Von 12.1 NonCDBs zu 18c Multitenant - ein Erfahrungsbericht

Robert Ortel
Hypoport Systems GmbH
Berlin

Schlüsselworte
NonCDB, CDB, PDB, container, pluggable, database, migration, upgrade, 12c, 18c, 12.1.0.2, Konsolidierung, Erfahrungen, Hochverfügbarkeit

Einleitung

In einem Umfeld einer gewachsenen Infrastruktur entstand die Idee zur Vereinheitlichung und Vereinfachung und Konsolidierung der vorhandenen Datenbank-Infrastruktur. Die Saat dazu hatte die Multitenant-Option gesät.
Ich werde unsere ehemalige Infrastruktur kurz vorstellen und erläutern, welche Überlegungen mich von Multitenant überzeugt haben. Insbesondere die Erfahrungen rund um die Upgrades und Migrationen, sowie des Betriebes über die ersten Wochen, möchte ich vorstellen und wie ich auch ohne RAC eine möglichst hohe Verfügbarkeit erreichen (wollen).

Motivation

Über viele Jahre entstanden und gingen Datenbanken, bereicherten neue Server die Infrastruktur und wurden unterschiedliche Lizenzen für verschiedene Zwecke und Situationen eingesetzt. Auch wenn die Bedeutung der Datenbanken keinesfalls schrumpfte, stand die vorhandene Hardware und die Anforderungen an diese in keinem Verhältnis mehr.
Benötigtes RAM, lokaler Plattenplatz auf den physischen Servern (der mit jedem Patch Bundle weiter wächst) und eine selbstgemachte Limitierung, welche aus den eigenen Installationsmechanismen entsteht (2x gleicher DB-Name pro Server ist nicht möglich) begrenzten die Konsolidierung und ohnehin war Konsolidierung bisher kaum ein Thema für unsere Oracle DBs. So kam mir eine Konsolidierungsidee in den Sinn, welche mit der Multitenant Option die Anzahl der Server reduzieren und gleichzeitig für eine deutlich bessere Ausnutzung der Lizenzen sorgen sollte.
Primär die technische Umsetzung soll hier dargestellt werden.

Die alte Infrastruktur

Verteilt auf 16 physischen Servern liefen 76 DB Instanzen, verteilt auf 2 Standorte. Wegen Oracle's Lizenzpolitik setze ich nur physische Server exklusiv für die Oracle DBs ein. All diese DB Instanzen der beiden Editionen liefen unter 12.1.0.2 als Single Instance und NonCDB. Weil diese sehr stabil ist und – nach meiner Erfahrung - nur im Falle von Hardware-Problemen den Dienst versagen, genügte die Single Instance bisher unseren Anforderungen. Hardware-Probleme erfordern dann zwar Hand anzulegen, doch dank verfügbarer Ersatzhardware und schnellen Mechanismen zur Inbetriebnahme dieser Ersatzhardware toleriere ich diese möglichen Ausfallzeiten. Eine Migration zu Singletenant war bisher wegen Problemen in Oracle's Migrations-Skript (`noncdb_to_pdb.sql`) bei der Migration von 11.2.0.4 nach 12.1.0.2 vertragt worden.
Auf jedem der physischen Server läuft ASM und stellt die LUNs von Shared Storages jeweils als 3 Diskgruppen pro Datenbank zur Verfügung. Diese 3 Diskgruppen pro Datenbank speichern dann jeweils Datafiles, Logfiles & Dateien der Recovery Area. Auf produktiven Datenbanken sind diese Diskgruppen redundant über 2 Shared Storages ausgelegt (normal redundancy). Auf den Test und Entwicklungssystemen arbeiten ich bisher ohne Redundanz seitens ASM (external redundancy).
Nur für die Dateien des Backups verwende ich reguläre Dateisysteme, entweder gemounted per NFS oder als LUN von einem Shared Storage mit einem Dateisystem.

Durch den Einsatz von ASM auf jedem Server setze ich auch die Funktionalität von Oracle Restart überall ein.

