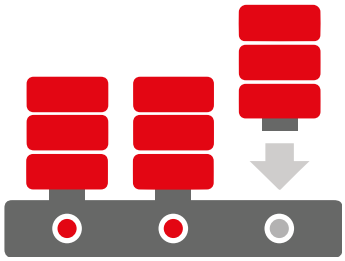

Von 12.1 NonCDBs zu 18c Multitenant – ein Erfahrungsbericht

Robert Ortel, Hypoport Systems GmbH

In einem Umfeld einer gewachsenen Infrastruktur entstand die Idee zur Vereinheitlichung, Vereinfachung und Konsolidierung der vorhandenen Datenbank-Infrastruktur. Die Saat dazu hatte die Multitenant-Option gesät. Der Artikel stellt die ehemalige Infrastruktur des Autors kurz vor und erläutert, welche Überlegungen ihn von Multitenant überzeugt haben. Er stellt insbesondere die Erfahrungen rund um die Upgrades und Migrationen sowie den Betrieb über die ersten Wochen vor und wie er auch ohne RAC eine möglichst hohe Verfügbarkeit erreichen will.

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, standen die vorhandene Hardware und die Anforderungen an diese in keinem Verhältnis mehr.

Benötigter RAM, lokaler Plattenplatz auf den physischen Servern (der mit jedem Patch-Bundle anwächst) und die Limitierung, den gleichen Datenbank-Namen nicht mehrfach pro Server verwenden zu können, begrenzten die Konsolidierung. Ohnehin war Konsolidierung bisher kaum ein Thema für die Oracle-Datenbanken im Unternehmen des Autors. So kam ihm eine Konsolidierungsidee in den Sinn, mit der Multitenant-Option die Anzahl der Server zu reduzieren und gleichzeitig für eine deutlich bessere Ausnutzung der Lizenzen zu sorgen. Primär wird nachfolgend die technische Umsetzung dargestellt.

Die alte Infrastruktur

Verteilt auf 16 physische Servern und zwei Standorte liefen 76 Datenbank-Instanzen. Aufgrund der Oracle-Lizenzpolitik setzte man nur physische Server exklusiv für die Oracle-Datenbank ein. All diese Datenbank-Instanzen der beiden Editionen liefen unter 12.1.0.2 als Single Instance und NonCDB. Weil diese sehr stabil ist und nach Erfahrung des Autors nur im Falle von Hardware-Problemen den Dienst versagt, genügte die Single Instance bisher den Anforderungen. Hardware-Probleme erfordern dann zwar ein manuelles Eingreifen, doch dank verfügbarer Ersatz-Hardware und schneller Mechanismen zu deren Inbetriebnahme toleriert das Unternehmen diese möglichen Ausfallzeiten. Eine Migration zu Singletenant war bisher wegen Problemen im Oracle-Migrations-Skript („noncdb_to_pdb.sql“) bei der Migration von 11.2.0.4 nach 12.1.0.2 vertagt worden.

Auf jedem der physischen Server läuft ASM und stellt die LUNs von Shared Storages jeweils als drei Diskgruppen pro Datenbank zur Verfügung. Diese speichern dann jeweils Datafiles, Logfiles und Dateien der Recovery Area. Auf produktiven Datenbanken sind diese Diskgruppen redundant über zwei Shared Storages ausgelegt („normal redundancy“). Auf den Test- und Entwicklungssystemen arbeitet man bisher ohne Redundanz seitens ASM („external redundancy“). Nur für die Dateien des Backups verwendet man reguläre Dateisysteme, entweder gemountet per NFS oder als LUN von einem Shared Storage mit einem Dateisystem. Durch den Einsatz von ASM auf jedem Server setzt man auch die Funktionalität von Oracle Restart überall ein.

Optimierungsidee

Die meisten der sechzehn 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 Datenbank-Namen das Verwenden eines neuen Servers nötig machte, der (vor allem für Test und Entwicklung) meist bereits existierte und nicht erst bestellt werden musste.

