Einfache SAP-Migrationen mit Cross-Platform Transportable Tablespaces

Andris Perkons
Oracle Deutschland B.V. & Co. KG
Düsseldorf

Jan Brosowski
Oracle Deutschland B.V. & Co. KG
Walldorf

Schlüsselworte
SAP Oracle Transportable Tablespaces export import

Einleitung

Seit Version 10 der Oracle DB besteht die Möglichkeit, ganze Tablespaces plattformübergreifend zu migrieren. Der Vorteil dieser Migrationsmethode besteht darin, dass Vorbereitungs- und Durchführungszeiten relativ kurz sind. Diese Migrationsmethode ist für SAP-Datenbanken seit vielen Jahren freigegeben. Im Rahmen der Oracle Optimized Solution für SAP wurde die Methode ausführlich getestet, und die wichtigsten Parameter zur Beschleunigung des Verfahrens identifiziert. Mit über 5000 Zugriffen ist die Methode die am häufigsten nachgefragte Migrationsmethode in der Oracle Optimized Solution for SAP. Dieser Vortrag stellt die Methode dar und liefert eine Schritt-für-Schritt-Anleitung zur Migration von SAP-Systemen mittels Transportable Tablespaces. Er zeigt, wie ein Endianwechsel oder der Umstieg von Filesystem-basierten Datenbanken auf ASM durchgeführt wird. Die wichtigsten Parameter zur Beschleunigung dieses Verfahrens werden erläutert. Im Nachgang wird dann auf die Möglichkeit zur Verkürzung der Downtime durch eine inkrementelle Migration sowie Neuerungen in Oracle 12c eingegangen.

Understanding Transportable Tablespaces

Transportable tablespaces (TTS) provide a proven method for transferring Oracle Database instances for SAP systems from any source platform to any destination platform. Introduced with Oracle Database 10*g*, this approach offers a reliable choice for Oracle Database migrations between platforms with different endianness, particularly when a shorter downtime than that of the R3LOAD method is needed. In addition to a shorter downtime, a TTS migration typically requires very little SAP-specific knowledge.

The principal idea behind the TTS method is to export the Oracle Database instance's metadata that describes the database's data files in a platform-independent format, and then to make both the metadata and the data files available on the destination side of the migration. As an example, this article describes a migration of an Oracle Database instance to a file system based on Oracle Automatic Storage Management (Oracle ASM).

Generally, a TTS migration is executed in three phases: Export, Transport, and Import. This article explains the steps in each phase and their dependencies, providing detailed examples of the steps in a validated sample migration. It also examines the factors that influence the length of an overall Oracle Database migration using TTS. Beyond the procedure given here to migrate an Oracle Database instance, additional steps are necessary to move other components in the SAP environment (such as SAP file systems, binaries, configurations, and so on).

Other tasks must also be performed after the Oracle Database migration is complete. (For details, refer to the other articles in this series.) Figure 1 gives an overview of the migration tasks and activities in a TTS approach.

A TTS migration can follow different paths:

- An export followed by a metadata import
- A transfer of data files between source and destination platform
- A combination of both if not all steps can be run in parallel

An important consideration when determining the flow of a migration process is the downtime required for the SAP system. As depicted in Figure 1, TTS imposes some amount of downtime from the step of exporting the metadata until the database is validated on the destination side. The total downtime is determined by how fast and how parallelized the steps can be. When using TTS for a system copy, the time it takes for the metadata export determines the downtime required on the source system.

There are several factors that influence the length of the export and import processes:

- **The number of database objects.** Only the metadata describing database objects is exported, so the total number of objects directly impacts the time spent for both metadata export and import.
- **The bandwidth and method used to copy data files from source to destination.** The size of data files in the file system, rather the amount of data in the database, is what's relevant in determining copy times. In addition, data files might contain some unused data blocks that have not yet been overwritten with useful data. The bandwidth of the interconnect also matters if data files are copied during the migration.
- **CPU speed.** Because the metadata export is a single-threaded process, a CPU capable of very high single-thread performance, such as Oracle's SPARC M7 processor, can greatly reduce the runtime of the export process.

Export Phase

Pre-Export Checks

Step 1. Identify all the tablespaces and data files that need to be transported.

