Da fliegt die Kuh – rasante Datenbank-Klone durch „copy on write“

Robert Marz, ist-people

Datenbanken aus der Produktion zu kopieren, ist eine I/O-intensive Angelegenheit. Das Herstellen von Klonen für Test- und Entwicklungssysteme kann für große Datenmengen Stunden dauern. Oracle hat mit dem Release 11*g* R2 die Funktion „clonedb“ eingeführt. Damit wird die Laufzeit des Klonvorgangs in den Bereich von Sekunden verkürzt. Der Einsatz von „copy on write“ (cow) in dNFS spart Plattenplatz und die geschickte Wahl des Filesystems auf der Quellsystem-Seite beschleunigt den Klon-Prozess zusätzlich.

Die allermeisten Datenbanken existieren in mehreren Versionen: Kopien, die mehr oder weniger Ähnlichkeit mit der Produktion haben, werden für Test- und Abnahmeumgebungen gebraucht. Entwickler benötigen eigene Datenbanken, auf denen sie sich austoben können. Diese sollten, zumindest in Bezug auf die Datenmengen, der Produktion möglichst nahe kommen, damit man bei der Produktivsetzung keine allzu bösen Überraschungen erlebt *(siehe Abbildung 1)*.

Das Erstellen echter Klone von Datenbanken ist Zeit-, I/O- und Ressourcenintensiv. Es müssen nicht unerhebliche Mengen an Daten kopiert werden, die im Anschluss Platz auf dem Filesystem belegen. Bedingt durch den hohen Zeitaufwand ist die Aktualisierungsfrequenz vollwertiger Klone typischerweise gering. Die Ähnlichkeit zum Original, Strukturen und Daten betreffend, nimmt mit der Zeit ab.

Abbildung 1: Der Bedarf an Kopien der Produktions-Datenbank ist hoch

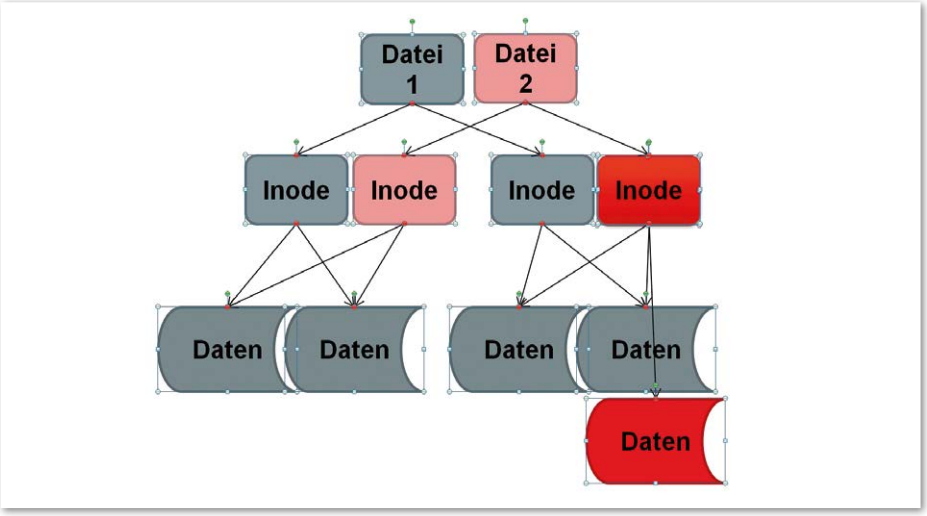

Abbildung 2: Bei Reflinks belegen nur die Inodes und veränderte Datenblöcke Platz

Thin Cloning mit „copy on write”

Das herkömmliche Kopieren dupliziert alle Blöcke einer Datei. Moderne Dateisysteme unterstützen als Alternative „Reflinks“, die die Datenblöcke des Originals unangetastet lassen und eine frische Verwaltungsstruktur anlegen, deren Inodes auf die ursprünglichen Datenblöcke zeigen. Das bedeutet, dass zunächst kein Speicherplatz belegt wird. Beim Ändern eines Datenblocks wird ein frischer Block angelegt und der Inode der zugehörigen Datei angepasst. Damit verhalten sich Original und Kopie wie eigenständige Dateien, die nur durch Änderungen zusätzlichen Platz belegen *(siehe Abbildung 2)*.

„Copy on write“ (cow) ist eine Strategie, die nicht auf Filesysteme beschränkt ist. Ihren Ursprung hat sie in der Verwaltung von Speicher- und Prozessstrukturen im RAM. Der Einsatz im Storage ermöglicht Snapshots und Transaktionen. Dateisysteme, die cow einsetzen, sind unter anderem OCFS2, ZFS und btrfs.

