Oracle Hidden Secret: DNFS Cloning

Sebastian Solbach, ORACLE Deutschland B.V. & Co. KG

Es ist weitestgehend bekannt, dass Oracle seit der Datenbank-Version 11*g* einen NFS-3-Client in die Binärdateien implementiert hat. Dagegen schon eher ein Hidden Secret ist, dass diese NFS-Implementation eine Snapshot-Funktionalität besitzt, die optimal geeignet ist, um die Datenbank zu Testzwecken zu klonen, auch wenn das darunterliegende NAS-System Snapshots nicht unterstützt.

Diese Snapshot-Funktionalität hat beim 11*g*-Launch keine große Beachtung erfahren. Mit der Version 12*c* rückt diese Technologie aber wieder in den Fokus, da es in Verbindung mit Pluggable Databases durchaus um effiziente Cloning-Mechanismen geht und damit dem DNFS Clone auch wieder mehr Bedeutung zukommt. Aber auch mit existierenden 11*g*-Umgebungen hat diese Funktion viele Anwendungsfelder.

Direct NFS (DNFS)

Direct-NFS- beziehungsweise DNFS-Funktionalität ist die optimale Anbindung für Network Attached Storage (NAS) wie einer ZFS-Appliance, die per NFS angebunden ist. DNFS ersetzt für die Binärdateien den betriebssystemeigenen NFS-Client oder ergänzt, wie im Falle von Windows, das Betriebssystem um einen zertifizierten NFS-Client für die Oracle-Datenbank.

DNFS bietet dabei erhebliche Performance-Verbesserungen: Diese werden einerseits durch die Eliminierung des Betriebssystem-NFS-Layers bei einem Datenbank-Storage-Zugriff erreicht, andererseits wird eine bessere Performance dadurch erzielt, da der Storage nun über bis zu vier Pfade angesprochen werden kann (bei betriebssystemeigenen NFS-Clients nur zwei). Beim Einsatz von NFS-Storage und einer 11*g*-Datenbank empfiehlt es sich also, in jedem Fall DNFS einzusetzen.

Neben den Performance-Vorteilen bietet DNFS, wie schon erwähnt, noch eine weitere wichtige Funktion: DNFS Clone – die Möglichkeit, eine Datenbank zu Testzwecken zu klonen. Hierbei wird nicht, wie bei einem „RMAN DUPLICATE“, die komplette Datenbank kopiert, sondern es kommt die „Copy on Write“-Technologie zum Einsatz, um lediglich geänderte Blöcke festzuhalten. Damit ist ein DNFS Clone nicht nur schnell erzeugt, sondern nimmt auch relativ wenig Plattenplatz ein. Schaut man sich die Implementation näher an, stellt man fest, dass diese Funktionalität auch zur Verfügung steht, wenn die Produktiv-Datenbank selbst nicht auf NFS liegt.

Einrichtung DNFS unter 11*g*

Grundsätzlich können die Dateien einer Datenbank lokal im Filesystem, in ASM oder auf einem NFS-Filer liegen. Liegen sie auf NFS, wird normalerweise dieses Filesystem über den betriebssystemeigenen NFS-Client gemountet und erlaubt somit den Zugriff auf Datendateien, als ob es sich um ein lokales Dateisystem handelt.

Direct NFS erlaubt es nun, dass das Oracle Executable ohne den Umweg über den NFS-Client des Betriebssystems selbst direkt auf dieses NFS-Filesystem zugreifen kann. Hierzu muss der notwendige NFS-Client-Code im Oracle Executable aktiviert sein. Wenn alle Voraussetzungen erfüllt sind, reichen der Standard-NFS-Eintrag für Oracle-Datenbank-Dateien in der Datei „/etc/fstab“ und ein „make -f ins_rdbms.mk dnfs_on“ aus *(siehe Listing 1)*.

Unter 12*c* ist dieser Schritt nicht mehr notwendig, da hier der Default „dnfs_on“ ist. Anmerkung: Neu bei 12c ist auch, dass nun externe Tabellen mit DNFS angespro-

chen werden können. Davon profitiert auch der SQL Loader.

Wie die Anbindung eines NFS-Filers mit DNFS im Detail funktioniert, steht in der MOS Note DNFS: „Example About How To Setup DNFS (Direct NFS) On Oracle Release 11.2 [ID 1452614.1]“. Die Einsatzmöglichkeiten und Vorteile von DNFS sind im Whitepaper „Oracle Direct NFS Client with Oracle Database 11*g* Release 1“ beschrieben.