Mit der Idee hinter Multitenant wurde nach und nach klar, dass sich dies grundlegend ändern ließ. Die sechzehn Server könnte man auf zwei Server reduzieren (ohne Betrachtung der Ersatz-Hardware). Diese zwei Server würden jedoch deutlich mehr RAM benötigen. Zusätzlich ließe sich über die zwei Standorte eine Replikation betreiben, die neue Vorteile unter anderem für die Hochverfügbarkeit bietet. Nicht zuletzt bringt das Cloning von PDBs mit Multitenant neue Möglichkeiten für Entwicklung und Tests mit sich. Neben den Optimierungen rund um Platzbedarf im Server-Rack, Stromverbrauch und

Wartungsaufwand für die vielen Datenbank-Server, die Verbesserung der Hochverfügbarkeit und der neuen Features war auch eine Optimierung der Lizenzen möglich.

Die neue Infrastruktur

Auf jedem der zwei neuen Server läuft nun allerdings nicht nur eine Container Database (CDB), sondern jeweils zwei. 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 zweite CDB auf dem Server aushelfen und die PDBs nach einem Neustart (wegen Anpassung der SGA/PGA) aufnehmen, die zuvor in der ersten CDB liefen. Außerdem lassen sich so die Patching-Downtimes verkürzen, die meist zwei Mal im Jahr für Release-Updates durchgeführt werden. Die zweite CDB kann man bereits patchen und muss dann nur die PDBs umhängen. Die Ausführung von „datapatch" gelingt dann wieder online.

Auf jedem der zwei neuen Server läuft wieder ASM und damit Oracle Restart. Jede CDB erhält wieder ihre üblichen drei 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 zusätzlich Plattenplatz ein, da sich nun auch alle eine Art Hochlast-Reserve 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 zwei Shared Storages versorgt. So schützt die Redundanz vor Datenfehlern und dem Ausfall eines Storage.

An jedem der zwei Standorte wird einer der Server laufen und eine Ersatz-Hardware bereitstehen. Der aktive Server an jedem Standort wird per Data Guard eine Replikation auf den anderen aktiven Server am anderen Standort vollführen. Wie von Oracle vorgesehen, laufen die 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 etc.) nutzen zu können.

Migrationspfad

Für den Migrationspfad von 12.1.0.2 NonCDBs zu 18.3.0.0 Multitenant bieten sich viele Möglichkeiten. In diesem Fall ist Ausführungssicherheit bei der Migration wichtiger als die Kürze der Downtime, da nächtliche Downtimes vergleichsweise einfach zu erhalten sind. Als einziger Oracle-DBA möchte der Autor die Methoden möglichst sicher beherrschen, was nur möglich ist, wenn dies nicht so viele sind. So fiel die Entscheidung auf die klassische Migration in zwei Phasen, analog zu dem, wie es auch Mike Dietrich *(siehe „https://mikedietrichde.com/2017/03/08/converting-an-12-1-non-cdb-and-plug-it-into-an-12-2-cdb")* für das Upgrade von 12.1 nach 12.2 empfiehlt: 12.1 NonCDB nach 18.3 NonCDB nach 18.3 PDB. Alternativ kommt für einige Sonderfälle die Migration mit klassischem Datapump zum Einsatz.

Die zwei-phasige Migration dauert für alle Datenbanken, egal welcher Größe, nahezu gleich lang, weil nicht der gesamte Datenbestand kopiert werden muss. Das ist ein großer Vorteil, denn es gilt auch Multi-Terabyte-Datenbanken zu migrieren. Die zwei Kernphasen dauern ca. 25 Minuten (12.1 NonCDB nach 18.3 NonCDB) und 12 Minuten (18.3 NonCDB nach 18.3 PDB). Zusammen mit allen manuellen Vorbereitungen (insbesondere Checks), Zwischenschritten (wie DST.-Update) und Nachbereitungen (Anpassungen jeder PDB) war pro Datenbank knapp eine Stunde erforderlich.