As a first step, it is critical to identify the tablespaces and data files that need to be exported to the destination platform. To do this, log in to the Oracle Database instance for the SAP system as `ora<SID>` or user `oracle` (depending whether you are using the old or new user schema for software ownership).

```
SQL> set pagesize 1000;
SQL> select tablespace_name,file_name from dba_data_files order by
tablespace_name,file_name;
TABLESPACE_NAME   FILE_NAME
--------------------------------------------------
PSAPSR3         /oracle/PR1/sapdata3/sr3_1/sr3.data1
PSAPSR3         /oracle/PR1/sapdata3/sr3_10/sr3.data10
PSAPSR3         /oracle/PR1/sapdata3/sr3_11/sr3.data11
PSAPSR3         /oracle/PR1/sapdata3/sr3_12/sr3.data12
PSAPSR3         /oracle/PR1/sapdata3/sr3_13/sr3.data13
PSAPSR3         /oracle/PR1/sapdata3/sr3_14/sr3.data14
PSAPSR3         /oracle/PR1/sapdata3/sr3_15/sr3.data15
PSAPSR3         /oracle/PR1/sapdata3/sr3_16/sr3.data16
```

```
PSAPSR3         /oracle/PR1/sapdata3/sr3_17/sr3.data17
PSAPSR3         /oracle/PR1/sapdata3/sr3_18/sr3.data18
PSAPSR3         /oracle/PR1/sapdata3/sr3_19/sr3.data19
PSAPSR3         /oracle/PR1/sapdata3/sr3_2/sr3.data2
PSAPSR3         /oracle/PR1/sapdata3/sr3_20/sr3.data20
PSAPSR3         /oracle/PR1/sapdata3/sr3_21/sr3.data21
PSAPSR3         /oracle/PR1/sapdata3/sr3_22/sr3.data22
PSAPSR3         /oracle/PR1/sapdata3/sr3_23/sr3.data23
PSAPSR3         /oracle/PR1/sapdata3/sr3_24/sr3.data24
PSAPSR3         /oracle/PR1/sapdata3/sr3_25/sr3.data25
PSAPSR3         /oracle/PR1/sapdata3/sr3_3/sr3.data3
PSAPSR3         /oracle/PR1/sapdata3/sr3_4/sr3.data4
PSAPSR3         /oracle/PR1/sapdata3/sr3_5/sr3.data5
PSAPSR3         /oracle/PR1/sapdata3/sr3_6/sr3.data6
PSAPSR3         /oracle/PR1/sapdata3/sr3_7/sr3.data7
PSAPSR3         /oracle/PR1/sapdata3/sr3_8/sr3.data8
PSAPSR3         /oracle/PR1/sapdata3/sr3_9/sr3.data9
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_1/sr3740.data1
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_10/sr3740.data10
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_11/sr3740.data11
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_12/sr3740.data12
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_13/sr3740.data13
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_14/sr3740.data14
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_15/sr3740.data15
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_16/sr3740.data16
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_17/sr3740.data17
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_18/sr3740.data18
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_2/sr3740.data2
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_3/sr3740.data3
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_4/sr3740.data4
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_5/sr3740.data5
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_6/sr3740.data6
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_7/sr3740.data7
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_8/sr3740.data8
PSAPSR3740      /oracle/PR1/sapdata2/sr3740_9/sr3740.data9
PSAPSR3USR      /oracle/PR1/sapdata4/sr3usr_1/sr3usr.data1
PSAPUNDO001     /oracle/PR1/sapdata1/undo_1/undo.data1
PSAPUNDO001     /oracle/PR1/sapdata1/undo_2/undo.data2
PSAPUNDO002     /oracle/PR1/sapdata1/undo002_1/undo002.data1
SYSAUX          /oracle/PR1/sapdata1/sysaux_1/sysaux.data1
SYSTEM          /oracle/PR1/sapdata1/system_1/system.data1
SYSTEM          /oracle/PR1/sapdata1/system_2/system.data2
50 rows selected.
```

Listing 1: Identifying tablespaces for migration.

Not all tablespaces must be transported. SYSTEM, SYSAUX, TEMP, and PSAPUNDO must be replaced by new empty tablespaces of the same type on the destination system. In Listing 1, the tablespaces PSAPSR3, PSAPSR3740, and PSAPSR3USR must be transported. Note that both the name and the number of tablespaces will differ for each database.

Step 2. Determine the character set.