Optimierungsidee

Die meisten der 16 Server waren bei weitem nicht ausgelastet. Ein großer Teil der Hardware war nur in Verwendung, weil neuer RAM, lokaler Plattenplatz und/oder die Limitierung der DB-Namen das verwenden eines neuen Servers nötig machte, welcher (vor allem für Test und Entwicklung) meist bereits existierte und nicht erst bestellt werden musste.
Mit der Idee hinter Multitenant wurde mir nach und nach klar, dass sich dies grundlegend ändern ließe. Die 16 Server könnte man auf 2 Server reduzieren (ohne Betrachtung der Ersatz-Hardware), welche Dank Blade-Server-Konzepten noch weniger Platz als manche der alten Server einnehmen. Diese 2 Server würden jedoch deutlich mehr RAM benötigen. Zusätzlich ließe sich über die 2 Standorte eine Replikation betreiben, welche neue Vorteile u.a. für die Hochverfügbarkeit bietet. Und nicht zuletzt bringt das Cloning von PDBs mit Multitenant neue Möglichkeiten für Entwicklung und Test mit sich.
Neben den Optimierungen rund um Platzbedarf im Serverrack, Stromverbrauch und Wartungsaufwand für die vielen DB Server, die Verbesserung der Hochverfügbarkeit und der neuen Features war auch eine Optimierung der Auslastung der Lizenzen drin.

Die neue Infrastruktur

Auf jedem der 2 neuen Server läuft nun jedoch nicht nur eine Container Database (CDB), sondern jeweils 2. Nur eine CDB ist quasi aktiv, konsumiert einen großen Anteil an RAM für SGA/PGA und lässt Pluggable Databases (PDBs) laufen. Die andere CDB ist zwar geöffnet, aber nur im restricted mode und nur mit sehr wenig SGA/PGA gestartet.
Kommt es zum permanenten Ausfall einer CDB kann die 2. CDB auf dem Server aushelfen und die PDBs nach einem Neustart (wg. Anpassung der SGA/PGA) aufnehmen, welche vorher in der 1. CDB liefen. Außerdem lassen sich so die Patching-Downtimes verkürzen, welche meist 2x im Jahr für Release Updates durchgeführt werden. Die 2. CDB kann man bereits patchen und muss dann nur die PDBs umhängen. Die Ausführung von `datapatch` geht dann wieder online.
Auf jedem der 2 neuen Server läuft wieder ASM und damit Oracle Restart. Jede CDB erhält wieder ihre üblichen 3 Diskgruppen für Datafiles, Logfiles und Recovery Area. Da sich alle PDBs dann den Platz für Logfiles und Recovery Area mit der CDB teilen, spart man so zusätzlich Plattenplatz ein, da sich nun auch alle eine Art Hochlastreserve teilen und nicht mehr jede DB ihre eigene Reserve bekommt. Jede PDB erhält nur noch eine Diskgruppe für ihre Datafiles. In dieser neuen Infrastruktur werden alle Diskgruppen redundant (normal redundancy) betrieben und von 2 Shared Storages versorgt. So schützt die Redundanz vor Datenfehlern und dem Ausfall von einem Storage.
Über die 2 Standorte wird in jedem einer der Server laufen und eine Ersatzhardware bereitstehen. Der aktive Server an jedem Standort wird per Dataguard eine Replikation auf den anderen aktiven Server am anderen Standort vollführen.
Wie von Oracle vorgesehen laufen unsere CDBs unter 18.3 mit lokalem Undo, also einem Undo-Tablespace in jeder PDB, um alle neuen Features von Multitenant (Flashback PDB, PDB Point-In-Time-Recovery, …) nutzen zu können.

Migrationspfad

Als Migrationpfad von 12.1.0.2 NonCDBs zu 18.3.0.0 Multitenant bieten sich viele Möglichkeiten. In meinem Fall ist Ausführungssicherheit bei der Migration wichtiger als die Kürze der Downtime, da ich nächtliche Downtimes vergleichsweise einfach erhalten kann, ich jedoch der einzige Oracle DBA bin und möglichst wenige Methoden möglichst sicher beherrschen will.

