Migration einer 7 TB Datenbank von AIX nach Linux

Bernd Patolla
CONCORDIA Versicherungen AG
Luzern, Schweiz

Schlüsselworte
Datenbank-Migration, Cross-Plattform, Cross-Endian, XTTS.

Einleitung
Im Rahmen einer Plattform-Migration von AIX nach Red Hat Linux 7.4 mussten mehrere zentrale Oracle 12c Datenbanken migriert werden. Die Datenbanken waren von einigen GB bis zu 7 TB gross.
In diesem Manuskript beschreibe ich die Vor- und Nachteile der verschiedenen betrachteten Migrationsmethoden wie Export/Import, manueller TTS, rman backups and XTTS.
Der Schwerpunkt des Vortrages beinhaltet das verwendete Verfahren XTTS (-> MOS 2005729.1).
Zuerst stelle ich die Voraussetzungen und die Konfiguration des Verfahrens vor (Phase 1).
Anschliessend gehe ich auf das verwendete Migrations-Setup (NFS), die initiale Phase (Phase 2) als auch die Zwischenschritte (Roll-Forward-Phase 3) ein, welche mehrfach wiederholt wurde.
Die finalen Schritte (Phase 4, finale Migration und 5, Abschlussarbeiten) werden im Detail erklärt.
Dabei wird auch auf die Objekte eingegangen, welche nicht mit dem Verfahren transportiert werden (z.B. Sequencen, SQL-Profiles und SQL-Baselines) und wie diese kopiert werden können.

Unsere Umgebungen
Jahrelang haben wir unsere Oracle 12c Datenbanken auf einer virtualisieren AIX-Umgebung (PowerVM) betrieben. Als Server-Hardware setzten wir dabei zwei IBM Power P870 mit insgesamt 24 Cores für den Datenbank-Pool ein. IBM Features wie CPU Sharing und Overprovisioning wurden intensiv genutzt. Mit diesem Setup konnten wir eine gute Performance der Datenbanken garantieren.
Da der Lieferant unserer Kern-Applikation bestimmte Funktionalitäten unter IBM AIX nicht mehr unterstützte und die Plattform AIX für Neukunden nicht mehr zertifizierte, haben wir ein Projekt für die Evaluation einer neuen System-Plattform gestartet.
Unsere Wahl fiel auf Red Hat Enterprise Linux auf HP X86-Server mit VMware für die Virtualisierung.
Neben der Änderung der Server-Plattform fanden auch weitreichende Änderungen an dem gesamten Plattform-Stack statt. Das Storage-System für die Linux-Umgebung wurde neu evaluiert (SSD only) und installiert. Als Diskverwaltung für Datenbanken wird unter Linux Oracle ASM mit Raw Device Mapping benutzt, unter AIX wurden alle DB-Files im Filesystem abgelegt.
Die Datenbank-Migrationen sollten dann im Rahmen eines normalen Release (mit Update der Applikationen) erfolgen. Daher war der mögliche Zeitrahmen für die Datenbank-Migration sehr eng gesteckt.
Kurz zusammengefasst die Eckpunkte unserer Datenbank-Migration:

- von AIX nach Linux (Cross-Endian)
- von Filesystem nach ASM
- Dauer ca. 1 Stunde (während Release)

Mögliche Migrations-Verfahren

Anforderung	Exp/Imp	DataGuard	GoldenGate	TTS	CTAS	XTTS
CrossEndian	Ja	Nein	Ja	eingeschränkt	Ja	Ja
FS -> ASM	Ja	Ja	Ja	Nein	Ja	Ja
Zeit <= 1Std	Nein	Ja	Ja	Nein	Nein	Ja
Komplexität	gering	mittel	mittel	hoch	gering	mittel
Lizenz notwendig?	Nein	Ja	Ja	Nein	Nein	Nein

Export / Import
Unterstützt Cross-Endian-Migrationen. Aufgrund der erwarteten langen Laufzeit des Verfahrens wurde dies für den vorgestellten Fall nicht näher angeschaut.
Jedoch haben wir Export/Import für kleine Datenbanken (< 1TB) genutzt.

DataGuard
Unterstützt keine Cross-Endian-Migration zwischen AIX on Power und Linx on X86 (-> MOS 413484.1). Ist u.U. lizenzpflichtig.

