Da fliegt die Kuh
Rasante Datenbankklone durch cow (copy-on-write)

Robert Marz
ist-people GmbH
Frankfurt

Schlüsselwörter
Datenbank Klone, clonedb, cow (copy-on_write)

Einleitung

Datenbanken für Test- und Entwicklungssysteme aus der Produktion zu klonen, ist typischerweise eine recht I/O-intensive Angelegenheit, die bei großen Datenbanken schon mal Stunden dauern kann.

Seit Oracle 11gR2 gibt es clonedb, das diesen Vorgang nicht nur auf wenige Sekunden verkürzt, sondern durch den Einsatz von copy-on-write (cow) Technologie in dNFS jede Menge Platz auf der Platte spart.

Die Wahl des richtigen Filesystems auf der Quellsystem-Seite beschleunigt den Klonprozess zusätzlich.

Dieser Vortrag versucht folgende Fragen zu beantworten:

- Wie funktioniert die cow-Technologie?
- Wie kann die Oracle Datenbank dazu gebracht werden, sie zu verwenden?
- Wie richtet man dNFS und clonedb ein?
- Welches Dateisystem verwende ich?
- Und wie funktioniert das über Rechnergrenzen hinweg?

Warum Klonen?

Die allermeisten Datenbanken existieren in mehr als einer Ausführung:
Kopien, die mehr oder weniger Ähnlichkeit mit der Produktion haben, werden für Test- und Abnahmeumgebungen gebraucht. Und auch Entwickler wollen eigene Datenbanken haben, die zumindest von den Datenmengen der Produktion möglichst ähnlich sind, um bei der Produktivsetzung keine allzu bösen Überraschungen zu erleben.

Abbildung 1:Der Bedarf nach Kopien der Produktionsdatenbank ist hoch

Das Erstellen echter Kopien oder Klone von Datenbanken ist zeit-, IO- und ressourcenintensiv. Schließlich müssen nicht unerhebliche Mengen an Daten kopiert und Platz für diese vorgehalten werden.

Durch den hohen Zeitaufwand werden vollwertige Klone typischerweise nur selten aktualisiert und haben mit den Dateninhalten der Produktionsumgebung nach einiger Zeit nur noch wenig zu tun.

Thin Cloning mit cow (copy-on-write)

Ein schnellerer Weg zum Klon, als vollständiges Kopieren der Datenfiles, erfolgt über sogenannte „reflinks“, eine Mischung aus Link und Kopie.

Werden beim normalen Kopieren alle Datenblöcke dupliziert, werden beim reflink nur die Inodes dupliziert, die Datenblöcke bleiben unangetastet. Erst wenn in einer der Dateien ein Datenblock geändert wird, wird dieser nicht überschrieben, sondern als neuer Block angelegt und Inode der zugehörigen Datei angepasst. Dadurch bleiben die Datenblöcke der anderen Dateien unverändert.

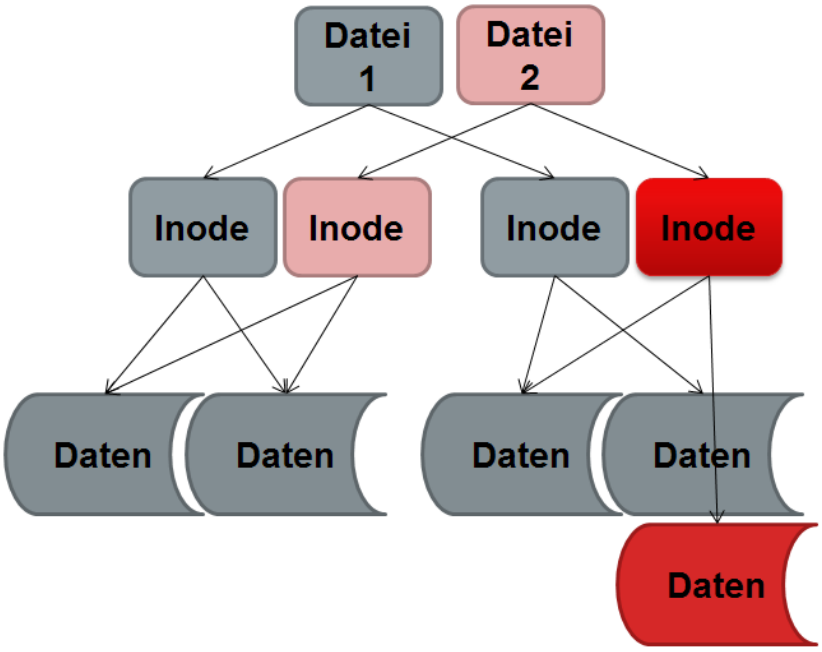

Abbildung 2: Bei Reflinks belegen nur die Inodes und veränderte Datenblöcke Plabtz

Copy-on-write wird sowohl bei der Verwaltung von Speicherstrukturen im RAM, als auch in Storage- und natürlich in Dateisystemen zum Einsatz, bei Letzteren ermöglicht es 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?
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…

Exkurs: NFS Client von Oracle: dNFS
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:

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

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

Der dNFS-Client ist für die IO-Anforderungen von Datenbanken optimiert worden und 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 z.B. Zugriffpfade auf Server definieren kann. Als Minimalkonfiguration 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 clondb 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 Vorraussetzung benötigt clonedb Zugriff auf einem Satz kopierter Datenfiles, die der neuen Klon-Datenbank auch Read-Only zur Verfügung gestellt werden können. Für diese Datenfiles wird dann ein neues Controlfile erzeugt:

```
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'
, …
```

Danach kommt der spannende Teil:

Für jede Datendatei aus der Originaldatenbank aus dem Backup wird folgender PL/SQL-Aufruf ausgeführt:

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

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, welches 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, der aus der Backup-Datei gelesen.