So fiel die Entscheidung auf die klassische Migration in 2 Phasen, analog zu dem, wie es z.B. auch Mike Dietrich[1] für das Upgrade von 12.1 nach 12.2 empfiehlt: 12.1 NonCDB → 18.3 NonCDB → 18.3 PDB. Alternativ kommt für einige Sonderfälle die Migration mit klassischem Datapump zum Einsatz.
Die 2-phasige Migration dauert für alle DBs, egal welcher Größe, nahezu gleich lang, weil nicht der gesamte Datenbestand kopiert werden muss. Das ist ein großer Vorteil, denn es gilt für mich auch Multi-Terabyte DBs zu migrieren. Die 2 Kernphasen dauern ca. 25 (12.1 NonCDB → 18.3 NonCDB) und 12 Minuten (18.3 NonCDB → 18.3 PDB). Zusammen mit allen manuellen Vorbereitungen (insb. Checks), Zwischenschritten (z.B. DST. Update) und Nachbereitungen (Anpassungen jeder PDB) benötige ich pro DB knapp eine Stunde.
Der Pfad mit Datapump kann bei kleinen DBs einen Geschwindigkeitsvorteil bieten und dient als Fallback für alle anderen DBs. Nur bei den wenigen großen DBs mit mehreren Terabytes steht diese Option dann quasi nicht zur Verfügung, weil sie viel länger bräuchte.

Checks vor der Migration
Im Zuge der Checks vor der Migration halte ich mich an verschiedene Empfehlungen. Auch wenn es keine Checklist für die manuelle Migration nach 18c NonCDB zu geben scheint, halte ich mich zumindest an die Checklist für die Migration nach 12.2 NonCDB: Doc ID 2173141.1. Auch wenn es Teil der Checklist ist, möchte ich auch die Ausführung von `dbupgdiag.sql` (siehe Doc ID 556610.1) hervorheben. Es liefert zum Zustand der DB vor dem Upgrade eine gute Zusammenfassung, welche insbesondere für den Support im Problemfall nötig ist. Nicht Teil der Checklist, dafür aber von Mike Dietrich[1 & 2] empfohlen und über Doc ID 884522.1 zu beschaffen, ist Oracle's Database Pre-Upgrade Utility, welches ich auch vor jedem Upgrade ausführe. Sind all diese Checks erfolgreich, starte ich das Upgrade.

Erfahrungen Migration - DBMS_PDB.CHECK_PLUG_COMPATIBILITY
Mit den ersten Tests für die Migration hielt ich mich an den Vorschlag von Mike Dietrich: 12.1 NonCDB → 18.3 NonCDB → 18.3 PDB. Während das Upgrade nach 18.3 NonCDB von Beginn an tadellos durch lief, gab es gleich beim ersten Versuch für den 2. Schritt Probleme. Denn vor der Migration zu einer PDB steht der Check mittels `DBMS_PDB.CHECK_PLUG_COMPATIBILITY` an, ob diese kompatibel ist. Doch in der Release-Version von 18.3.0.0 kommt es dabei immer zu einem gravierendem Fehler: `ORA-07445 [__intel_ssse3_rep_memcpy()+6471]`. Ein Ergebnis des Checks erhielt man nicht.
Durch einen Service Request konnte das als Bug 28502403 identifiziert werden, wofür es zwischenzeitlich bereits den Patch gibt. Jener beseitigt das Problem erfolgreich und ist bereits im Release Update 18.4 enthalten. 18.4 konnte ich bisher aber nicht testen.
Zwischenzeitlich hatte ich mir einen Workaround überlegt. Die Migration kann man auch entlang eines anderen Pfades vollziehen: 12.1 NonCDB → 12.1 PDB → 18.3 PDB. In 12.1.0.2 arbeitet der Aufruf `CHECK_PLUG_COMPATIBILITY` tadellos und so ist dies eine Alternative. Da ich jedoch Datenbanken mit beiden National Charactersets betreue (UTF16 und UTF8), sind in diesem Fall für den ersten Migrationsschritt 2 verschiedene 12.1 CDBs bereitzuhalten, denn 12.1 unterstützt nicht beide National Charactersets in einer CDB (im Gegensatz zu 18c). So muss je nach National Characterset in der NonCDB die passende CDB für die Migration gewählt werden.
Mit diesem Vorgehen hatte ich einige erfolgreiche Migrationen, aber ich stieß vereinzelt auch auf Probleme. Die Migration zu einer 12.1 PDB verursachte z.B., dass die View `SYS.USER_AUDIT_POLICIES` hinterher invalid war. Ein Neukompilieren beseitigte das Problem aber