Der Pfad mit Datapump kann bei kleinen Datenbanken einen Geschwindigkeitsvorteil bieten und dient als Fallback für alle anderen Datenbanken. Nur bei den wenigen großen Datenbanken mit mehreren Terabytes steht diese Option dann quasi nicht zur Verfügung, weil sie viel länger brauchen würde.

Checks vor der Migration

Im Zuge der Checks vor der Migration hielt sich der Autor an verschiedene Empfehlungen. Auch wenn es keine Checkliste für die manuelle Migration nach 18c NonCDB zu geben scheint, hält er sich zumindest an die Checkliste für die Migration nach 12.2 NonCDB: Doc ID 2173141.1. Es ist zwar Teil dieser Checkliste, er möchte dennoch die Ausführung von „dbupgdiag.sql" (siehe Doc ID 556610.1) hervorheben. Es liefert zum Zustand der Datenbank vor dem Upgrade eine gute Zusammenfassung, die insbesondere für den Support im Problemfall nötig ist.

Nicht Teil der Checkliste, dafür aber von Mike Dietrich *(siehe „https://mikedietrichde.com/2018/09/21/improved-preupgrade-jar-for-oracle-12-2-and-18c")* empfohlen und über Doc ID 884522.1 zu beschaffen, ist das Oracle-Database-Pre-Upgrade-Utility, das vor jedem Upgrade ausgeführt wurde. Sind all diese Checks erfolgreich, startet das Upgrade.

Erfahrungen mit „DBMS_PDB.CHECK_PLUG_COMPATIBILITY"

Mit den ersten Tests für die Migration hielt man sich an den Vorschlag von Mike Dietrich: 12.1 NonCDB nach 18.3 NonCDB nach 18.3 PDB. Während das Upgrade nach 18.3 NonCDB von Beginn an tadellos funktionierte, gab es gleich beim ersten Versuch mit dem zweiten 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 gravierenden Fehler: „ORA-07445 [__intel_ssse3_rep_memcpy() +6471]". Ein Ergebnis des Checks erhält man nicht.

Durch einen Service Request konnte dies als Bug 28502403 identifiziert werden, wofür es zwischenzeitlich einen Patch gibt. Dieser beseitigt das Problem erfolgreich und ist ebenfalls im Release-Update 18.4 enthalten. 18.4 konnte bisher allerdings noch nicht getestet werden.

Inzwischen hatte man sich einen Workaround überlegt. Die Migration lässt sich auch entlang eines anderen Pfads vollziehen: 12.1 NonCDB nach 12.1 PDB nach 18.3 PDB. In 12.1.0.2 arbeitet der Aufruf „CHECK_PLUG_COMPATIBILITY" tadellos und so ist dies eine Alternative. Da man jedoch Datenbanken mit beiden National Charactersets betreut (UTF16 und UTF8), sind in diesem Fall für den ersten Migra-

tionsschritt zwei 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 gab es einige erfolgreiche Migrationen, man stieß aber vereinzelt auch auf Probleme. In einem 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 dem Support), sodass es nur noch maximal einen Eintrag jeweils zu „APPLY" und „ROLLBACK" gibt, lief „CHECK_PLUG_COMPATIBILITY" erfolgreich durch.

Eine andere Migration zu einer 12.1 PDB verursachte, dass die View „SYS.USER_AUDIT_POLICIES" hinterher „invalid" war. Ein Neukompilieren beseitigte das Problem allerdings nicht. Weil die Verfügbarkeit des Patches zum Bug 28502403 kurz nach dem Auftreten dieses Problems gegeben war, konnte man diesen Migrationsweg wieder verwerfen, ohne dazu eine Lösung zu finden. Nach Verifikation des Patches gegen den „ORA-07445" beim „CHECK_PLUG_COMPATIBILITY" kehrte man zum ursprünglichen Migrationspfad (12.1 NonCDB nach 18.3 NonCDB nach 18.3 PDB) zurück und verwendet diesen bis heute.