While newer SAP systems (such as the source system in this migration example) are Unicode systems using UTF8 character sets for both the character and national character sets, older SAP systems might use different character sets. Because character set information is needed later during the creation of the destination database, it should be checked and noted:

```
SQL> select parameter, value from v$nls_parameters where
parameter='NLS_CHARACTERSET';
PARAMETER                 VALUE
------------------------- -----------------------
NLS_CHARACTERSET          UTF8

SQL>
```

Listing 2: Checking character set information.

Step 3. Move table SAPUSER to a transportable tablespace.

Note: This step is required only if moving from Oracle Database 11*g* Release 2 to the same release on a different platform. Starting with Oracle Database 12*c*, the use of Secure Storage in File System (SSFS) is mandatory.

When upgrading from Oracle Database 11*g* Release 2 to Oracle Database 12*c*, it's necessary to create an SSFS on the target environment that is accessible by all SAP ABAP system components. For details, read SAP DOC-49333 (SSFS Implementation for Oracle Database) and SAP Note 1622837 (New Connect Method of AS ABAP to Oracle via SSFS). (Access to SAP Notes requires logon and authentication to the SAP Service Marketplace.)

The table SAPUSER owned by user Oracle `OPS$<SAPSID>ADM` is usually located in the SYSTEM tablespace, which is not a transportable tablespace. For this reason, the table SAPUSER must be moved to a tablespace that can be transported, as in the following:

```
SQL> select tablespace_name from dba_tables where table_name='SAPUSER';
TABLESPACE_NAME
------------------------------
SYSTEM

SQL> alter table ops$rm1adm.sapuser move tablespace psapsr3usr;
Table altered.

SQL> select tablespace_name from dba_tables where table_name='SAPUSER';
TABLESPACE_NAME
------------------------------
PSAPSR3USR
```

Listing 3: Moving the SAPUSER table.

For a database enabled for SSFS, the SELECT command will output the following:

```
SQL> select tablespace_name from dba_tables where table_name='SAPUSER';
No row selected
```

Listing 4: Output of the SELECT command with SSFS enabled.

Step 4. Verify that the tablespace set is self-contained.

For transportable tablespaces, it is required that the set of tablespaces to be transported together is self-contained. Self-contained means that there are no dependencies to tablespaces beyond the set and that the only dependencies are between tablespaces within the set.

If the number of tablespaces is small, this check can be done manually:

```
SQL> dbms_tts.transport_set_check('PSAPSR3','PSAPSR3740','PSAPSR3USR',
true)
SQL> select violations from transport_set_violations;
```

Listing 5: Manually checking whether the tablespace set is self-contained.

If the number of tablespaces is large, the following `ttscheck.sql` script collects the names of all tablespaces that must be transported, starts the check procedure, and outputs violations, if they are any:

```
set serveroutput on;
declare
tslist varchar2(32767);
first boolean:=true;
begin
/* Create a comma separated list of all tablespaces that need to be
transported.*/
for c in (select tablespace_name from dba_tablespaces where
contents='PERMANENT' and tablespace_name != 'SYSTEM' and tablespace_name !=
'SYSAUX') loop

if first = true then
first:=false;
else
tslist:=tslist||',';
end if;
tslist:=tslist||c.tablespace_name;
end loop;
/* Output of the tablespaces list, then start the transport check */
dbms_output.put_line('Checking: '||tslist);
dbms_tts.transport_set_check(to_clob(tslist),false,false);
/* Output violations if there are any.*/
for c in (select violations from transport_set_violations) loop
dbms_output.put_line(c.violations);
end loop;
end;
/
```

Listing 6: The **ttscheck.sql** script can be used to check whether the tablespace set is self-contained.

If the tablespaces to be migrated are self-contained, the script produces output such as the following:

```
SQL> @ttscheck.sql
Checking: PSAPSR3,PSAPSR3740,PSAPSR3USR
PL/SQL procedure successfully completed.
```

Listing 7: Output from the **ttscheck.sql** script if tablespace set is self-contained.