Zum Testen von DNFS und Ausprobieren des DNFS Clone ist allerdings kein NAS-Storage notwendig, es kann auch ein normaler Linux Server/Openfiler oder ein anderer Software-NFS-Server verwendet werden. Dies ist von Oracle zwar für den produktiven Einsatz nicht unterstützt, zum Testen reicht es aber. Zum Einrichten eines NFS-Servers stehen im Internet viele Anleitungen, auch Oracle hat eine simple Beschreibung im Zusammenhang mit DNFS: „Step by Step − Configure Direct NFS Client (DNFS) on Linux (11*g*) [ID 762374.1]“.

```
$ cd $ORACLE_HOME/rdbms/lib
$ make -f ins_rdbms.mk dnfs_on
rm -f /opt/oracle/product/11.2.0.3/db/lib/libodm11.so;
cp /opt/oracle/product/11.2.0.3/db/lib/libnfsodm11.so /opt/oracle/
product/11.2.0.3/db/lib/libodm11.so
```

Listing 1

```
rman target sys/oracle@orcl
RMAN> BACKUP AS COPY DATABASE FORMAT '/opt/oracle/backup/%U';
```

Listing 2

```
sqlplus sys/oracle@orcl
SQL> ALTER DATABASE BACKUP CONTROLFILE to TRACE;
```

Listing 3

Direct NFS Clone – ein Backup als Grundlage

Basis für ein DNFS Clone ist immer ein Backup, also ein RMAN-Backup oder ein Snapshot eines Storage-Systems. Dieses Backup muss über NFS erreichbar sein und erklärt auch, warum die Produktiv-Datenbank nicht auf NFS liegen muss. Für DNFS Clone muss kein eigenes Backup angelegt sein, es kann jedes vollständige Backup via Image-Kopie herangezogen werden. Ein Cold-Backup ist ebenso als Grundlage möglich wie ein Hot-Backup einer Datenbank im Archivelog-Modus. Möchte man explizit ein Backup für DNFS Clone via RMAN anlegen, so könnte dies über den Befehl „BACKUP AS COPY DATABASE FORMAT“ erfolgen *(siehe Listing 2)*.

Dabei entsteht auf „/opt/oracle/backup“ eine Image-Kopie. Es wird davon ausgegangen, dass „/opt/oracle/backup“ ein Verzeichnis eines NFS-Filers ist, das auf dem Testserver unter „/nfsmount“ gemountet wird. Zum Testen würde es auch reichen, dieses Verzeichnis mithilfe eines NFS-Shares zu exportieren, also dafür zu sorgen, dass es von einem NFS-Client gemountet werden kann. Es muss aber hier explizit erwähnt werden, dass dies nicht von Oracle unterstützt wird. Damit die Test-Datenbank erzeugt werden kann, muss sie ein Controlfile besitzen. Dieses

```
$ cp /opt/oracle/diag/rdbms/orcl/orcl/trace/orcl_ora_2413.trc createclonecontrolfile.sql
$ vi createclonecontrolfile.sql

STARTUP NOMOUNT
CREATE CONTROLFILE REUSE SET DATABASE "CLONE" RESETLOGS  ARCHIVELOG
    MAXLOGFILES 16
    MAXLOGMEMBERS 3
    MAXDATAFILES 100
    MAXINSTANCES 8
    MAXLOGHISTORY 292
LOGFILE
  GROUP 1 '/nfsmount/backup/clone/redo01.log'  SIZE 50M BLOCKSIZE 512,
  GROUP 2 '/nfsmount/backup/clone/redo02.log'  SIZE 50M BLOCKSIZE 512
DATAFILE
  '/nfsmount/backup/data_D-ORCL_I-1340774340_TS-SYSTEM_FNO-1_04o7cn44',
  '/nfsmount/backup/data_D-ORCL_I-1340774340_TS-SYSAUX_FNO-2_03o7cn3a',
  '/nfsmount/backup/data_D-ORCL_I-1340774340_TS-UNDOTBS1_FNO-3_05o7cn4t',
  '/nfsmount/backup/data_D-ORCL_I-1340774340_TS-USERS_FNO-5_07o7cn5d'
CHARACTER SET AL32UTF8
;
```

Listing 4

Controlfile wird aus der Primär-Datenbank erzeugt *(siehe Listing 3)*.

Der Befehl liefert ein SQL-Skript zum Erzeugen der Controlfiles für die Clone-Datenbank. Das erzeugte Trace sieht man im „alert.log" der Datenbank und findet sich unter „$ADR_BASE/rdbms/<dbname>/<dbname>/trace/". Das darin stehende SQL sollte nun für die Backup-Datenbank angepasst werden: Der Datenbankname wird geändert, die Datendateien sollten auf die Backupdateien verweisen und für die Redologs ein neues Verzeichnis verwendet werden *(siehe Listing 4)*.