[1] https://mikedietrichde.com/2017/03/08/converting-an-12-1-non-cdb-and-plug-it-into-an-12-2-cdb
[2] https://mikedietrichde.com/2018/09/21/improved-preupgrade-jar-for-oracle-12-2-and-18c

nicht. Weil ich diesen alternativen Migrations-Weg aber nach kurzer Zeit wieder verworfen habe, kann ich dazu keine Lösung angeben.
In einem anderen Fall gab es Probleme mit dem Patch-Stand bei `CHECK_PLUG_COMPATIBILITY`. Es gab mehrere `ROLLBACK SUCCESS` und `APPLY SUCCESS` Meldungen in der View `DBA_REGISTRY_SQLPATCH` zu dem gleichen Bundle Patch. Erst nach manuellem löschen aus `DBA_REGISTRY_SQLPATCH` (nach Rücksprache mit Support), so dass es nur noch max. einen Eintrag jeweils zu `APPLY` und `ROLLBACK` gibt, lief `CHECK_PLUG_COMPATIBILITY` erfolgreich durch.
Nach Verifikation des Patches gegen den `ORA-07445` beim `CHECK_PLUG_COMPATIBILITY` kehrte ich jedoch zum ursprünglichen Migrationspfad (12.1 NonCDB → 18.3 NonCDB → 18.3 PDB) zurück und verwende den bis heute in der Mehrheit der Migrationsfälle.

Erfahrungen Migration – Migration Enterprise Manager Repository DB
Das Upgrade meiner Enterprise Manager Repository DB war das einzige Mal, das die Phase 12.1 NonCDB → 18.3 NonCDB an einem Problem scheiterte. In der letzten Phase (108) brach das Upgrade mit Perl-Fehlern ab und scheiterte im Upgrade-Logfile an:

```
SELECT TO_NUMBER('NONUPGRADED_TABLEDATA') FROM SYS.V$INSTANCE
*
ERROR at line 1:
ORA-01722: invalid number
```

Zu dem Zeitpunkt versuchte ich auch das Upgrade über die 12.1 PDB nochmal, welches ebenfalls im Zusammenhang mit `NONUPGRADED TABLEDATA` scheiterte (ich habe leider den genauen Fehler nicht mehr). In MOS fand sich dann Doc ID 2279497.1, welches anweist, das in diesem Fehlerfall vor dem Upgrade 2 Skripte auszuführen sind, um dieses Problem zu beseitigen. Nach einem Rückfall auf einen Stand der DB vor dem Upgrade und der Ausführung dieser 2 Skripte lief das Upgrade 12.1 NonCDB → 18.3 NonCDB erfolgreich durch.

Erfahrungen Migration - Platz auf Diskgruppe
Nach dem Upgrade der Datenbank von 12.1 NonCDB zu 18.3 NonCDB steht die Migration zu einer PDB auf dem Plan. An dieser Stelle scheiterte ich auch 2x, weil ich nicht beachtete, dass im Zuge dieser Migration der Platzbedarf leicht steigt. In diesen 2 Fällen war die Diskgruppe für die Datafiles der jeweiligen DB bereits nahezu voll und so kam es direkt beim `CREATE PLUGGABLE DATABASE` zu einem `ORA-15041: diskgroup „…" space exhausted`. Als Folge dessen gilt die erzeugte PDB dann als inaktiv und kann nicht gestartet werden kann. Doch da bereits Änderungen an den Datafiles stattgefunden haben, lässt sich auch die alte NonCDB nicht mehr starten. So blieb mir in diesen Fällen nur ein Restore auf einen Stand vor dieser Migration.
Im Moment des `CREATE PLUGGABLE DATABASE`, welches von `noncdb_to_pdb.sql` aufgerufen wird, entsteht ein neues Tempfile unter neuem Pfad auf der Diskgruppe, während das alte Tempfile der NonCDB liegen bleibt. Dafür wird ein weiteres GB Platz benötigt. Ist dafür nicht genug Platz vorhanden, kommt es zu diesem Problem.

