Rolling Patching mit DBMS_ROLLING und Active Data Guard

Nikolaus Eichler
Selbständiger Berater
42781 Haan

Schlüsselworte:

Minimal-Downtime, Rolling Patching, DBMS_ROLLING, Active Data Guard

Einleitung

Häufig besteht die Anforderung, Ausfallzeiten beim Patching möglichst kurz zu halten. Bei der herkömmlichen Methode des Patchings fehlt oft die Zeit, den Patch zu testen, um die Ausfallzeiten für die Benutzer möglichst zu minimieren. Das Datenbank-Package DBMS_ROLLING hilft in Kombination mit einer Active Data Guard (ADG) Umgebung dabei, dieses Ziel zu erreichen. Dieser Vortrag beschreibt alle Schritte dieses Prozesses und weist auf Stolpersteine hin, aber auch darauf, wie man sie vermeiden kann. Es wird erläutert, wie man mit Hilfe dieses Vorgehens einen Patch testen und dennoch eine reine Datenbank-Ausfallzeit von nur typischerweise 20 - 30 Sekunden haben kann. Der Vortrag stützt sich auf Erfahrungen mit dem Datenbank-Release 12.2.

Das DBMS_ROLLING Package – Was macht es ?

Das DBMS_ROLLING Package erleichtert sogenannte Rolling-Operationen (Upgrades, Patching, Strukturänderungen in der Datenbank, usw.), indem es viele Schritte automatisiert, die ansonsten manuell ausgeführt werden müssten. Das Prinzip von Rolling-Operationen mit Active Data Guard beruht darauf, dass die Datenbank-Benutzer, abgesehen von einer kurzen Auszeit, immer eine ansprechbare Datenbank zur Verfügung haben. Das funktioniert, indem die Benutzer zunächst auf der Primärseite arbeiten, während die Standby-Seite gepatcht wird. Danach erfolgt ein Rollentausch (Switchover) der Datenbanken. Die Benutzer können dann auf der neuen Primary Datenbank (der früheren Standby) weiterarbeiten, während die frühere Primary Datenbank gepatcht wird. Die Auszeit entsteht also nur durch das Switchover auf die gepatchte Datenbank und das Reconnect der Benutzer-Sessions.
Danach kann ein erneutes Switchover stattfinden, so dass die Ausgangskonfiguration wieder hergestellt ist, aber mit dem neuen Patch-Level.

Voraussetzungen

Um das DBMS_ROLLING Package einsetzen zu können, muss eine Active Data Guard-Lizenz vorhanden sein. Im Gegensatz zum 12.1 Release unterstützt das Package eine Data Guard Broker Konfiguration. Fast-Start Failover muss während des Einsatzes von DBMS_ROLLING aber ausgeschaltet sein. Der Maximum-Protection Mode muss ebenfalls ausgeschaltet sein.
Da der Betrieb einer ‚Transient Logical Standby'-Datenbank Einschränkungen bezüglich der unterstützten Datentypen hat, wurde im Vorfeld des Patchings überprüft, ob solche Datentypen in der vorliegenden Datenbank vorhanden sind:

```
SQL> select OWNER,TABLE_NAME from DBA_ROLLING_UNSUPPORTED;
```

Falls solche Datentypen vorhanden sind, müssen diese mit dem im „Data Guard Concepts and Administration" Handbuch beschriebenen Verfahren (Kapitel: Enabling EDS-Based Replication at a Logical Standby) repliziert werden.

In unserem Fall waren keine unsupporteten Datentypen vorhanden.

