Oracle® and Open Systems SnapVault®: Backup and Restore for UNIX® Platform

Darrin Chapman, Toby Creek, NetApp | TR-3377

Abstract

The Open Systems SnapVault (OSSV) client has extended the reach of Network Appliance™ SnapVault technology to the host. OSSV facilitates block-level incremental transfers from the open system platform directly to the secondary storage system. In Oracle environments, this approach can reduce the amount of data that must be transferred and perform dramatically more quickly than traditional file-based backup software.

NetApp Inc.

Table of Contents

1. Introduction

This document covers the techniques for backing up and restoring an Oracle Database running on a UNIX variant when a NetApp system is used for disk-based backup. Specifically, this report covers the following issues:

Backing up a database while the database is running ("hot" backup)

Recovering database files using OSSV

Recovering an entire database

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, has access to the administrative login for the server, and has administrative access to the server console.

The reader has at least basic NetApp administration skills and has administrative access to the controller or NearStore® system via the command-line interface.

The reader has at least basic Oracle Database administration skills and has access to the “sys” user password or another account with “sysdba” privilege.

The controller or NearStore system has the licenses necessary to perform the activities outlined in this document. Specifically, the controller or NearStore system will need the SnapVault secondary and SnapVault primary licenses installed.

The NetApp secondary system has the required block-level storage or 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 controller /NearStore console for clarity. Web-based management tools (DFM or NetVault) can also be used.

3. Backing Up an Oracle Database Using Open Systems SnapVault

3.1. Overview

Open Systems SnapVault (OSSV) functions by examining files for changes via two methods: modification time and block checksums. The modification time is a coarse estimation of the true amount of changed data, due to the fact that the modification time is updated when at least one block of the file is written. By using 4kB block checksums, OSSV is able to back up only the portions of the file that have changed.

OSSV can significantly reduce the amount of network traffic over traditional backup strategies by sending only incremental changes in increments of 4kB data blocks. Once the initial baseline transfer is complete, OSSV can send only changed blocks, effectively resulting in an “incremental forever” strategy.

During the OSSV transfer, the Oracle Database can be online (“hot” backup) or shut down (“cold” backup). Because most databases are not allowed downtime on a daily basis and more businesses operate 24x7, only the “hot” backup approach will be presented. Modifying the scripts and methods to implement a “cold” backup is trivial and can be left as an exercise to the reader.

3.2. Determining Backup Schedule and Retention Periods

Prior to establishing the initial full backup of the Oracle file system on the open systems platform, a backup schedule and retention policy should be determined. This technical report describes performing a scheduled incremental backup via a script that initiates a "manual" incremental (using `snapvault update`). Typically, the `snapvault snap sched` would be used to set up schedules and retention periods. Since we are concerned with the database state on the open systems platform, we are creating our own scripts that call `snapvault update` (incremental "pull" mechanism; NearStore system to open systems platform). Once complete, the script can be run as a `cron` job on a predetermined basis. An excellent resource for determining backup granularity retention policies is the: SnapVault Deployment and Configuration Guide.

3.3. Using Block-Level Incremental (BLI) Settings

Open Systems SnapVault BLI backup is designed to minimize the transfer of data that has not changed since a previous backup operation. OSSV uses checksums to identify portions of a file that have changed between a previous and the current backup.

A BLI backup recognizes that a file has changed based on a time stamp and checksum algorithm. Exactly which blocks have changed is determined, and only those blocks are sent to the secondary storage system.

Typically, incremental backups are more frequent, reduce the amount of time required to back up data, and minimize the resources required to perform backups when compared to baseline or full backups. BLI significantly reduces the amount of data that needs to be transferred to backup storage as well as the amount of data that must be stored on backup storage disks.

Changed blocks are recognized based on checksum values calculated and preserved for each block by the OSSV agent. Checksums are calculated on 4kB blocks of file data stored on an internal database. These checksum database files are stored in the OSSV internal DB directory. Each relationship has its own checksum file directories. Approximately 2% of the baseline ends up being your checksum file database size.

First, time stamps of files are compared to the time of the last successful backup operation. After being identified, a checksum is performed on that file. By default, every block of every file has a checksum operation performed against it during baseline operations. This is referred to as "high" BLI and results in typically longer transfer times and more CPU and disk consumption on the primary system. You can configure block incremental processing to trade off efficiencies among four variables: primary system CPU utilization, disk consumption, network bandwidth utilization, and Open Systems SnapVault transfer time. Enabling block-level incremental updates normally causes a checksum value to be calculated for every block of every file during the initial Open Systems SnapVault baseline transfer. As a result, baseline transfer execution time, CPU utilization, disk consumption, and network bandwidth utilization are increased compared to incremental transfers that are not block related. In this case, it is possible that significant resources can be consumed by calculating checksum values for static files that never change. The checksum levels can be configured as high, low, or off, using the svconfigurator utility.