Data File Conversion

If the bit ordering or underlying storage architecture changes between the migration source and destination platforms, Oracle Recovery Manager (Oracle RMAN) offers a means of converting data files between platforms. To check whether a conversion is necessary, compare the endianness of both platforms, as in the following:

```
SQL > select platform_name, endian_format from V$TRANSPORTABLE_PLATFORM

PLATFORM_NAME                            ENDIAN_FORMAT
---------------------------------------- ---------------
Solaris[tm] OE (32-bit)                  Big
Solaris[tm] OE (64-bit)                  Big
Microsoft Windows IA (32-bit)            Little
Linux IA (32-bit)                        Little
AIX-Based Systems (64-bit)               Big
HP-UX (64-bit)                           Big
HP Tru64 UNIX                            Little
HP-UX IA (64-bit)                        Big
Linux IA (64-bit)                        Little
HP Open VMS                              Little
Microsoft Windows IA (64-bit)            Little
IBM zSeries Based Linux                  Big
Linux x86 64-bit                         Little
Apple Mac OS                             Big
Microsoft Windows x86 64-bit             Little
Solaris Operating System (x86)           Little
IBM Power Based Linux                    Big
HP IA Open VMS                           Little
```

Listing 8: Comparing the endianness of both platforms.

Although a data file conversion script can be created manually, using Oracle RMAN to create one automatically is recommended. But even if the conversion script is created automatically with Oracle RMAN, the script file still typically requires some slight modifications.

Step 1. Create the conversion script with Oracle RMAN.

Because this article describes an Oracle Database migration to an Oracle ASM destination and new data files must be created in Oracle ASM anyway, the actual data file conversion is performed on the target platform. On the source side, Oracle RMAN first collects the data needed to create the new data files on the target and puts the needed commands into the conversion script `/oracle/conv.rman`. On the source platform, the structure of the Oracle RMAN command is as follows (the command sequence is also included in the script `create_conversion_script.rman`):

```
convert database on target platform
convert script '/oracle/conv.rman'
new database 'PR3'
format '+DATAC1';
```

Listing 9: The script **`create_conversion_script.rman`** contains the Oracle RMAN command on the source platform.

The values for the database name and format are determined by the target database. In this case, the Oracle ASM volume "+DATAC1" specifies the destination database, which is renamed from PR1 to PR3. The CONVERT SCRIPT parameter defines the name of the conversion script that will be executed manually on the destination host to convert data file copies in batch mode. (For more information, see "Transporting Data Across Platforms" in the *Oracle Database Backup and Recovery User's Guide*.

Running the Oracle RMAN script on the source platform builds the conversion script `/oracle/conv.rman` for the target:

```
$ rman target /
Recovery Manager: Release 12.1.0.2.0 - Production on Mon Mar 14 11:09:00
2016

Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights
reserved.

connected to target database: PR1 (DBID=994759330)

RMAN> @create_conversion_script.rman

RMAN> convert database on target platform
2>   convert script '/oracle/conv.rman'
3>   new database 'PR3'
4>   format '+DATAC1';
Starting conversion at source at 14-MAR-16
[…]
Edit init.ora file /oracle/PR1/121/dbs/init_+DATAC1.ora. This PFILE will be
used to create the database on the target platform
Run RMAN script '/oracle/conv.rman' on target platform to convert data
files
To recompile all PL/SQL modules, run utlirp.sql and utlrp.sql on the target
platform
To change the internal database identifier, use DBNEWID Utility
Finished conversion at source at 14-MAR-16
```

```
RMAN> **end-of-file**
```

Listing 10: Running the Oracle RMAN script on the source platform.

Step 2. Modify the created script.

The generated conversion script often includes statements that must be modified to match specifics of the destination platform. The script also typically includes information about tablespaces that will not be migrated (UNDO, SYS, and so forth), and should be tuned according to the target platform's ability to parallelize tasks.

Common modifications that are needed include

- Entering the correct location for the PFILE on the destination host. Although the Oracle RMAN script generates a PFILE, this PFILE is not usable and will be replaced by an SAP-compliant PFILE in a later step.
- Changing the PARALLELISM value to a reasonable value. This value is determined by the capability of the I/O system in the destination platform, the number of data files to import, and the number of available threads. In the example given here, the PARALLELISM value is set to 16 because the destination platform has 16 parallel threads available. (Note that each SPARC M7 processor has 32 cores with eight hardware threads per core. Cores are typically allocated to virtual machines, also known as Oracle VM Server for SPARC logical domains.)
- Verifying that the *path to the data files is correct on the destination platform*.

Appendix A provides an example of the conversion script generated by Oracle RMAN along with a modified version.

Metadata Export for SAP Database Objects

Exporting metadata for the database objects owned by the SAPSR3 user requires downtime of the entire SAP system, because the SAP-related tablespaces must be put in read-only mode. During the metadata export process, data files can also be copied or converted. Make sure that the file systems for the source database are accessible on the target system (for example, via NFS mounts).