Historie

Bis zum Datenbank Release 11.2.0.1 mussten Physical Standby Datenbanken exakt den gleichen Release- und Patch-Stand haben, wie die Primary Datenbank. Daraus ergab sich, dass zum Patching beide Seiten offline sein mussten. Erst mit dem Release 11.2.0.2 gab es bestimmte Patches, die im Patch-README den folgenden Hinweis zeigten: ‚This patch is Data Guard Standby-First Installable'. Mit diesen Patches war es erstmals möglich, unterschiedlich versionierte Data Guard Konfigurationen im Rahmen einer ‚Physical Standby'-Konfiguration zu betreiben und somit ein Rolling Upgrade durchzuführen
Seit dem Release 10.1.3 gibt es aber das Konzept der ‚Logical Standby Database' bei welchem die Standby Datenbank nicht durch identische Block-Kopien synchronisiert wird, sondern durch die Anwendung von SQL-Statements, die aus den Redo Log-Informationen der Primary Datenbank gewonnen werden. Dieses Verfahren hat zwar auch Nachteile in Bezug auf Limitierungen bei der Unterstützung komplexer Datentypen, aber den Vorteil, dass es den Betrieb unterschiedlicher Datenbank-Release-Stände in einer Konfiguration erlaubt.
Mit dem Release 11.1 kam dann das Konzept der ‚Transient Logical Standby-Datenbank. Damit wurde es sehr einfach möglich, eine Physical Standby Datenbank temporär in eine Logical Standby Datenbank zu verwandeln und somit auch mit zwei unterschiedlichen Releases in einer Konfiguration zu arbeiten. Das ist die Grundlage des ‚Rolling Patching' (und auch des Rolling Upgrades). ,
Die dafür erforderlichen Schritte werden seit dem Release 12.1 vom DBMS_ROLLING-Package entscheidend unterstützt.

Das Verfahren im Praxistest

Dieses Verfahren habe ich angewandt, um zusammen mit einem einen Kunden auf einer 12.2 Multitenant Datenbank ein Patch Set Update (PSU) einzuspielen. Ziel dabei war es, die Ausfallzeit für die Benutzer zu minimieren und zu testen, wie tolerant dieses Verfahren auf Fehler reagiert und wie hilfreich diese Fehlermeldungen dann sind.
Ausgangspunkt war eine fertig konfigurierte Active Data Guard-Umgebung in einer Data Guard Broker Konfiguration mit einem Observer (Fast-Start-Failover Konfiguration)

Im Unterschied zum Release 12.1 unterstützt das DBMS_ROLLING Package in der 12.2 das Arbeiten mit dem Data Guard Broker. Der Observer muss aber nach wie vor ausgeschaltet sein.
Um zu testen, inwieweit das Package Hilfestellung bei Fehlern leistet, haben wir uns dabei an zwei Stellen nicht exakt an das Handbuch gehalten. Eine Voraussetzung für den Einsatz des Packages war (s.o.), dass der Fast-Start-Failover Prozess abgeschaltet ist. In unserem Fall war der Fast-Start-Failover Prozess nicht abgeschaltet, sondern der Status zeigte im Broker ein ‚disconnected' an. Uns interessierte, ob dieser Status ausreicht, um das Package erfolgreich auszuführen.

Beginn

Das DBMS_ROLLING Package besteht aus acht Prozeduren, von denen im einfachsten Fall nur fünf gebraucht werden, um den kompletten Zyklus des Patchens einer Primary- und einer Standby-Datenbank abzuschließen. Ein sechster Schritt ist empfehlenswert und zwei weitere Prozeduren sind optional.
Wenn die oben beschriebenen Voraussetzungen gegeben sind, dann versetzt man die Standby-Datenbank vom READ ONLY Modus (Active Data Guard) in den Mount-Status.
Danach ruft man die folgenden Prozeduren des DBMS_ROLLING-Packages als User SYS (Bei einer Multitenant Datenbank wird der User SYS des Root-Containers verlangt) auf.

1.) Als Parameter übergibt man der ersten Prozedur den DB_UNIQUE_NAME der Standby-Datenbank, welche die zukünftige Primary werden soll:

```
SQL> EXEC DBMS_ROLLING.INIT_PLAN (future_primary => 'stdby');

PL/SQL procedure successfully completed.
```

Danach kann man die Plan-Parameter mit folgendem Statement überprüfen:

```
SELECT SCOPE, NAME, CURVAL FROM DBA_ROLLING_PARAMETERS ORDER BY SCOPE,
NAME;
```

```
SCOPE      NAME                           CURVAL
---------- ------------------------------ ------------------------------
prod       INVOLVEMENT                    FULL
prod       MEMBER                         NONE
stdby      INVOLVEMENT                    FULL
stdby      MEMBER                         TRAILING
           ACTIVE_SESSIONS_TIMEOUT        3600
           ACTIVE_SESSIONS_WAIT           0
           BACKUP_CONTROLFILE             rolling_change_backup.f
           DGBROKER                       1
           DICTIONARY_LOAD_TIMEOUT        3600
           DICTIONARY_LOAD_WAIT           0
           DICTIONARY_PLS_WAIT_INIT       300
```

Die angezeigten Parameter können gegebenenfalls mit:

```
EXEC DBMS_ROLLING.SET_PARAMETER(scope IN VARCHAR2, -
name IN VARCHAR2, -
value IN VARCHAR2);
```

geändert werden.