GoldenGate
Unterstützt eine Konfiguration mit AIX als Extract und Linux als Replikat (Cross-Endian). Wir hatten GoldenGate temporär für den Datentransport von der ERP-Applikation unter AIX in das DWH unter Linux im Einsatz gehabt. Die notwendige Downtime für den Wechsel der Datenbank ist sehr gering. Jedoch ist GoldenGate lizenzpflichtig.

Transportable Tablespaces
Hoher manueller Aufwand und lange Unterbruchszeit für den Transport der Datafiles bei grossen Datenbanken zwischen den Servern. Kann die Datafiles nicht direkt vom Filesystem nach ASM kopieren (Zwischenschritte sind notwendig). Lizenzen sind keine notwendig.

CTAS
Unterstützt Cross-Endian-Migrationen. Die Dauer der Migration ist sehr lange. Alle Indizes (bei unserer Applikation ca. 5000) müssen anschliessend neu erzeugt werden, dadurch noch längere Laufzeit.

XTTS
Das Verfahren **C**ross Plattform **T**ransportable **T**able**S**paces ist sehr gut in der Note MOS 2005729.1 (für Oracle 12c) resp. MOS 1389592.1 (für Oracle 11) beschrieben und basiert auf Transportable Tablespaces.
Mittels unvollständiger Backups und einem konsistenten inkrementellem finalen Backup können auch grosse Datenmengen in relativ kurzer Zeit migriert werden. Das Toolkit wird von Oracle auf den o.a. Seiten bereitgestellt.

Varianten-Entscheid
Aufgrund der möglichen kurzen Unterbruchs-Zeit (in der finalen Phase) und der Tatsache, dass keine zusätzlichen Lizenzen notwendig sind, haben wir uns für das XTTS-Verfahren entschieden. Durch die Verwendung von rman für den Transport ist auch eine direkte Migration von Filesystem-Datafiles in ASM möglich.
Ich gehe im Folgenden nicht auf die einzelnen Kommandos ein sondern stelle eher unsere Erfahrungen mit dem Toolkit in den Vordergrund.

Migrations-Verfahren
Eine Migration mit XTTS unterscheidet folgende Phasen:

1. Installation und Konfiguration
2. initiale Migration
3. Roll-Forward (mehrfach)
4. finale Migration
5. Abschlussarbeiten

Phase 1: Installation & Konfiguration
Das Toolkit für XTTS kann von der Oracle Support Seite aus den o.a. Dokumenten herunter geladen und auf einem System ausgepackt und konfiguriert werden. Erst nach der Konfiguration sollte das gesamte Verzeichnis auf das andere System kopiert werden. Empfohlen wird der gleiche Pfad auf dem Quell- und Ziel-System.
Für den Datenaustausch zwischen den beiden Systemen wird der Austausch von SSH-Keys empfohlen (ssh und scp ohne Passwort-Eingabe).
Die Ziel-Datenbank muss vorab ("leer") erstellt werden und den gleichen Zeichensatz wie die Quell-Datenbank haben. Ebenso sollten die User, welche in den zu transportierenden Tablespaces Daten haben, schon jetzt angelegt werden.
Backup-Destination: Für das Ziel der Backups auf Disks (XTTS unterstützt kein Tape) haben wir ein grosses Filesystem auf dem Ziel-System erstellt und dies per NFS an das Quell-System exportiert. Eine bessere Alternative ist das Quell-System als NFS-Server zu definieren, so dass die Schreibzugriffe lokal erfolgen können. Dementsprechend sollte auf beiden Datenbanken der Event "`10298 trace name context forever, level 32`" gesetzt werden. Dies schaltet die NFS-Checks aus. Der Mount-Point sollte auf beiden Systemen gleich heissen. Die beiden Variablen `backupformat` und `stageondest` zeigen auf diesen Mountpoint.
Auch kann für NFS die Parallelität der Backups (`parallel`, `rollparallel`) erhöht werden. In unserem Environment (10Gbit, Jumbo-Frames) haben wir mit einer Parallelität von 8 sehr gute Erfahrungen gemacht. Dazu muss auch in rman die Parallelität für den `device type disk` entsprechend gesetzt werden:

```
CONFIGURE DEVICE TYPE DISK PARALLELISM 8 BACKUP TYPE TO BACKUPSET;
```