Das Keyword „SET" ist dabei zusätzlich hinzuzufügen und wichtig, weil sonst dieses Kommando fehlschlägt, da der Datenbankname nicht zum Header der Datendateien passt. Damit die Datenbank gestartet werden kann, kopiert man am einfachsten das SPFile der Primär-Datenbank und ändert entsprechende Parameter. Hierzu gehören insbesondere die Parameter „Audit_file_dest, db_recovery_file_dest", „diagnostic_dest" und „control_files". Neu ist auch der „init.ora"-Parameter „CLONEDB", der auf „TRUE" gesetzt sein sollte, da ansonsten der Klon mit einem „ORA-01511" und „ORA-01141" fehlschlägt.

Listing 5 zeigt das Beispiel einer „initclone.ora"; die wichtigsten Änderungen sind rot markiert. Danach kann die Datenbank mit dieser „initclone.ora" gestartet und das Create Controlfile ausgeführt werden *(siehe Listing 6)*. Anschließend kommt der eigentliche Teil, den DNFS Clone zu aktivieren. Hierzu wird für jede einzelne Datendatei ein Klon erzeugt *(siehe Listing 7)*.

Dieser Befehl erzeugt nun die „Copy on Write"-Snapshot-Dateien. Sie müssen sich explizit ebenfalls in einem Verzeichnis befinden, das über DNFS angesprochen wird. Lokale Dateien sind nicht möglich. Dass diese Dateien kaum Platz brauchen, erkennt man nur mithilfe des Linux-Befehls „du". Die Dateien werden erst dann größer, wenn sich etwas an den Datendateien ändert.

Hat es sich beim Backup um ein HOT-Backup gehandelt, sind für die geklonten Datendateien ein Recovery und ein „open resetlogs" notwendig. Da jedoch im neu erstellten Controlfile keine Informationen zu den Backups von Archivelogs enthalten sind, wird der „RECOVER"-Befehl nach den entsprechenden Archivelogs fragen *(siehe Listing 8)*.

```
db_name='CLONE'
memory_target=512M
processes = 150
audit_file_dest='/opt/oracle/admin/clone/adump'
audit_trail ='db'
db_block_size=8192
db_domain=''
db_recovery_file_dest='/opt/oracle/fast_recovery_area'
db_recovery_file_dest_size=2G
diagnostic_dest='/opt/oracle'
dispatchers='(PROTOCOL=TCP) (SERVICE=ORCLXDB)'
open_cursors=300
remote_login_passwordfile='EXCLUSIVE'
undo_tablespace='UNDOTBS1'
control_files = (/nfsmount/backup/clone/clone.ctl)
compatible ='11.2.0'
clonedb = TRUE
```

Listing 5

Ab nun steht die Test-Datenbank zur Verfügung und man kann mit dieser arbeiten. Dabei werden nur Änderungen an den Datenfiles in den geklonten Files gespeichert und die Original-Backup-Dateien bleiben bestehen. Dies lässt sich gut anhand der Zeitstempel beziehungsweise der Dateigrößen beobachten.

Selbstverständlich erlaubt dieses Feature auch das Anlegen mehrerer Test-Datenbanken auf denselben Dateien. Wenn man die Test-Datenbank nicht mehr benötigt, löscht man einfach deren Files. Dies hat keine Auswirkungen auf die Produktion oder das Backup der Produktion. Das Backup hingegen sollte nicht gelöscht werden, da sonst die Test-Datenbanken nicht mehr funktionieren.

In der MOS Note „Clone your dNFS Production Database for Testing [ID 1210656.1]" ist zusätzlich zu einem weiteren Beispiel auch ein Perl-Skript enthalten, das den Clone-Prozess automatisiert und das Umbenennen der Dateien damit vereinfacht. Die Funktionalität des DNFS Clone bietet eine schnelle, platzsparende und günstige Variante, Testdatenbanken aufzusetzen.

Links und Referenzen

[1] Oracle Direct NFS Client with Oracle Database 11g Release 1: http://www.oracle.com/technetwork/articles/directnfsclient-11gr1-twp-129785.pdf

[2] Note ID 1210656.1: Clone your dNFS Production Database for Testing

[3] Note ID 1452614.1: DNFS: Example About How To Setup DNFS (Direct NFS) On Oracle Release 11.2

[4] Note ID 762374.1: Step by Step - Configure Direct NFS Client (DNFS) on Linux (11g)

[5] DBA Community Tipp: Direct NFS Clone – Klonen der Oracle Datenbank zum Testen, http://tinyurl.com/dba_community

Sebastian Solbach
sebastian.solbach@oracle.com

Alle weiteren Listings finden Sie online unter:
http://www.doag.org/index.php?id=1719