2.) Mit dem folgenden Statement wird ein Plan der durchzuführenden Tests und Kommandos erstellt:

```
SQL> EXEC DBMS_ROLLING.BUILD_PLAN
```

In unserem Fall bekamen wir die Fehlermeldung:

```
SQL> EXEC DBMS_ROLLING.BUILD_PLAN
BEGIN DBMS_ROLLING.BUILD_PLAN; END;

*
ERROR at line 1:
ORA-45438: database is not in mounted mode
ORA-06512: at "SYS.DBMS_ROLLING", line 16
ORA-06512: at line 1
```

Der Grund für diese Meldung war, dass die Standby Datenbank im READ ONLY Modus lief (wegen ADG)

Nach Starten der Standby-Datenbank in den Mount Status konnte auf das BUILD_PLAN erfolgreich wieder gestartet und durchlaufen werden,

Überprüfen des Planes:

Das folgende SELECT_Statement zeigt alle Schritte des ermittelten Planes an:

```
SELECT instid, target, phase, description FROM DBA_ROLLING_PLAN;
```

```
INSTID TARGET     PHASE        DESCRIPTION
------ ---------- ------------ ------------------------------------------------------------
     1 prod       START        Notify Data Guard broker that DBMS_ROLLING has started
     2 stdby      START        Notify Data Guard broker that DBMS_ROLLING has started
     3 prod       START        Verify database is a primary
     4 prod       START        Verify MAXIMUM PROTECTION is disabled
     5 stdby      START        Verify database is a physical standby
     6 stdby      START        Verify physical standby is mounted
     7 stdby      START        Verify future primary is configured with standby redo logs
     8 prod       START        Verify server parameter file exists and is modifiable
     9 stdby      START        Verify server parameter file exists and is modifiable
   ...
   ...
   ...
    83 prod       FINISH       Notify Data Guard broker that DBMS_ROLLING has finished
    84 stdby      FINISH       Notify Data Guard broker that DBMS_ROLLING has finished
    85 stdby      FINISH       Restore Supplemental Logging

85 rows selected.
```

3.) Jetzt kann der Plan ausgeführt werden:

```
SQL> EXEC DBMS_ROLLING.START_PLAN;

PL/SQL procedure successfully completed.
```

Entgegen unseren Erwartungen brachte die Ausführung des unter 2.) erstellten Plans leider nicht immer spezifische Meldungen bei fehlgeschlagenen Tests.

So bekamen wir eine z.B. eine Exception, die im „Database PL/SQL Packages and Types Reference“ für das DBMS_ROLLING Package gar nicht erwähnt war:

```
ORA-45489: required condition was not satisfied.
```

Leider gaben die Support-Seiten auch keine Auskunft darüber, was die Ursache dieser Meldung sein könnte. Letztlich stellte sich heraus, dass es nicht reichte, dass der Fast-Start-Failover Server ‚disconnected‘ war, sondern er musste tatsächlich im Broker mit:

```
DGMGRL> disable fast_start failover
```
und
```
DGMGRL> stop observer all
```

deaktiviert sein.

Danach konnte das DBMS_ROLLING.START_PLAN erneut und ohne weitere Maßnahmen noch einmal gestartet werden.

Nach dem START_PLAN kann man sehen, dass die Standby-Datenbank in eine Logical Standby umgewandelt wurde:

```
SQL> SELECT DB_UNIQUE_NAME,OPEN_MODE,DATABASE_ROLE FROM V$DATABASE;

DB_UNIQUE_NAME   OPEN_MODE    DATABASE_ROLE
---------------- ------------ -------------
stdby            READ WRITE   PRIMARY
```

Wenn dieser Schritt durchlaufen ist, kann die frühere Standby-Datenbank mit dem üblichen Verfahren gepatcht werden.