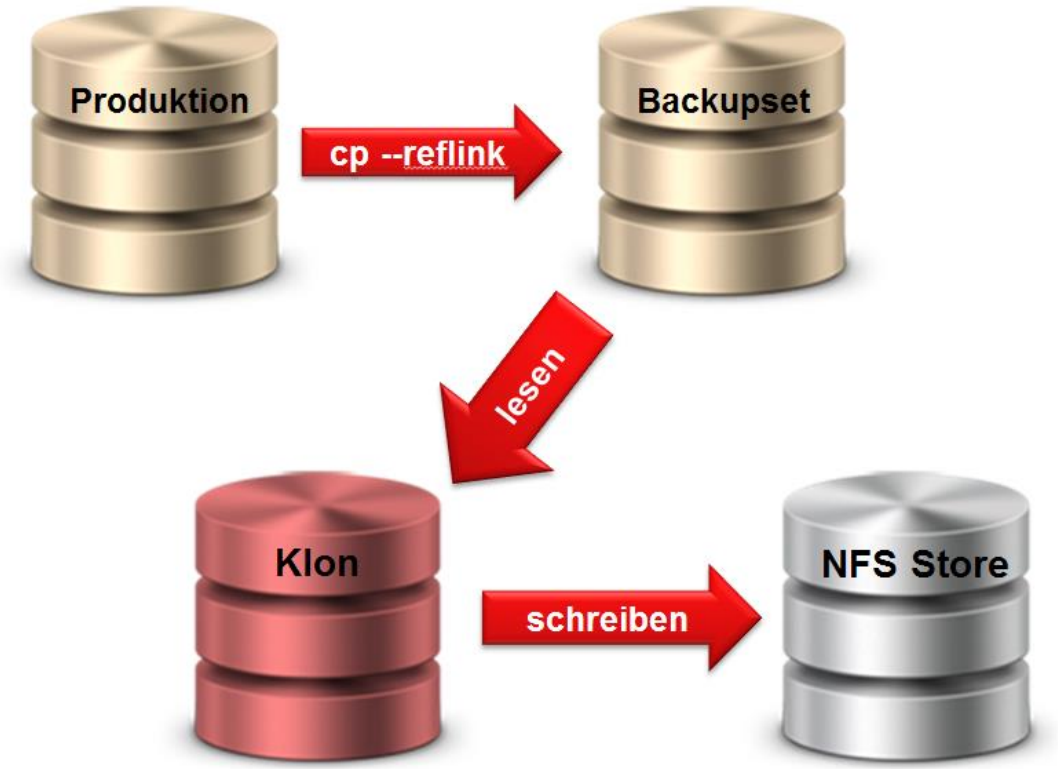

Abbildung 3 clonedb Klone lesen aus dem Backupset der Produktion und schreiben geänderte Blöcke über dNFS

Performance-Einbußen sind kaum zu spüren. Oracle spricht bei eigenen Messungen von 3-10% Nachteil gegenüber direktem Zugriff auf die Datenfiles.

Viele Entwicklungsumgebungen auf vielen Rechnern

Die Beschreibungen von clonedb gehen davon aus, dass die Klone auf demselben Server, wie die Originaldatenbank 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 Basisbackups. 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:

```
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;
```

Voraussetzung ist, dass die Datendateien auf einem cow-fähigem Dateisystem liegen. Auch in diesem Fall belegen die Dateien zunächst keinen Platz im Dateisystem, sondern wachsen erst, wenn sich die Originaldateien ändern.

Welches Dateisystem

Welches cow-Dateisystem soll man nun verwenden?

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-FS, ausgesprochen Butter-FS, manchmal auch Better-FS – wird ebenfalls von Oracle entwickelt und gilt als das zukünftige Standard-FS 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 das durchaus performant und stabil funktioniert.

Bleibt noch das Oracle-Cluster-Filesystem OCFS2. Das 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

Mit clonedb und dNFS können schlanke Datenbankklone in beinahe beliebig großer Zahl und sehr hohen Geschwindigkeiten erstellt werden. Solange sich die Datenänderungsrate in den Klonen in Grenzen hält, ist der benötigte Plattenplatz um ein vielfaches kleiner, als der, der für „fette“ Klone benötigt würde.

Die Quelltexte zu den in diesem Vortrag gezeigten Demonstrationen können nach der DOAG 2013 Konferenz von unserem its-people-Blog heruntergeladen werden:

http://www.its-people.de/blog

Kontaktadresse:

Robert Marz

its-people GmbH
Lyoner Straße 44-48

60528 Frankfurt am Main

Telefon:	+49 (69) 247521-00
Fax:	+49 (69) 247521-021
E-Mail:	robert.marz@its-people.de
Internet:	www.its-people.de