Erfahrungen mit Enterprise Manager Repository DB

Das Upgrade der Enterprise Manager Repository DB war das einzige Mal, dass die Phase 12.1 NonCDB nach 18.3 NonCDB an einem Problem scheiterte. In der letzten Phase (108) brach das Upgrade mit Perl-Fehlern ab und scheiterte laut Upgrade-Logfile an Listing 1.

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

Listing 1

Zu diesem Zeitpunkt versuchte man auch das Upgrade über die 12.1 PDB erneut, das ebenfalls im Zusammenhang mit „NONUPGRADED_TABLEDATA" scheiterte. In MOS fand sich dann Doc ID 2279497.1, die anweist, dass in diesem Fehlerfall vor dem Upgrade zwei Skripte auszuführen sind, um dieses Problem zu beseitigen. Nach einem Rückfall auf einen Stand der DB vor dem Upgrade und der Ausführung dieser Skripte lief das Upgrade 12.1 NonCDB nach 18.3 NonCDB erfolgreich durch.

Erfahrungen mit dem Platz auf der Diskgruppe

Nach dem Upgrade der Datenbank von 12.1 NonCDB auf 18.3 NonCDB steht die Migration zu einer PDB auf dem Plan. An dieser Stelle scheiterte man auch mal, weil nicht beachtet wurde, dass im Zuge dieser Migration der Platzbedarf leicht steigt. In diesem Fall 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 als inaktiv und kann nicht gestartet werden. 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", das vor „noncdb_to_pdb.sql" aufzurufen ist, 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 mit Database Services unter 18c

In der 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 *(siehe „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")* sei die „OSOPER"-Rolle nur optional festzulegen, woraus man schloss, dass dies ebenfalls für die Existenz der Betriebssystem-Gruppe „oper" gilt. Ein Service Request kam ebenfalls 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,

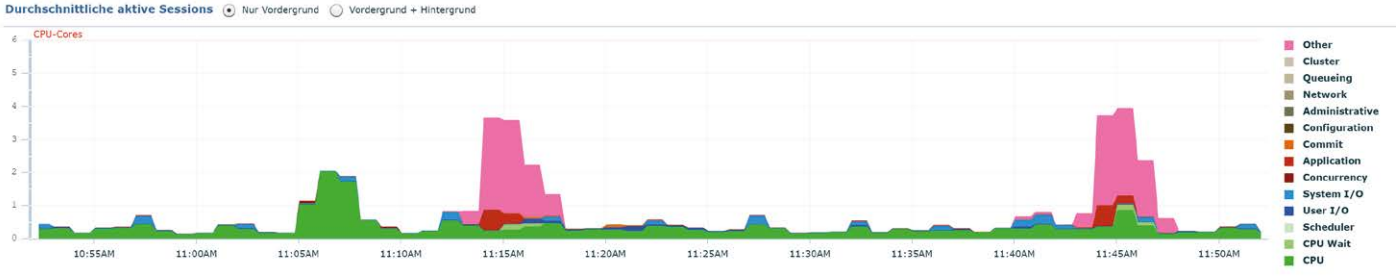

Abbildung 1: Wait-Spitzen alle dreißig Minuten

TRIOLOGY { Agility for your business.

Oracle APEX- und PL/SQL-Entwickler (m/w/d)

TRIOLOGY GmbH

Agile Entwicklungsmethoden, exzellente Code-Qualität und aktuelle Technologien – dafür steht TRIOLOGY.
Gestaltungskraft, Expertise und Leidenschaft zeichnen unsere 100 Mitarbeiter aus. Sie auch? Dann wachsen Sie mit uns und den Herausforderungen, nutzen Sie Ihren Freiraum und werden Sie Teil unseres engagierten Expertenteams! We need your talent.

Unser Angebot

Bei TRIOLOGY erwartet Sie eine herausfordernde und vielseitige Tätigkeit in einem engagiertem Team mit attraktiven Gehaltskonditionen, flachen Hierarchien und einem Miteinander mit „Wir“-Gefühl. Regelmäßige Weiterbildungsmöglichkeiten und Zusatzangebote wie z.B. Physiotherapie, Englischunterricht sowie Events schaffen das ideale Umfeld für Ihren persönlichen Erfolg.