Das heißt, Datenbank und Listener müssen heruntergefahren werden und dann wird mit ‚opatch‘ gepatcht und gegebenenfalls ‚datapatch‘ aufgerufen.

Da die Datenbank bei Aufruf des DBMS_ROLLING.START_PLAN Packages in eine

‚Transient Logical Standby' Datenbank umgewandelt wird, muss nach dem Patchen der log-apply auf der Standby-Seite mit dem folgenden Statement gestartet werden:

```
ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

Nach diesen Schritten kann sollen die Rollen der Datenbanken getauscht werden, um die ursprüngliche Standby Datenbank in eine Primary Datenbank umzuwandeln und die frühere Primary- in eine Standby Datenbank.

4.) Jetzt ist die Standby-Seite gepatcht. Der nächste Aufruf, mit dem die Datenbank-Rollen getauscht werden, muss noch von der Primary Datenbank erfolgen:

```
EXEC DBMS_ROLLING.SWITCHOVER
```

Falls das Switchover probiert wird, bevor der Logical Apply erfolgt ist (s. Punkt 3.) , bekommt man die folgende Fehlermeldung:

```
BEGIN dbms_rolling.switchover; END;

*
ERROR at line 1:
ORA-45427: logical standby Redo Apply process was not running
ORA-06512: at "SYS.DBMS_ROLLING", line 89
ORA-06512: at line 1
```

Das ist aber unkritisch, da man nach dem:

```
ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
```

den Switchover-Aufruf einfach wiederholen kann.

Wichtig: Wenn DBMS_ROLLING mit einer Broker Konfiguration genutzt wird, **muss** für das Switchover die Prozedur DBMS_ROLLING.SWITCHOVER genutzt werden! Das Switchover darf nicht aus dem Broker kommen!

Jetzt ist der Zeitpunkt gekommen, wo die frühere Primary Datenbank gepatcht werden kann.
Da das ‚datapatch' schon auf der neuen Primary-Seite ausgeführt wurde, werden die vom ‚datapatch' durchgeführten Änderungen beim Switchover auf die neue Standby Datenbank repliziert. Daher muss ‚datapatch' hier nicht mehr aufgerufen werden.

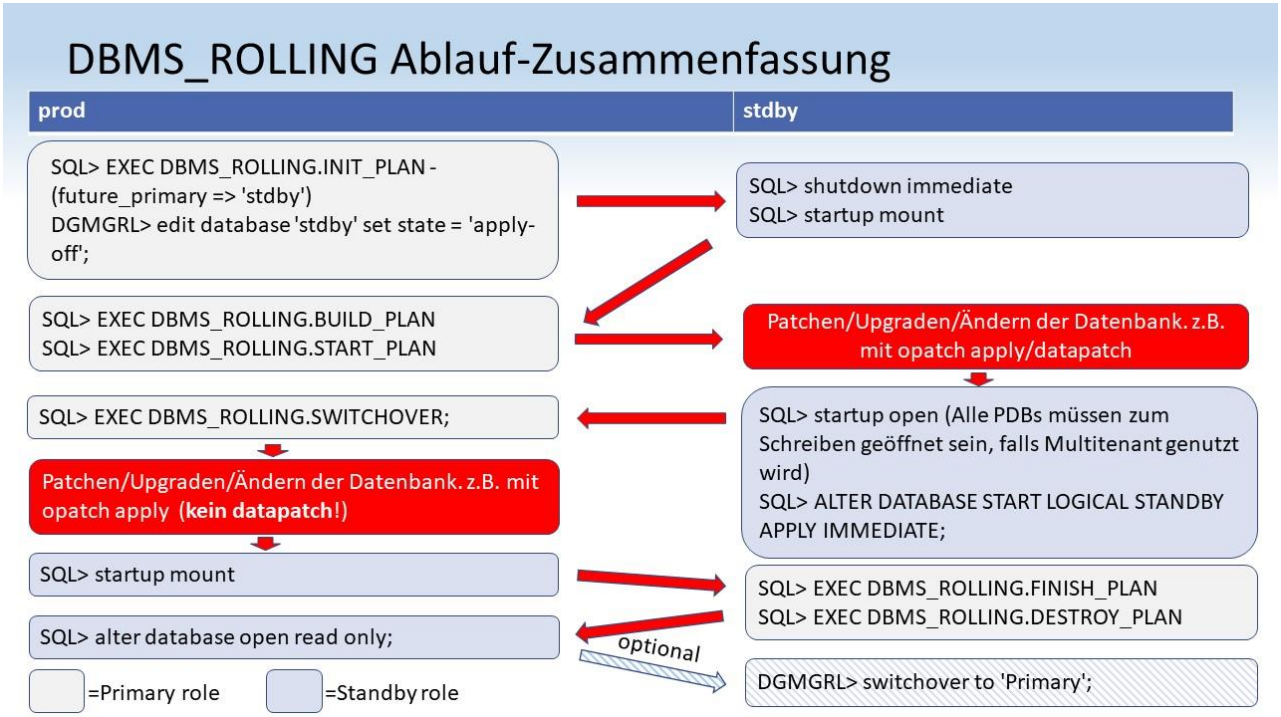
5.) Nach dem Switchover ist die alte Primary Datenbank zur TRANSIENT LOGICAL STANDBY umgewandelt worden. Damit die nächste Prozedur, DBMS_ROLLING.FINISH_PLAN , sie wieder in eine Physical Standby umwandeln kann, muss die Datenbank in den Mount-Status versetzt werden. Danach wird:

```
EXEC DBMS_ROLLING.FINISH_PLAN
```

ausgeführt und die Datenbank wieder im READ ONLY Modus geöffnet:

```
SQL> ALTER DATABASE OPEN READ ONLY;
```

Damit sind alle unbedingt notwendigen Prozeduren durchlaufen, die man für den Prozess das Rolling Patching braucht.
Hier noch einmal ein Überblick über die Reihenfolge der Aufrufe. Details zum Patching wurden weggelassen. In der Kopfzeile der folgenden Abbildung befinden sich die Namen der beteiligten Datenbanken. Die Primary- und die Standby-Role sind, wie in der Legende zu sehen, farblich unterschiedlich gekennzeichnet:



6.) Damit für künftige Patches ein neuer ROLLING_Plan erstellt werden kann, sollte als letzte Prozedur ein:

EXEC DBMS_ROLLING.DESTROY_PLAN

ausgeführt werden.

7.) Für den Fall, dass während des Patchens sowohl die Primary- als auch die Standby-Datenbank durch jeweils eine eigene Standby Datenbank abgesichert werden, kann man die Zuordnung dieser Standby-Datenbanken mit der

DBMS_ROLLING.SET_PARAMETER

vornehmen. Diese Konfiguration haben wir nicht getestet.

8.) Falls während des ROLLING_Patchings der ursprüngliche Zustand der Konfiguration wieder hergestellt werden soll, kann dies mit der Prozedur

DBMS_ROLLING.ROLLBACK_PLAN

erreicht werden. Dies ist aber nur möglich, solange noch kein

```
DBMS_ROLLING.SWITCHOVER
```

aufgerufen wurde.

Fazit

Das DBMS_ROLLING Package unterstützt das Rolling Patching sehr effektiv und nimmt dem Administrator viele manuelle Schritte ab. Es ist weitgehend fehlertolerant, indem die Prozeduren dieses Packages sich bei Fehlbedienungen nach Beseitigung der Fehlerursachen, wie z.B. falschem MOUNT-Status der Datenbank oder nicht erfolgtem LOGICAL APPLY wieder aufrufen ließen, ohne dass vorherige Schritte invalidiert wurden. Fehler werden sowohl in Tabellen, als auch im Alert.log der Datenbank mitprotokolliert. In 2 Fällen waren sie nicht sehr aussagekräftig, aber in den Meisten Fällen bekommt man Hinweise darauf, warum ein Aufruf fehlgeschlagen ist.

Kontaktadresse:

Nikolaus Eichler
Selbständiger Berater
Am Nachbarsberg 18
D-42781 Haan

Telefon: +49 (0) 160-96227899
E-Mail nik.eichler@gmail.com