> **High.** Always computes checksums, on baseline transfers and incremental updates. Primary impact: highest CPU utilization on the baseline transfer

Network impact: lower amount of data transferred on updates (incremental)
Secondary impact: same
When to implement: if all files are subject to small changes

Low. Compute checksums on changed files and only on updates; no checksum performed during baseline.

Primary impact: more CPU utilization during updates, faster baseline transfer
Network impact: large amount of data transferred during baseline, lower amounts of data transferred during updates
Secondary impact: same
When to implement: if a small subset of files is likely to change

Off. No checksums are calculated at any time; similar to older versions of OSSV. Full files are transferred once identified as being changed files.

Primary impact: fast baseline transfer, less impact on CPU during file system scan
Network impact: large amount of data transferred during baseline, potentially large amounts of data (large files) during updates as well
Secondary impact: same
When to implement: if a small subset of files is likely to change or files are changing completely

3.4. Creating a Baseline Transfer (Establishing a Relationship)

At this point, we have determined and configured schedules and retention policies for our environment. Once our script is in place, we will add it as a job to the `crontab`. However, SnapVault does not yet know which directories to back up or where to store them (which qtree) on the secondary. A baseline (level-0) transfer must first be initiated.

To provide SnapVault with this information, use the `snapvault start` command on the secondary.

In response to command line or NDMP-based management interface input, the SnapVault secondary storage system (controller or NearStore system) requests initial baseline (entire file system requiring backup) image transfers of directories specified for backup from an open systems platform. These transfers establish SnapVault relationships between the open systems platform *directories* and the SnapVault secondary *qtrees*.

The open systems platform, when prompted by the secondary storage system, transfers initial base images of specified directories to a qtree location on the secondary storage system. Once the baseline transfer has completed transfer, the secondary system will create a Snapshot™ copy (the baseline Snapshot copy) of the volume containing the destination qtree. If multiple transfers are occurring, faster transfers will be in a “quiescing” state until *all* transfers have completed.

A new Snapshot copy is created each time a backup is performed, and a large number of Snapshot copies can be maintained according to a schedule configured by the backup administrator. Each Snapshot copy consumes an amount of disk space equal to the differences between it and the previous Snapshot copy.

The Oracle Database will need to be in a file system, and that file system and its subdirectories will require that the baseline (relationship creation) transfer be completed before scheduled incremental

backups can begin. The `snapvault start` command (executed on the secondary system) will establish two things: (1) the full baseline transfer, including the baseline Snapshot copy and (2) the relationship creation. An alternate mechanism is to use an NDMP-based management application such as DataFabric® Manager (2.2 or higher) to perform the baseline transfer.

Note that in order for the baseline to be recoverable, Oracle should be in hot backup mode. Creating the baseline can take a significant amount of time and generate a large amount of network traffic. If recoverability of the baseline is less important than minimizing the impact of the baseline transfer, Oracle can be left in normal operating mode during the baseline process. If possible, the baseline can be created during off-peak hours to further minimize the impact to the server and network.

Examples of the `snapvault start` command used in our configuration:

```
zero> snapvault start -S benz:/u01 /vol/ossv/u01

zero> snapvault start -S benz:/u02 /vol/ossv/u02

zero> snapvault start -S benz:/u03 /vol/ossv/u03

zero> snapvault start -S benz:/u04 /vol/ossv/u04
```

The system “benz” is our primary open systems platform; “zero” is our R200 NearStore system (secondary); and /u01, /u02, /u03, and /u04 are all file systems on our UNIX system containing various components of the Oracle Database (control files, archives, redo logs, tablespace files, etc.). These commands would transfer all data in /u01, /u02, /u03, and /u04 to the R200 and place the data in qtrees /vol/ossv/u01 to /vol/ossv/u04. A mapping or relationship is created between directory and qtree at this point. A baseline Snapshot copy is created. Now "updates" or incremental transfers can occur. Each time an update occurs for a particular directory, the changed data is stored in a separate Snapshot copy on the NearStore system.

All file systems containing parts of the Oracle Database (binaries, data, control, log, and archived log files) should be added to the backup, though not all may have the same backup frequency. For example, the Oracle binaries need only be backed up if a patch is applied, and archived log files may be backed up more frequently during the day. All Oracle instances that have database-related files in the file system structures being transferred must be in backup mode at the time the OSSV update is begun.

Once the baseline transfer has completed successfully (“idle” output achieved from `snapvault status` on secondary) and the script is complete, the `cron` schedules can be configured. It is a good idea to plan and implement these schedules for the database backup prior to baseline transfer.

3.5. Performing an OSSV Backup

The scripts presented in the appendices of this paper implement a complete Oracle backup using OSSV. These scripts can be run manually or on a scheduled basis using a host-based scheduling tool.