Ihre Herausforderung

- Selbstständiges Betreuen und Beraten von Projekten und Kunden
- Analysieren, Konzipieren, Entwickeln und Testen von modernen Webanwendungen und Datenbanklösungen mit Oracle Application Express (APEX) und PL/SQL in agilen und klassischen Projekten
- Modellieren und integrieren von Datenbanken und Webanwendungen in Enterprise-Architekturen
- Mitwirken bei der Gestaltung und Weiterentwicklung unseres Kompetenzbereichs

Das bringen Sie mit

- Studium oder Ausbildung mit IT-Bezug sowie Projekterfahrung Erfahrung in der Entwicklung komplexer Webanwendungen mit Oracle Application Express (APEX) und PL/SQL
- Sehr gute Kenntnisse in Internettechnologien, insbesondere HTML, CSS und JavaScript
- Gute Kenntnisse im Bereich SQL und Datenmodellierung Lösungs- und kundenorientierte Arbeitsweise sowie sicheres Auftreten im Projekt
- Sehr gute Deutschkenntnisse sind ein Muss
- Reisebereitschaft innerhalb Deutschlands

Senden Sie uns Ihre Bewerbung gern über das Online-Bewerbungsformular oder **jobs@triology.de**. Für Fragen steht Ihnen unser Team Human Resources auch gern telefonisch unter **+49 531 23528-52** zur Verfügung. Besuchen Sie uns online unter **triology.de** oder bei

auch ohne dieser Gruppe einen User zuzuweisen oder sie mit der Rolle „OSOPER" zu verknüpfen.

Erfahrungen mit wiederkehrenden Wait-Spitzen

Nach kurzer Zeit des Betriebs einiger PDBs in einer CDB fiel auf, dass es alle dreißig Minuten zu Wait-Spitzen von „Sync ASM Rebalance" und „enq: RC - Result Cache: Contention" kommt *(siehe Abbildung 1)*. Diese werden vom Enterprise Manager Agent (hier Version 13.3.0.0.0) verursacht, der mit mehreren Queries die Tablespace-Usage ermitteln möchte und dazu unter anderem „cdb_tablespace_usage _metrics" abfragt. Auch ein manuelles „select * …" auf dieser View verursacht eine kleinere Wait-Spitze für diese Waits.

Die PDBs, die bisher migriert wurden und auf dieser CDB laufen, scheinen darunter für den Moment nicht zu leiden. Jedoch verursachen sie 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 Service Request dazu noch kein Bug geöffnet worden.

Nach einiger Zeit kam der Support zu dem Schluss, dass der Underscore-Parameter „_use_cached_asm_free_space" auf „TRUE" zu setzen ist, um das Problem zu beseitigen. Es handelt sich um eine Wiederkehr von Bug 22243217, der zwar mit 12.1 korrigiert wurde, mit 12.2 aber zurückkehrt. Unter 18.3 beseitigen das Setzen dieses Parameters und ein Datenbank-Neustart das Problem.

Erfahrungen mit Core Dump bei Diskgroup-Dependency-Anpassung

Im Zuge der Migration der NonCDBs in PDBs auf einer CDB findet in diesem Fall bei jeder Migration einiges an Wartungsarbeit rund um die ASM-Diskgruppen statt (zwecks Erhöhung der Redundanz). Als Nebenwirkung wurde die Liste der Diskgruppen, von der die CDB-Ressource in Oracle Restart abhängig zu sein scheint, um viele Diskgruppen erweitert, zu denen nun keinerlei Abhängigkeit mehr besteht. Sofort war klar, dass dies bei einem Neustart der CDB Probleme verursacht, weil eben die Diskgruppen ohne echte Abhängigkeit auch nicht mehr mountbar sind.

```
[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
[…]
#
```

Listing 2

Zusätzlich gibt es in der Umgebung Diskgruppen mit identischen Namen, die 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, die zu migrieren sind, deren Störungspotenzial nicht bekannt war. Die Liste musste also gekürzt werden. Ein Aufruf von „srvctl modify database" zur Anpassung/Kürzung der Diskgruppen-Liste endete jedoch wegen eines „SIGSEGV" in „libhasgen18.so" in einem Core Dump *(siehe Listing 2)*.