Erfahrungen Betrieb - Database Services unter 18c
In unserer Umgebung mit Oracle Restart kommen natürlich Database Services zum Einsatz. Das soll auch mit 18c weiterhin so sein.
Doch beim ersten Versuch, nach dem Upgrade unter 18c einen Service per `srvctl add service` anzulegen, kam es zum Fehler `CRS-2918: Authorization failure: operating system`

`group 'oper' does not exist on server '…'`. Gemäß Dokumentation[3] sei die OSOPER Rolle nur optional festzulegen, woraus ich schloss, dass dies ebenfalls für die Existenz der Betriebssystem-Gruppe `oper` gilt. Ein Service Request kam auch zu dem Schluss, dass dies ein Fehler in der Dokumentation sei.
Das reine Anlegen der Gruppe `oper` genügte um den Fehler aus der Welt zu schaffen, auch ohne dieser einen User zuzuweisen oder sie mit der Rolle OSOPER zu verknüpfen.

Erfahrungen Betrieb - wiederkehrende Wait-Spitzen
Nach kurzer Zeit des Betriebes einiger PDBs in einer CDB fiel auf, dass es alle 30 Minuten zu Wait-Spitzen von `Sync ASM Rebalance` und `enq: RC - Result Cache: Contention` kommt (siehe **Fehler! Verweisquelle konnte nicht gefunden werden.**).

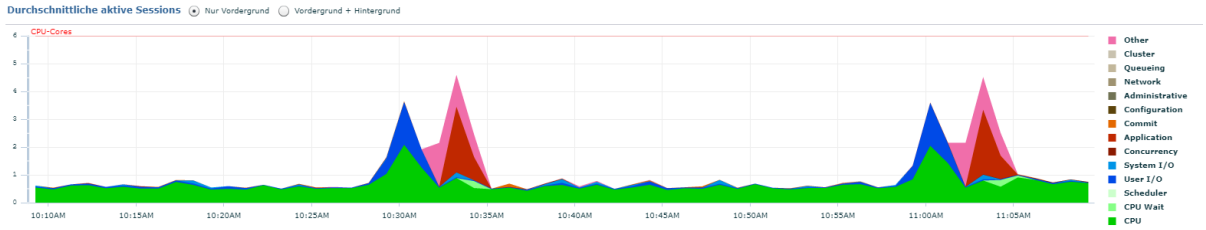

Abb. 1, Wait-Spitzen alle 30 Minuten

Diese werden vom Enterprise Manager Agent (bei uns Version 13.3.0.0.0) verursacht, welcher mit mehreren Queries die Tablespace-Usage ermitteln möchte und dazu u.a. `cdb_tablespace_usage_metrics` abfragt. Auch ein manuelles `select * …` auf dieser View verursacht eine kleinere Wait-Spitze für diese Waits.
Die PDBs, welche bisher migriert wurden und auf dieser CDB laufen, scheinen darunter für den Moment nicht zu leiden. Jedoch verursachen diese kaum signifikante Last und so ist es denkbar, dass diese Einschätzung bei höherer Last und komplexeren PDBs nicht mehr gilt. Für den Moment ist in dem SR dazu noch kein Bug geöffnet worden.