XTTS arbeitet auf der Basis von Tablespaces. Daher muss vor dem Einsatz überprüft werden, ob die zu transportierenden Daten in "abgeschlossenen" Tablespaces sind (Self-Contained Set of Tablespaces):

```
EXECUTE DBMS_TTS.TRANSPORT_SET_CHECK ('TS1,TS2', TRUE);
```

mit anschliessender Überprüfung:

```
SELECT * FROM TRANSPORT_SET_VIOLATIONS;
```

Wenn NFS als "Transport"-Mechanismus für die Daten benutzt wird, sollte der Parameter `metatransfer` auskommentiert werden. Ansonsten wird das XTTS-Toolkit die Backups vom Quell- auf das Zielsystem mittels scp kopieren. Im gleichen Kontext sollten die Parameter `destuser` und `desthost` nicht gesetzt werden.
Empfehlenswert ist auch den Patch zum Bug 25670970 (`ORA-600 [25027] at insert into securefile lob after cross endian XTTS`) sowohl auf Quell- als auch Ziel-System einzuspielen.
Die Anzahl möglicher Backups im Change Tracking File sollte ensprechend erhöht werden: Jedes Migrations-Backup wird im BCT mitprotokolliert (-> MOS 1192652.1) und daher wird die Default-Anzahl von 8 Backups nicht ausreichend sein.
Weitere wichtige Parameter sind:

- `asm_home` (der Ziel-Plattform)
- `asm_sid` (der Ziel-Plattform)
- `plattformid` (der Quell-Plattform, da die Umwandlung auf der Ziel-Plattform erfolgt)

- `desttmpdir`

Alle Parameter in der `xtt.properties`-Datei sind gut dokumentiert. Ich empfehle zuerst die Kommentare zu den Parametern zu studieren.

Phase 2: Initiale Migration

Vor jedem Backup und Restore mit XTTS sollte darauf geachtet werden, dass der `default device type disk` gesetzt ist. Der Backup funktioniert zwar mit `default device type tape`, jedoch landen die Backups dann irgendwo und sind für den Restore-Prozess nicht sichtbar.
Auf der Quell-Datenbank wird ein Full-Backup der zu migrierenden Daten (Tablespaces) erstellt (Level 0 Backup). Da das Backup online erfolgt und ohne Backup-Modus der Tablespaces, ist es ein inkonsistentes Backup. Die erste Phase sollte daher in einer möglichst ruhigen Zeit auf der Quell-Datenbank durchgeführt werden.
Wir haben die erste Migrations-Phase ca. eine Woche vor dem finalen Schritt an einem Sonntagabend durchgeführt. Man muss mit der gleichen Zeitdauer wie bei einem Full Backup rechnen (bei gutem NFS- und Netzwerk-Setup).
Nach dem Backup sind u.a. die folgenden Files erzeugt worden:

- `xttplan.txt`: Liste von Tablespaces mit SCN vom Backup als auch der Filenummern der Datafiles
- `incrbackups.txt`: Liste der erstellten Backup-Files
- `txtbkupmap.txt`: Mapping zwischen Filenummer und Backup-File
- `xttnewdatafiles.txt`: Liste von Filenummer zu neuem Datafile

Den Restore auf der Zielplattform haben wir am Montagvormittag durchgeführt, da auf dem Ziel-System keine andere DB aktiv ist.

Phase 3: Roll Forward (wiederholend)

Jede Nacht (nach der Batch-Verarbeitung) haben wir dann die incrementellen Backups durchgeführt.
Um die Phase 2 zu automatisieren, habe ich kleine Shell-Scripts für die Quell- und Zielsysteme geschrieben, die sich mittels Kontroll-Files (und ssh/scp) koordinieren.
Der `default device type` wird als `disk` definiert (da unsere Backups standardmässig auf Tape gehen ist dies notwendig).
Danach findet die eigentliche Migrations-Phase 2 mit dem inkrementellen Backup statt.
Direkt nach dem Backup wird der Device Type wieder auf Tape umgestellt.
Die Files

- `xttplan.txt`
- `tsbkupmap.txt`
- `incrbackups.txt`

werden in das XTTS-Verzeichnis auf dem Zielsystem mit scp kopiert. Jetzt können die Arbeiten auf dem Ziel-System starten.
Zum Schluss lasse ich noch die neue "From"-SCN für den nächsten Lauf berechnen.
Auf dem Ziel-System wird der `default device type` auf `disk` umgestellt und die Recovery gestartet. Anschliessend werden die Koordinationsfiles sowohl lokal als auch remote gelöscht.
Die Phase zwei dauert ungefähr solange wie ein normaler inkrementeller Backup. In unserer Umgebung lief der Backup der Phase zwei weniger als 10 Minuten.
Wir haben die Phase 2 von Dienstag bis Freitag insgesamt viermal durchlaufen.