Wie bringt man nun die Datenbank dazu, cow für Datendateien zu benutzen? Eigentlich gar nicht. Mit dem Patchset 11.2.0.2 wurde allerdings der Oracle-direct-NFS-Client „dNFS“ eingeführt. Und damit geht es auf einmal doch …

Wenig beachtet und doch sehr mächtig gehört dNFS zum Lieferumfang jeder Datenbank, die eine Versionsnummer 11.2.0.2 oder höher trägt. Und das auf allen Unix-Plattformen und sogar unter Windows. Im Auslieferungszustand ist das Feature zunächst deaktiviert und muss erst durch Relinken eingeschaltet werden *(siehe Listing 1)*.

Danach verwenden alle I/O-Operationen der Datenbank, die auf ein im Betriebssystem eingebundenes NFS-Share zugreifen, automatisch den Oracle-NFS-Client und nicht mehr die nativen Funktionen des Betriebssystems. Das gilt für den Zugriff auf Datendateien genauso wie für RMAN-Backups oder Datapump-Exports.

Der dNFS-Client ist für die I/O-Anforderungen von Datenbanken optimiert worden. Er greift zum Beispiel parallel mit mehreren TCP-Streams auf den NFS-Server zu statt nur mit einem, wie das die Clients der Betriebssysteme tun. Es gibt eine eigene Konfigurationsdatei „oranfstab“, in der man unter anderem Zugriffspfade auf Server definieren kann. Als Minimal-Konfiguration reicht der Eintrag in der „/etc/mtab“, der beim Mounten durch das OS automatisch vorgenommen wird. Die Nutzung von dNFS kann in diesen Performance-Views überwacht werden:

- v$dnfs_servers
- v$dnfs_files
- v$dnfs_channels
- v$dnfs_stats

Cow in der Datenbank: „clonedb“

Zusammen mit dNFS hat Oracle „clonedb“ eingeführt, zunächst nicht als offizielles Feature, sondern als Prozeduren und Funktionen im System-Package „dbms_dnfs“. Beschrieben ist es in der MOS-Note „Clone your dNFS Production Database for Testing (Doc ID 1210656.1)“. Dort gibt es auch ein Perl-Script zum Herunterladen, das beim ersten Erzeugen der nötigen Skripte hilft und so zum Verständnis der Mechanismen beiträgt.

Als Voraussetzung benötigt „clonedb“ Zugriff auf einen Satz kopierter Datenfiles,

```
$ cd $ORACLE_HOME/rdbms/lib
$ make -f ins_rdbms.mk dnfs_on
```

Listing 1

```
sqlplus / as sysdba
STARTUP NOMOUNT PFILE=?/dbs/initKLON.ora
CREATE CONTROLFILE REUSE SET DATABASE KLON RESETLOGS
    character set WE8ISO8859P15
LOGFILE
  GROUP 1 '/oradata/KLON/onlinelog/KLON_log1.log' SIZE 100M BLOCKSIZE
512,
  GROUP 2 '/oradata/KLON/onlinelog/KLON_log2.log' SIZE 100M BLOCKSIZE
512
DATAFILE
  '/base_copy/PRODDB/oradata/o1_mf_ak_data_h8om7smf_.dbf'
, …
```

Listing 2

```
dbms_dnfs.clonedb_renamefile
  ( '/base_copy/PRODDB/oradata/o1_mf_ak_data_h8om7smf_.dbf'
  , '/nfsstore/o1_mf_ak_data_h8om7smf_.dbf'  );
```

Listing 3

die der neuen Klon-Datenbank „read only“ zur Verfügung gestellt werden können. Für diese Datenfiles wird dann ein neues Controlfile erzeugt *(siehe Listing 2)*.

Danach kommt der spannende Teil: Für jede Datendatei aus dem Backup der Originaldatenbank wird ein PL/SQL-Aufruf ausgeführt *(siehe Listing 3)*. Dies teilt der Datenbank mit, dass alle Änderungen an der Originaldatei in die Datei aus dem zweiten Parameter geschrieben werden sollen. Dieser Dateipfad muss ein NFS-Share sein, das die Datenbank über dNFS anspricht. Ob die Datei tatsächlich auf einem entfernten Host, NAS oder dem lokalen Server liegt, ist dabei egal.

Danach kann man – gegebenenfalls nach einem Recovery – die Datenbank öffnen und ganz normal mit ihr arbeiten. Die Zieldateien auf dem NFS-Share belegen zunächst keinen Platz. Erst wenn geänderte Blöcke dort hineingeschrieben werden, wachsen sie langsam bis maximal zur Größe der Originaldatei. Beim Lesen wird zuerst geschaut, ob der Block bereits geändert wurde, und falls nicht, wird er aus der Backup-Datei gelesen *(siehe Abbildung 3)*. Performance-Einbußen sind kaum zu spüren. Oracle spricht bei eigenen Messungen von drei bis zehn Prozent Nachteil gegenüber direktem Zugriff auf die Datenfiles.