Wie sich in einem Service Request zeigte, verträgt „srvctl modify database" für die Liste mit Diskgruppen keine sonderlich lange Liste: Die Grenze liegt zwischen acht Diskgruppen mit 150 Zeichen und neun Diskgruppen und 167 Zeichen, 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, dass man die Diskgruppen ohne echte Abhängigkeit 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 stärker zu kürzen als nötig. Laut Support soll das sogar im Betrieb möglich sein. Es wurde jedoch nur durchgeführt, als die CDB offline war. Dazu hat man per „srvctl modify database" die Liste auf drei Diskgruppen reduziert. Im Zuge des Restart der CDB und ihrer PDBs haben sich die wirklich gültigen Abhängigkeiten von selbst regeneriert. Alternativ hätte man mit dem „-nodiskgroup"-Parameter die Liste auch gänzlich zurücksetzen können. Später setzte der Support dann noch nach, dass Manipulationen an dieser Diskgruppen-Liste nicht empfohlen sind. Einzig das gänzliche Zurücksetzen, zusammen mit einem Datenbank-Neustart, sei bei Bedarf empfohlen, da so alle echten Abhängkeiten neu gesetzt werden.

Erfahrungen mit „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 Abbildung 2)*. Manche kommen mit rund 400 MB aus, während andere bereits mehr als das Zehnfache davon benötigen. Die Suche nach den Ursachen brachte zwei Ergebnisse: Vor allem der Optimizer Statistics Advisor, der 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, weil er vereinzelt viele Findings zu falsch ermittelten Statistiken hat und sich diese durch tägliche Prüfung auf Dauer anhäufen.

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 beziehungsweise der Migration schon so gewesen sein, denn dieses Feature existiert seit 10g.

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

Abbildung 2: Die Größe unterschiedlicher „SYSAUX"-Tablespaces in einer CDB

Ich kann keine Empfehlung zur Konfiguration der beiden Mechanismen abgeben. Für den Moment habe ich beide auf den Standardeinstellungen belassen und akzeptiere den Platzverbrauch.

Grundsätzliche Beobachtungen

Bevor 18c für On-Premises veröffentlicht wurde, hat der Autor bereits erste Erfahrungen mit 12.2 gesammelt. Einige Beobachtungen davon gelten auch weiterhin unter 18c: Bei Upgrades früherer Versionen (etwa 11.2.0.4 nach 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. Bis 12.1.0.2 wurden fünf JAR-Dateien aus dem Home geladen („jaxrpc-api.jar", „ejb.jar", „dms.jar", „jssl-1_1.jar" und „soap.jar"). Diese Dateien gehören jedoch seit 12.2 nicht mehr zum Lieferumfang. Wie sich zeigte, werden diese aber gar nicht mehr benötigt.

Beim Testen mit Multitenant unter 12.2 fiel 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 Service Request 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 die PDB-Cloning-Funktionalität von RMAN sinnvoll für den Fall, dass man für Diagnose- oder Datenrettungszwecke eine PDB mit einem früheren Stand in einer anderen CDB wiederherstellt, während jedoch das Original weiterläuft. Dafür schien „DUPLICATE DATABASE" zusammen mit „UNTIL TIME" geeignet *(siehe Listing 3)*.

Doch das 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 Service Request 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, die 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 Release-Update für 18c (18.4) gab es anfangs Probleme, dieses erfolgreich auf meine Grid-Infrastructure-Installationen anzuwenden. Die Installation von 18.4 schlug fehl, weil „opatchauto" die Rollback-Skripte von 18.3 nicht finden konnte. Dies war jedoch ein eigener 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. Hier war das nicht der Fall und so war dieser Fehler vorprogrammiert.