The flow of the scripts is as follows:

1. Verify that a baseline relationship is established for the Oracle directories.
2. Call “beginhb.sh” to place all database tablespaces into backup mode.

3. For each Oracle directory, initiate the SnapVault update via an “rsh” call to the NearStore system.
4. **Wait for all transfers to complete; SnapVault *must be idle* prior to bringing the database out of hot backup mode.**
5. Call “endhb.sh” to return the database to normal operational mode.

The scripts contained in the appendix can easily be converted to a form usable on Windows® platforms with the addition of a UNIX toolkit or a PERL interpreter and the ONTAPI™ management interface libraries.

4. Database Restore and Recovery Using OSSV

4.1. Restoring a Database File

Certain scenarios (test purposes, DR scenarios, deleted tablespace, etc.) unfortunately require restores.

In our demonstration, the database file `/u03/OSSV/TEST9.DBF` has been accidentally deleted. Prior to beginning the restore using the OSSV agent, the tablespace must be taken offline using SQLPlus:

```
SQL> alter tablespace TEST9 offline immediate;
```

Once the tablespace is offline (“`Tablespace altered`” output from SQLPlus), we can perform the OSSV restore “snapvault restore” from the primary (the UNIX host):

```
benz# $INSTALL_DIR/bin/snapvault restore -S
zero:/vol/ossv/u03/OSSV/TEST9.DBF /u03/OSSV/TEST9.DBF
```

During the restore, it is a good idea to monitor the status of the transfer. Log in to the NearStore system or secondary system and monitor `snapvault status` or `snapvault status -l` to keep track of amount of data transferred, transfer time, and SnapVault state. Once a state of `Idle` is achieved, we can check the primary system for the restored file. When the file is in place, one last step is necessary to complete the restore process in an Oracle environment.

In SQLPlus, the tablespace must be recovered to bring it current with the rest of the database. To achieve this, we use:

```
SQL> recover tablespace TEST9;
```

Once the media recovery successfully completes (output to console `Media recovery complete`.), we can bring TEST9’s tablespace back online again:

```
SQL> alter tablespace TEST9 online;
```

Simply wait for `Tablespace altered` output from SQL. At this point, the tablespace recovery is complete.

4.2. Restoring and Recovering an Entire Database

In cases where the entire database is lost (such as a malicious intrusion or disaster recovery scenario), Open Systems SnapVault can quickly restore the database to operational status.

The general outline for this process is as follows:

1. If necessary, configure the hardware and restore the operating system first. Be sure to restore any Oracle tunables to the kernel configuration files.
2. Restore the Oracle installation and database files using OSSV (SnapVault restore). If the Oracle binaries were not backed up using OSSV, reinstall and patch to the correct level.
3. Once restored, the tablespaces are in hot backup mode, as they were when the backup was created. Database recovery will need to be performed.

```
$ sqlplus /nolog
SQL> connect sys/change_on_install as sysdba
Connected.
SQL> startup mount;
ORACLE instance started.

Total System Global Area  521635192 bytes
Fixed Size                   730488 bytes
Variable Size             234881024 bytes
Database Buffers          285212672 bytes
Redo Buffers                 811008 bytes
Database mounted.

SQL> recover automatic database;
…
SQL> alter database open;
Database altered.
```


5. Conclusions

The combination of a NetApp storage system and Open Systems SnapVault offers the Oracle Database administrator compelling advantages in terms of backup and recovery. Use of block-level incremental backups, combined with Snapshot technology, can dramatically optimize the Oracle Database backup operation. Backup performance and recovery performance are dramatically improved over conventional network backup solutions, dramatically 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 with Open Systems SnapVault. 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 authors immediately.

7. References

NetApp Data Protection Portal
www.netapp.com/solutions/data_protection.html

NearStore Overview
www.netapp.com/products/nearstore/

SnapVault Software
www.netapp.com/products/filer/snapvault.html

Leveraging SnapVault in a Heterogeneous Environment
www.netapp.com/tech_library/3234.html

SnapVault Deployment and Configuration
www.netapp.com/tech_library/3240.html

Nearline Applications for the Enterprise Utilizing NearStore
www.netapp.com/tech_library/3187.html