Phase 4: Finale Migration

Direkt vor dem Start der finalen Phase haben wir schon die SQL Profile und die SQL Baselines mittels Staging-Tabellen und einfachem Export/Import in die Ziel-Datenbank kopiert und aktiviert.

Das Release startete dann mit dem Stop der Applikation(en). Anschliessend (um Fehlermeldungen beim TTS-Import zu vermeiden) haben wir alle Grants gelöscht (alternativ können auch alle gegranteten User in der Ziel-Datenbank vorab erstellt werden).
Eine Applikation hat AQs benutzt, so dass wir die Queues in der Quell-DB gestoppt haben.
Für die letzte Phase (um einen konistentes Backup zu erstellen) werden die Tablespaces in den ReadOnly Modus gesetzt. Anschliessend starteten wir das XTTS-Tool.
Bei unseren Migrationen war der im Backupset enthaltene TTS-Dump nicht lesbar. Wir haben daher einen separaten TTS-Export durchgeführt (`expdp … transport_tablespaces=…`) und das Dumpfile auf das Zielsystem kopiert. Für das XTTS-Tool müssen die drei Files

- `xttplan.txt`
- `tsbkupmap.txt`
- `incrbackups.txt`

in das TMPDIR auf dem Zielsystem kopiert werden.
Auf dem Zielsystem erfolgt dann der letzte Recovery-Schritt (`xttdriver --resincrdump`). Der TTS-Import wird dabei aber aufgrund des oben aufgeführten Problems nicht funktionieren, jedoch wird das File `xttplugin.txt` erstellt. Das File haben wir als Parameter-File umgeschrieben und zeigen dabei auf unser eigenes TTS-Dump-File.
Beim Import gab es dann aber trotzdem ein paar Fehler: So konnten wir zwei Tabellen nicht importieren (Ursache unbekannt). Diese haben war mittels expdp/impdp migriert.
Ist der TTS-Import "erfolgreich" abgeschlossen, sollte mit `rman` ein `validate tablespace` durchgeführt werden.
Der spannendste Augenblick ist das Öffnen der Tablespaces in den Read/Write-Modus.

Phase 5: Abschlussarbeiten
Nun können die Grants, Sequenzen, … wieder erstellt werden. Nicht vergessen: Die Quotas für die User auf den "neuen" Tablespaces!
Die SQL-Profile und Baselines können (auch schon früher) aus den Staging-Tabellen aktiviert werden.
Wir haben neben dem obigen `rman validate` auch noch einen sogenannten Kennzahlen-Vergleich laufen lassen. Als Basis diente uns ein Vergleich zwischen unserem DWH und der Applikation (ob die Daten richtig transportiert und verarbeitet wurden). Wir haben den Applikations-Teil genommen und hatten so ca. 57 Tests, welche wir sowohl auf der AIX- als auch der Linux-Seite zum gleichen Zeitpunkt haben laufen lassen. Der Kennzahlen-Test konnte denn auch erfolgreich abgeschlossen werden.

Zeitablauf Migration:

