Snapshot image. 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 operation 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 JFS API. It takes the file system mount point and an optional timer as arguments. The C program source can be found in the appendices. 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 process. 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 -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 technology is being employed. This script is provided in the appendices of this paper.

3. If using HP-UX 11.0 running on 64-bit machines, install patch PHKL_20973.

4.3. Creating a Snapshot Backup of a JFS File System

To create a Snapshot backup of a JFS 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 creating the Snapshot image.

NOTE: The use of the vxfreeze program on certain file systems can cause unexpected results. The / (root), /usr, /tmp, and /var file systems generally should not be frozen. Freezing these file systems will cause vxfreeze to behave unexpectedly, but is not considered to be dangerous.

4.4. Restoring JFS File Systems Using SnapRestore

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

1. Unmount all file systems on logical volumes in the volume group to be restored.

```
# umount /u01
# umount /u02
```

2. Deactivate the LVM volume group. If a volume-level SnapRestore command is used, all volume groups with LUNs in the filer volume being restored must be deactivated.

```
# vgchange -a n oravg
```

3. Use the appropriate SnapRestore command on the filer. To restore a volume group consisting of two LUNs:

```
filer> lun offline /vol/vol1/oraserv0.lun
filer> lun offline /vol/vol1/oraserv1.lun
filer> snap restore -t file -s snap.0
/vol/vol1/oraserv0.lun
filer> snap restore -t file -s snap.0
/vol/vol1/oraserv1.lun
filer> lun online /vol/vol1/oraserv0.lun
filer> lun online /vol/vol1/oraserv1.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.
```

4. Reactivate the LVM volume group.

```
# vgchange -a y vgora
```

5. Remount the restored file systems once the SnapRestore command has completed.

```
# mount /dev/vgora/lvol1 /u01
# mount /dev/vgora/lvol2 /u02
```

4.5. Mounting Writable Snapshot LUNs

The mounting of writable Snapshot 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 LUNs is as follows:

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

```
filer> lun create -b
/vol/vol1/.snapshot/snap.0/oraserv0.lun
/vol/vol1/oraservsnap0.lun
filer> lun create -b
/vol/vol1/.snapshot/snap.0/oraserv1.lun
/vol/vol1/oraservsnap1.lun
filer> lun map /vol/vol1/oraservsnap0.lun oraserv
lun map: auto-assigned oraserv=2
filer> lun map /vol/vol1/oraservsnap1.lun oraserv
lun map: auto-assigned oraserv=3
```

2. Scan for new disk devices on the host.

```
# ioscan -fnC disk
```

3. Install new special files for the disk if none exist already.

```
# ioinit -i
```

4. Use the "sanlun" or "ioscan" command to determine the paths to the new disk devices.

```
# sanlun lun show -p
```

or

```
# ioscan -funC disk
```

5. Change the volume group identifier on the Snapshot LUNs. Only specify one path to each device on the command line.

```
# vgchgid /dev/rdsk/c2t0d2 /dev/rdsk/c2t0d3
```

6. Create a new volume group for the Snapshot LUNs.

```
# mkdir /dev/vgorasnap
# mknod /dev/vgorasnap/group c 64 0x040000
# vgimport vgorasnap /dev/dsk/c2t0d2 /dev/dsk/c4t0d2
/dev/dsk/c2t0d3
  /dev/dsk/c4t0d3
```

This procedure will depend on the number of volume groups already on the host. All paths should be included on the import command line.

7. Activate the volume group.

```
# vgchange -a y vgorasnap
```

8. Mount the file systems from the logical volumes in the volume group.

```
# mount /dev/vgorasnap/lvol1 /u01.snap
# mount /dev/vgorasnap/lvol2 /u02.snap
```

9. When the LVM volume group is no longer needed, it can be unmounted and destroyed.

```
# vgchange -a n orasnapvg
# vgexport orasnapvg
# umount /u01.snap
# umount /u02.snap
filer> lun destroy -f /vol/vol1/aixservsnap0.lun
filer> lun destroy -f /vol/vol1/aixservsnap1.lun
```

5. Conclusions

A Network Appliance filer offers the UNIX administrator using HP-UX LVM and JFS compelling advantages in terms of backup and recovery. Use of Snapshot technology, combined with conventional backup-to-tape techniques, can dramatically optimize the server backup operation. Retaining a number of online Snapshot images allows the system administrator to restore file systems without the necessity to restore from tape in many circumstances. Backup and recovery performance is dramatically improved over that of 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

JFS 3.3 filesystem documentation
http://docs.hp.com/hpux/onlinedocs/B3929-90011/B3929-90011.html

Network Appliance Product Documentation
http://now.netapp.com/

Precompiled HP-UX software depots
http://hpux.cs.utah.edu

8. Appendices

8.1. vxfreeze.c Source Code

```
/*

 * vxfreeze.c, version 1.1 - January 18, 2004
 *
 * 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
 *
 * On Solaris, compile with:
 *   (g)cc -I /opt/VRTSvxfs/include -o vxfreeze vxfreeze.c
 * On HP-UX, compile with:
 *   gcc -D HPUX -o vxfreeze vxfreeze.c
 *  or
 *   cc -Aa -D HPUX -o vxfreeze vxfreeze.c
 *
 * Revision History
 * 1.0 - Initial release on Solaris
 * 1.1 - Ported to HP-UX
 */

#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>

#ifdef HPUX
#include <mntent.h>
```

```
#include <sys/signal.h>
#include <sys/procset.h>
#else
#include <sys/mnttab.h>
#endif

#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;
#ifdef HPUX
  struct mntent *mnttab_entry;
  int fs_found=0;
#else
  struct mnttab mnttab_entry, mnttab_filter;
#endif
  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);
  }

#ifdef HPUX
  mnttabfile=setmntent(MNT_MNTTAB, "r");
  while ( fs_found != 1 ) {
  mnttab_entry=getmntent(mnttabfile);
  if ( !mnttab_entry ) {
    fprintf (stderr, "ERROR: filesystem %s not found in MNTTAB or is
                        not VXFS.\n", mntpt);
    exit(1);
  }
  if ( !strcmp(mnttab_entry->mnt_dir,mntpt)
    && !strcmp(mnttab_entry->mnt_type,"vxfs") ) {
    fs_found=1;
    }
  }
  endmntent(mnttabfile);
#else
  /* 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);
#endif


  /* 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());
      setsid();
      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="/u01 /u02"

# Snapshot prefix
SNAPPREFIX="snap"

# Number of snapshot images 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.