Erfahrungen Betrieb - Core Dump bei Diskgroup Dependancy Anpassung
Im Zuge der Migration der NonCDBs in PDBs auf einer CDB findet bei jeder Migration einiges an Wartungsarbeit rund um ASM Diskgruppen statt. Als Nebenwirkung davon wurde die Liste der Diskgruppen, von der die CDB-Resource in Oracle Restart abhängig zu sein scheint, um viele Diskgruppen erweitert, zu denen nun keinerlei Abhängigkeit mehr besteht. Zusätzlich gibt es in meiner Umgebung Diskgruppen mit identischen Namen, welche auf verschiedenen Servern aktiv sind. Auf dem einen Server mit den CDBs kam es so zu Namenskonflikten mit Diskgruppen in der Abhängigkeitsliste und Diskgruppen von NonCDBs, welche zu migrieren sind. Auch bei einem Neustart der CDB verursacht diese, zu lange Liste Probleme, weil vermeintlich benötigte Diskgruppen nicht online sind.
Ein Aufruf von `srvctl modify database` Zwecks Anpassung/Kürzung des Diskgruppen-Liste endete aber in einem Core Dump wegen eines `SIGSEGV` in `libhasgen18.so`.

[3] https://docs.oracle.com/en/database/oracle/oracle-database/18/cwlin/standard-oracle-database-groups-for-database-administrators.html#GUID-0A789F28-169A-43D6-9E48-AAE20D7B0C44

```
[oracle@sol044 ~]$ srvctl modify database -d CDB1SOL044 -diskgroup
DG_DATA_[…],[…]_MIS_DS
#
# A fatal error has been detected by the Java Runtime Environment:
#
# SIGSEGV (0xb) at pc=0x00007f5396ca29c8, pid=18219, tid=0x00007f53d2a83700
[…]
# Problematic frame:
# C [libhasgen18.so+0x1119c8] clsrteAddListArg+0x270
[…]
#
```

Wie sich in einem SR zeigte verträgt `srvctl modify database` für die Liste mit Diskgruppen keine sonderlich lange Liste: zwischen 8 Diskgruppen mit 150 Zeichen und 9 Diskgruppen und 167 Zeichen liegt die Grenze, ab der es zu diesem Fehler kommt. Die Bugs zu diesem Problem lauten 24590494, 28642469 und 27588725. Bisher (Stand 14.11.2018) ist keiner davon korrigiert.
Als Workaround half mir, dass man die problematischen Diskgruppen zumindest stoppen/unmounten konnte, ohne dass die CDB sich davon stören lässt. Dann kann man im Hintergrund die LUNs tauschen/ändern und eine andere Diskgruppe mit dem gleichen Namen wieder mounten.
Die einzige Möglichkeit zum erfolgreichen Kürzen dieser Abhängigkeitsliste ist, sie übereifrig zu kürzen. Laut Support soll das sogar im Betrieb möglich sein. Ich habe das aber nur durchgeführt, als die CDB mal offline war. Dazu habe ich per `srvctl modify database` die Liste auf 3 Diskgruppen reduziert. Im Zuge des Restarts der CDB und ihrer PDBs haben sich die wirklich gültigen Abhängigkeiten von selbst regeneriert. Alternativ hätte ich auch mit dem `-nodiskgroup` Parameter die Liste gänzlich zurücksetzen können.

Erfahrungen Betrieb – SYSAUX Tablespace Wachstum
Nach etwas Laufzeit der CDB mit immer mehr PDBs fiel ein sehr ungleichmäßiges Wachstum der `SYSAUX`-Tablespaces in den PDBs auf (siehe Abb. 2). Manche kommen mit weniger als 1GB aus, währende andere bereits mehr als 6GB benötigen. Vor allem der Optimizer Statistics Advisor, welcher mit 12.2 neu eingeführt wurde, ist für hohen Platzverbrauch im SYSAUX-Tablespace bekannt und kann bei Bedarf in die Schranken verwiesen werden (Doc ID 2305512.1). In meinem Fall geht ein Teil des hohen Platzverbrauches auch auf dessen Konto.
Die andere Quelle für den Platzverbrauch ist der Verlauf von Optimizer-Statistiken über die letzten 31 Tage (Standardwert). Datenbanken mit vielen Objekten haben viele Statistiken und damit auch einen größeren Platzbedarf für deren Verlauf. Hier kann man bei Bedarf die Aufbewahrungsdauer reduzieren (Doc ID 329984.1 & Doc ID 452011.1). Das muss vor dem Upgrade und der Migration schon so gewesen sein, denn dieses Feature existiert seit 10g, es ist mir nur nicht aufgefallen.
Ich kann keine Empfehlung zur Konfiguration der beiden