Vor Migration:	AIX: - #Datafiles so erweitert, dass keine zusätzlichen während der Migationsphase notwendig werden (Vorsichtsmassnahme). LX: - Schema-User angelegt, default tablespace users
22.04. 21:55	Phase 2: Start Backup AIX, Dauer bis 23.04. 01:50
23.04. 08:53	Phase 2: Start Restore Linux, Dauer bis 10:33
24.04. 00:30	Phase 3: Start Backup AIX, Dauer ca. 7min (incrementelles Backup)
24.04. 00:45	Phase 3: Restore Linux, Dauer < 1min
25.04. 00:30	Phase 3: Start Backup AIX, Dauer ca, 7min
25.04. 00:45	Phase 3: Restore Linux, Dauer ca. 2min
26.04. 00:30	Phase 3: Start Backup AIX, Dauer ca. 8min
26.04. 00:45	Phase 3: Restore Linux, Dauer ca. 1min
27.04. 00:30	Phase 3: Start Backup AIX, Dauer ca. 8min
27.04. 00:45	Phase 3: Restore Linux, Dauer ca. 1min
27.04. tagsüber	SQL Profile, SQL BaseLines in Staging-Tabellen, AIX exportiert, LX importiert und Destaged
27.04. 17:00	Stop Applikation AIX, Export kleinere Schemata (separate Applikation)
27.04. 17:05	AIX: - Rechte (Grants) löschen, - AQs stoppen, - rman umkonfigurieren (default device type disk) - Tablespaces auf ReadOnly gesetzt - export Sequences
27.04. 17:15	AIX: - Start Phase 4: `xttdriver --bkpexport`, Dauer ca. 30min (wg TTS-Export) LX: - Import kleinere Schemata
27.04. 17:45	AIX: - TTS export gestartet LX: Phase 4: - `xttdriver --resincrdump`, Dauer wenige Minuten. - `xttplugin.txt` bearbeitet und als Parameter-File "erstellt" - `impdp xttplugin.par` gestartet: Fehlermeldung: 2 Tabellen können nicht importiert werden, viele Folgefehler (Indices, Constraints)
27.04. 18:11	AIX: - export der 2 fehlenden Tabellen, kopieren Dump-File nach LX LX: - TTS import fertig - Import der zwei fehlenden Tabellen - Import Sequences
27.04. 18:30	LX: - Tablespaces auf ReadWrite gesetzt - Tablespace-Quota definiert & default Tablespace User gesetzt. - Queues gestartet
27.04. 18:35	Migration abgeschlossen!
27.04. 18:45	Start Kennzahlen-Test. Applikation bleibt wg. Datenkonsistenz und möglicher Updates noch gestoppt.

Empfehlungen:

- Vor der Migration die Anzahl Datafiles so erhöhen, dass während der Migration keine Erweiterung notwendig wird. XTTS soll zwar zusätzliche Datafiles handlen können, das habe ich aber nicht getestet.
- SSH-Key-Austausch zwischen dem Quell- und Ziel-System
- NFS als "Transport"-Plattform hat sich bewährt. SCP zwischen AIX und LX ist sehr langsam (max 30MB/sec). Netzwerk (-Parameter) genau anschauen: Jumbo-Frames, sonstige Last, …
- Parallelität definieren. Wir haben mit dem Faktor 8 sehr gute Erfahrungen gemacht (gleiche Performance wie bei Full-Backups mit 4 Channels).
- Schema in der Ziel-Datenbank vor der Migration anlegen.
- Tablespace-Self-Containing überprüfen
- SQL Baselines, SQL Profile, … können schon oft vor der vierten Migrationsphase transportiert werden.
- vor jedem Schritt rman-Konfiguration, speziell `default device type` überprüfen und setzen.
- Auf sonstige Daten, welche nicht in den zu transportierenden Tablespaces liegen, achten!
- Test-Migrationen mit Kopien der produktiven Daten durchführen und alle (erzeugten) Daten/Files sammeln. Das kann im Notfall (selber erlebt) sehr hilfreich sein.
- die Files `xttplan.txt`, `tsbkupmap.txt` und `incrbackups.txt` anschauen und studieren. Bei einer vorhergehenden Migration ist die letzte Phase auf der Linux-Seite leider abgebrochen und ich musste das File für den TTS-Import (`xttplan.txt` resp. `xttplan.par`) manuell erstellen.
- Phase 2 möglichst in einer ruhigen Zeit durchführen, es wird schon eine merkbare Last auf der Datenbank erzeugt.
- Phase 3 automatisieren: Der Aufwand ist gering und die Fehlerquote wird erheblich reduziert.

Kontaktadresse:

Bernd Patolla
CONCORDIA Versicherungen AG
Bundesplatz 15
CH 6002 Luzern

Telefon:	+41 41 228 09 19
Fax:	+41 41 228 02 07
E-Mail	bernd.patolla@concordia.ch
Internet:	www.concordia.ch