Viele Entwicklungsumgebungen auf vielen Rechnern

Die verschiedenen Oracle-Beschreibungen von „clonedb“ gehen davon aus, dass die Klone auf demselben Server wie die Original-Datenbank erstellt werden. In der Praxis sollen die Klone durchaus aber auch auf anderen Servern laufen, was aber kein großes Problem darstellt, da auch die Basiskopie der Datenfiles via NFS beliebig vielen Rechnern zur Verfügung gestellt werden kann.

Bleibt noch das Problem der Erstellung des Basis-Backups. Wenn das unterliegende Storage-System Snapshots erstellen kann, ist die Sache schnell erledigt. Falls nicht, bietet es sich an, die Kopie der Datenfiles als Reflinks in Sekundenschnelle durchzuführen *(siehe Listing 4)*. Voraussetzung ist, dass die Datendateien auf einem „cow“-fähigen Dateisystem liegen. Auch in diesem Fall belegen die Dateien zunächst keinen Platz, sondern wachsen erst, wenn sich die Originaldateien ändern.

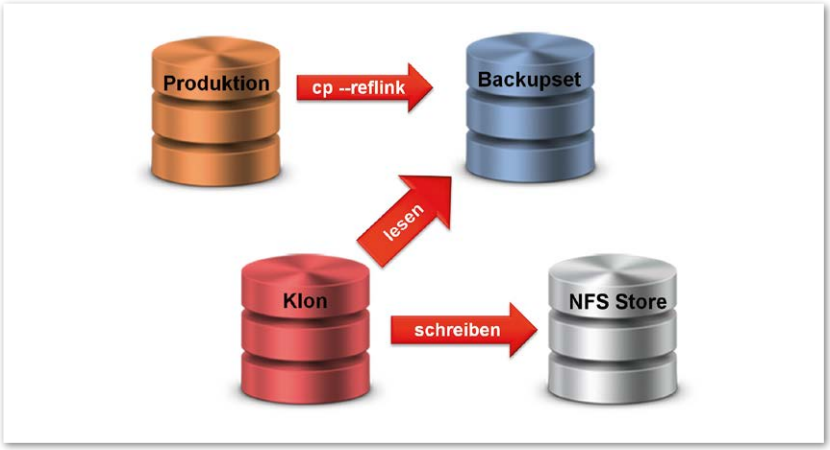

Abbildung 3: „clonedb"-Klone lesen aus dem Backup-Set der Produktion und schreiben geänderte Blöcke über dNFS

```
SQL> spool copy-db.sh
SQL> select ' cp --reflink "' ||file_name|| '" "$destdat"'  from dba_
data_files;
SQL> spool off
SQL> alter system archive log current;
SQL> alter database begin backup;
SQL> !sh copy-db.sh
SQL> alter database end backup;
```

Listing 4

Die Auswahl des Dateisystems

Oracle selbst zertifiziert keine Dateisysteme für den Einsatz mit der Datenbank, sondern ausschließlich Betriebssysteme inklusive deren unterstützte Dateisysteme. Auf Solaris-Betriebssystemen ist das von Oracle entwickelte ZFS sicherlich erste Wahl. ZFS gibt es zwar auch für Linux, allerdings nicht in den Standard-Distributionen, sondern als Erweiterung. Bis es mit den Kernels der großen Distributionen ausgeliefert wird, wird es noch eine Weile dauern.

Das btrfs-B-tree-Filesystem – ausgesprochen „Butter-FS", manchmal auch „Better-FS" – wird ebenfalls von Oracle entwickelt und gilt als das zukünftige Standard-Filesystem für Linux. Es hat mittlerweile einen stabilen Zustand erreicht, wird aber noch weiterentwickelt. Oracle rät im Moment in der MOS-Note „Supported and Recommended File Systems on Linux [ID 236826.1]" von einem Einsatz für Datenfiles in produktiven Umgebungen ab. Erfahrungen zeigen, dass es durchaus performant und stabil funktioniert.

Bleibt noch das Oracle-Cluster-Filesystem „OCFS2". Es ist von Oracle explizit für Datenfiles entwickelt worden und unterstützt „Reflinks". Allerdings klinkt es sich unter Linux nicht in den „--reflink"-Schalter von „cp" ein, sondern bringt ein Extra-Kommando „reflink" mit.

Fazit

„clonedb" und „dNFS" erstellen schlanke Datenbank-Klone in beinahe beliebig großer Zahl und sehr hoher Geschwindigkeit. Solange sich die Datenänderungsrate in den Klonen in Grenzen hält, ist der benötigte Plattenplatz um ein Vielfaches kleiner als der, der für andere Klone benötigt würde.

Robert Marz
robert.marz@its-people.de