Fazit

Nach kleineren Anfangsschwierigkeiten laufen die Migrationen für den Moment so reibungslos, wie man gehofft hatte.

```
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')";
```

Listing 3

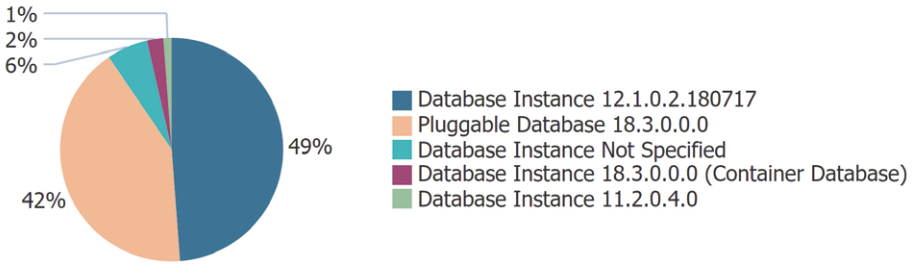

Abbildung 3: Übersicht aus dem Enterprise Manager zum Migrationsfortschritt

Von den 76 DB-Instanzen sind bisher 33 nach 18.3 PDB migriert *(siehe Abbildung 3)*. Natürlich gab es auch Probleme, doch dafür gab es immer zeitnah eine Lösung oder einen passablen Workaround. Für die Zukunft sieht man nur ein Problem am Horizont: Die Migration von Datenbanken mit Usern, deren Passwort noch auf 10g basiert. Es ist bekannt, dass einige Datenbanken davon betroffen sind. Ohne manuelle Änderungen des Passworts sind diese User nach der Migration ausgesperrt.

Im Betrieb zeigen sich noch ein paar Probleme, doch nichts davon ist ein Showstopper. Die meisten davon dürften zu lösen sein oder sind aktuell nur von theoretischer Natur. Während der Autor 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. Er ist sehr optimistisch, dass sich die Investition in die Multitenant-Option sowie die damit verbundene Migration und Konsolidierung nach 18c auch langfristig als Erfolg darstellen wird.

Robert Ortel
robert.ortel@hypoport-systems.de

DOAG Botschafter für Technologie 2018: Niels de Bruijn

Für sein großes Engagement innerhalb der DOAG wurde Niels de Bruijn, Themenverantwortlicher Oracle Application Express (Apex), auf der DOAG 2018 Konferenz + Ausstellung zum Botschafter für Technologie gekürt. Diese Auszeichnung, die ihm Robert Szilinski, DOAG-Vorstand und Leiter der Development Community überreichte, hat er sich verdient – da stimmte auch das Publikum zu, das den Preisträger mit einer La-Ola-Welle feierte, die mittlerweile zum Wahrzeichen der Apex-Community geworden ist.

Seine Leidenschaft für die Entwicklungsumgebung kommt nicht von ungefähr; auch beruflich ist Niels de Bruijn als Fachbereichsleiter für Apex bei der MT AG in Ratingen tätig. „Wer mit Apex gearbeitet hat, weiß, warum ich mich dafür einsetze: es funktioniert einfach. Aus meiner Sicht ist die Apex-Community einzigartig und ich bin stolz darauf, ein Teil davon zu sein.“, so der Konferenz- und Programmleiter der Apex Connect Konferenz, die er selbst ins Leben gerufen hat. Mit großem Erfolg: Im Jahr 2018 konnte die Veranstaltung ein neues Rekordhoch von 350 Teilnehmern aus 16 verschiedenen Ländern verzeichnen. De Bruijns Ziel ist es, dazu beizutragen, die Software-Entwicklungsumgebung auch in Deutschland noch populärer zu machen. De Bruijn freut sich über die Auszeichnung, die „für mich eine Bestätigung für meine, anscheinend gute, Arbeit in der Community ist“, so der frisch ernannte Botschafter.