CON_ID	TBS_NAME	USED_MB
14	SYSAUX	6,481
6	SYSAUX	4,553
1	SYSAUX	3,069
28	SYSAUX	2,526
10	SYSAUX	2,290
16	SYSAUX	2,165
30	SYSAUX	1,846
9	SYSAUX	1,707
4	SYSAUX	1,700
12	SYSAUX	1,627
11	SYSAUX	1,557
18	SYSAUX	1,501
22	SYSAUX	1,468
13	SYSAUX	1,431
21	SYSAUX	1,327
7	SYSAUX	1,327
25	SYSAUX	1,304
27	SYSAUX	1,185
8	SYSAUX	1,021
3	SYSAUX	806
24	SYSAUX	752
19	SYSAUX	533
31	SYSAUX	470
32	SYSAUX	458
20	SYSAUX	456
26	SYSAUX	448
15	SYSAUX	429
23	SYSAUX	429
29	SYSAUX	425
5	SYSAUX	398
17	SYSAUX	378

Abb. 2, Größe unterschiedlicher SYSAUX Tablespaces in einer CDB

Mechanismen abgeben. Für den Moment habe ich beide auf den Standardeinstellungen belassen und akzeptiere den Platzverbrauch.

Grundsätzliche Beobachtungen
Bevor 18c für On-Premise veröffentlich wurde, habe ich erste Erfahrungen bereits mit 12.2 gesammelt. Einige Beobachtungen davon gelten weiterhin unter 18c.

Nach dem man unter 12.2 von einer PDB auf einem Dateisystem (ASM wird dafür nicht unterstützt) per Snapshot Cloning eine Kopie erzeugt hatte, konnte man nur noch die Kopie öffnen, nicht mehr jedoch das Original. Von den Datafiles des Originals hatten sich die Permissions so geändert, das nur noch Lesezugriffe erlaubt waren. Eine manuelle Korrektur der Permissions korrigierte das Problem. Der Bug 25999762 in der Angelegenheit ist für 12.2 bisher nicht korrigiert.

Bei Upgrades früherer Versionen (z.B. 11.2.0.4 → 12.1.0.2) war es bereits nötig, eigene, in die DB JVM geladene JARs zu droppen. Erst nach dem Upgrade konnte man die JARs wieder laden. Das gilt auch weiterhin. Doch ich hatte bis 12.1.0.2 fünf JAR-Dateien aus dem Home geladen (`jaxrpc-api.jar`, `ejb.jar`, `dms.jar`, `jssl-1_1.jar` und `soap.jar`). Diese Dateien gehören aber seit 12.2 nicht mehr zum Lieferumfang. Wie es sich zeigte werden diese auch gar nicht mehr benötigt.

Beim Testen mit Multitenant unter 12.2 fiel mir auf, das die Views `[dba|cdb]_tablespace_usage_metrics` die Undo-Tablespaces nicht mehr anzeigen, obwohl sie sehr wohl unter `[dba|cdb]_tablespaces` auftauchen. In einem SR wurde das als Bug 24361167 identifiziert. Die Korrektur sei komplex, weshalb ein Fix erst für 19.1 angekündigt wurde. In der Tat besteht das Problem unter 18.3 weiterhin.