8. Appendices

8.1. backup_oracle.sh

```
#!/bin/ksh

# -----------------------
# Configuration variables
# -----------------------

# Oracle directories/filesystems
DIRECTORIES="/u01 /u02 /u03 /u04"

# The system hostname as known by the SV Primary
HOSTNAME="benz"

# The NearStore primary
NEARSTORE=zero

# Where the snapvault software is installed
SNAPVAULT_BIN=/usr/snapvault/bin

# The location of the backup scripts
SCRIPT_BIN="/u01/DARRIN"

# Oracle username
ORACLE_USER=oracle9

# Oracle System Identifier (SID)
ORACLE_SID=OSSV

# --------------------
# End of configuration
# --------------------

#
# Functions
```

```
#

# Die function
die()
{
      echo $1
      exit $2
}

#
# Main
#

# Set the path
PATH=$PATH:/usr/snapvault/bin:$SCRIPT_HOME
SNAPVAULT=$SNAPVAULT_BIN/snapvault

# Check configuration
if [ ! -f /usr/snapvault/bin/snapvault ]; then
      die "Cannot find snapvault executable (check /usr/snapvault/bin)" 1
fi

if [ ! -f $SCRIPT_BIN/beginhb.sh -a ! -f $SCRIPT_BIN/endhb.sh ]; then
      die "Cannot find oracle backup scripts (check SCRIPT_BIN)" 1
fi

# Check associations
for DIR in $DIRECTORIES; do
      SOURCEENT=`$SNAPVAULT status | awk '{print $1}' | grep $DIR'$'`
      if [ "$SOURCEENT" = "" ]; then
            die "No relationship exists for dir $DIR" 2
      fi
      HOST=`echo $SOURCEENT | cut -f1 -d:`
      DIRENT=`echo $SOURCEENT | cut -f2 -d:`
      if [ "$DIRENT" != "$DIR" -o "$HOST" != "$HOSTNAME" ]; then
            die "Ack!  Relationship for $DIR doesn't match primary $DIRENT" 2
      fi
done

# Put the database in hot backup mode
echo ""
echo "********************************"
echo "PLACING ORACLE IN HOTBACKUP MODE"
echo "********************************"
echo ""

sleep 5

echo "From Primary, Executing: beginhb.sh"
echo ""

su - $ORACLE_USER -c "$SCRIPT_BIN/beginhb.sh $ORACLE_SID"

# Start the SnapVault update
```

```
echo ""
echo "****************************************"
echo "STARTING INCREMENTAL BACKUP USING OSSV 2.0"
echo "****************************************"
echo ""

sleep 5

for DIR in $DIRECTORIES; do
      DESTENT=`$SNAPVAULT status | awk '{print $2 " " $1}' | grep $DIR'$' |
awk '{print $1}'`
      echo "From Secondary System, Executing: snapvault update $DESTENT"
      rsh $NEARSTORE "snapvault update $DESTENT" > /dev/null
done

# Give SnapVault time to wake up and start
sleep 15

# Loop until all status indicators are idle

echo ""
echo "****************************************"
echo "WAITING FOR SNAPVAULT TRANSFER TO COMPLETE"
echo "****************************************"
echo ""

echo "From Primary System, Executing: snapvault status"
echo "Waiting for IDLE status"
echo ""
IDLE=0
until [ $IDLE -eq 1 ]; do
      print -n  "."
      IDLE=1
      for DIR in $DIRECTORIES; do
            STATUSENT=`$SNAPVAULT status | awk '{print $5 " " $1}' | grep
$DIR'$' | awk '{print $1}'`
            if [ "$STATUSENT" != "Idle" ]; then
                  IDLE=0
            fi
      done
      sleep 2
done
echo ""
echo "TRANSFER COMPLETE!!!!"

# All status shows idle.  End hot backup

echo ""
echo "**********************************"
echo "TAKING ORACLE OUT OF HOTBACKUP MODE"
echo "**********************************"
echo ""
```

```
echo "From Primary, Executing: endhb.sh"
echo ""
su - $ORACLE_USER -c "$SCRIPT_BIN/endhb.sh $ORACLE_SID"

exit 0
```

8.2. beginhb.sh

```
#!/bin/sh
ORACLE_SID=$1
HBPATH=`dirname $0`

sqlplus -S /nolog @$HBPATH/beginhb.sql

exit 0
```

8.3. beginhb.sql

```
CONNECT system/manager
SET FEEDBACK off
SET PAGESIZE 0
SPOOL /tmp/begin.sql
SELECT
    'ALTER TABLESPACE ' ||
    tablespace_name ||
    ' BEGIN BACKUP;'
FROM
    dba_tablespaces
WHERE
    contents <> 'TEMPORARY';
SPOOL off
@/tmp/begin.sql
EXIT
```


8.4. endhb.sh

```
#!/bin/sh
ORACLE_SID=$1
HBPATH=`dirname $0`

sqlplus -S /nolog @$HBPATH/endhb.sql

exit 0
```

8.5. endhb.sql

```
CONNECT system/manager
SET FEEDBACK off
SET PAGESIZE 0
SPOOL /tmp/end.sql
SELECT
    'ALTER TABLESPACE ' ||
    tablespace_name ||
    ' END BACKUP;'
```

```
FROM
    dba_tablespaces
WHERE
    contents <> 'TEMPORARY';
SPOOL off
@/tmp/end.sql
EXIT
```