Ebenfalls beim Testen mit Multitenant - noch unter 12.2 - erschien mir die PDB-Cloning-Funktionalität von RMAN sinnvoll für den Fall, das man für Diagnose- oder Datenrettungszwecke eine PDB mit einem früheren Stand in einer anderen CDB wiederherstellt, während jedoch das Original weiterläuft.
Dafür schien mir `DUPLICATE DATABASE` zusammen mit `UNTIL TIME` geeignet, beispielsweise so:

```
rman auxiliary sys/xxx
Duplicate database to 'auxcdb' pluggable database pdb1 noopen backup location '/backup_aux' undo tablespace "pdb1:undotbs1" UNTIL TIME "TO_DATE ('2018-03-08_10:26:32', 'YYYY-MM-DD_HH24:MI:SS')";
```

Doch dergleichen schlägt immer mit einer merkwürdigen Fehlermeldung fehl: `RMAN-06019: could not translate tablespace name "pdb1:undotbs1"`. Es spielt keine Rolle, ob man dabei mit Shared Undo oder Local Undo arbeitet.
Ein SR dazu ergab, dass dieses Feature in 12.2 schlicht nicht unterstützt wird. Auch für 18.3 gilt das weiterhin. Ein Enhancement Request dafür läuft über Bug 27891119.
Die einzigen Alternativen sind der In-Place PITR (point in time recovery) auf der Original-CDB, doch dann kann das Original nicht weiterlaufen. Weiterhin beschreibt Doc ID 2142675.1 genau diesen PDB-PITR auf einem anderen Server (und so in einer anderen CDB) doch er basiert auf dem PITR der gesamten CDB. Dazu kann man beim Restore noch eine Liste der gewünschten PDBs mitgeben, während jedoch der Recovery eine Negativ-Liste aller Tablespaces aller PDBs benötigt, welche nicht wiederherzustellen sind. Das funktioniert zwar, ist aber extrem unhandlich und fehleranfällig. Entsprechend bleibt nur zu hoffen, dass über den Enhancement Request der DUPLICATE für PDBs künftig dazulernt.

Mit dem ersten RU für 18c (18.4) hatte ich anfangs Probleme, dieses erfolgreich auf meine ASM-Installationen anzuwenden. Die Installation von 18.4 schlug fehl, weil `opatchauto` die Rollback-Skripte von 18.3 nicht finden konnte. Dies war jedoch mein Fehler.
Bei dem neuen Installationsmodus von 18c enthält das Installations-Archiv bereits ein fertiges Home, dass an den Zielort zu kopieren oder dort zu entpacken ist. Hier gilt es sicherzustellen, dass dabei auch die versteckten Ordner `.patch_storage` und `.opatchauto_storage` am Zielort ankommen. In meinem Fall war das nicht der Fall und so war dieser Fehler vorprogrammiert.

Fazit
Nach kleineren Anfangsschwierigkeiten laufen die Migrationen für den Moment so reibungslos, wie ich gehofft hatte. Natürlich gab es auch Probleme, doch dafür gab es immer zeitnah eine Lösung oder einen passablen Workaround. Für die Zukunft sehe ich nur ein Problem am Horizont: Die Migration von Datenbanken mit Usern, deren Passwort noch auf 10g basiert. Ich weiß bereits das einige unserer DBs davon betroffen sind. Ohne manuelle Änderungen des Passwortes werden diese User nach der Migration ausgesperrt.
Im Betrieb zeigen sich noch ein paar Probleme, doch nichts davon kann als Showstopper bezeichnet werden. Die meisten davon dürften zu lösen sein oder sind aktuell nur von theoretischer Natur für mich.
Während ich anfangs die Exklusivität von 18c nur für die Cloud noch als Problem empfand, scheint sich dies aber für die Stabilität und die Beseitigung von Kinderkrankheiten gelohnt zu haben. Ich bin sehr optimistisch, dass sich unsere Investition in die Multitenant-Option und die damit verbundene Migration und Konsolidierung nach 18c auch langfristig als Erfolg darstellen wird.

Kontaktadresse:
Robert Ortel
Hypoport Systems GmbH
Klosterstraße 71
D-10179 Berlin

Telefon: +49 (0) 30 42086-2206
E-Mail robert.ortel@hypoport-systems.de
Internet: www.hypoport.de