Step 1. Set the tablespaces to read-only mode.

The source database must be open and mounted. Set the SAP-specific tablespaces that must be transferred to the destination database to read-only mode:

```
SQL> select tablespace_name, status from dba_tablespaces;

TABLESPACE_NAME                STATUS
------------------------------ ---------
SYSTEM                         ONLINE
SYSAUX                         ONLINE
PSAPUNDO001                    ONLINE
PSAPTEMP                       ONLINE
PSAPSR3                        ONLINE
PSAPSR3740                     ONLINE
```

```
PSAPSR3USR                     ONLINE
PSAPUNDO002                    ONLINE

8 rows selected.
SQL> alter tablespace PSAPSR3 read only;

Tablespace altered.

SQL> alter tablespace PSAPSR3740 read only;

Tablespace altered.

SQL> alter tablespace PSAPSR3USR read only;

Tablespace altered.

SQL> select tablespace_name, status from dba_tablespaces;

TABLESPACE_NAME                STATUS
------------------------------ ---------
SYSTEM                         ONLINE
SYSAUX                         ONLINE
PSAPUNDO001                    ONLINE
PSAPTEMP                       ONLINE
PSAPSR3                        READ ONLY
PSAPSR3740                     READ ONLY
PSAPSR3USR                     READ ONLY
PSAPUNDO002                    ONLINE

8 rows selected.
```

Listing 11: Setting SAP-specific tablespaces to read-only mode.

Step 2. Create a directory for dump files from the Oracle Database Data Pump Export utility (`expdp`).

Data Pump Export dump files require a directory in which they will be created. It is easier to define it once in Oracle Database rather than to check that all parameter files were created in the correct destination:

```
SQL> create directory migrate_pr1 as '/export/home/oracle/EXPORTFILES';
Directory created.
```

Listing 12: Creating a directory for Data Pump Export dump files.

If you are using a dump directory on a NFS-shared file system, make sure that the correct NFS mounting options (as recommended for Oracle RMAN and the Data Pump feature of Oracle Database) are in effect. Whenever possible, Oracle Direct NFS (dNFS) should be enabled to improve throughput and overall performance of Oracle RMAN backup and restore operations.

Step 3. Export using Data Pump.

The metadata export is performed in two parts—exporting the dictionary data and then exporting the metadata—which can be performed in parallel. Remember that it's the number of database objects (such as tables, indexes, views, and synonyms), rather than the database size, that largely determines the time required for the export process.

Step 3.1. Export dictionary data.

The first part of the Data Pump Export process is to export the dictionary data for the tables owned by SAPSR3. (If you are exporting an older database, also include `owner='ops$SIDadm'`). By excluding unnecessary tables, tablespaces, and statistics, the dictionary data parameter file limits the set of objects that are actually exported:

```
$ cat tts_export_dict.par
full=y
exclude=table:"in( select table_name from dba_tables where owner='SAPSR3')"
exclude=tablespace
exclude=statistics
compression=none
directory=migrate_pr1
dumpfile=dict.dmp
logfile=ttsexpdict.log
status=60
$ expdp parfile=tts_export_dict.par
```

Listing 13: Excluding unnecessary tables, tablespaces, and statistics.

To shorten the Data Pump Export runtime, follow these general recommendations:

- Use an SAP-conforming SPFILE and set an EVENT parameter in the SPFILE.
- Use a reasonable degree of parallelism. As mentioned previously, this value depends on the number of CPU cores available during the export and on the I/O capabilities of the storage system. As a starting point, set the degree of parallelism to the number of available CPU cores. If the export process is still too slow, increase the degree of parallelism while monitoring the utilization of the disk subsystem.

After starting the `expdp` command to export the dictionary data, the Data Pump Export operation begins. At any time, dialog mode can be activated by pressing CTRL+C. Doing so does not stop the export and displays additional information. Some useful subcommands include

- **status**. Provides information about the status of the Data Pump Export process and its worker threads. Setting status to 60, for example, refreshes status information every 60 seconds.
- **cont**. The Data Pump Export process exits dialog mode.
- **EXIT_CLIENT**. The Data Pump Export process continues to work in the background while displaying a shell. This enables logout without discontinuing the export operation.

See "Data Pump Export" in the *Oracle Database Utilities* documentation for more information.

Step 3.2. Export tablespace metadata.

The second part of the metadata export process is to export the metadata of the SAP-specific tablespaces. In the migration example scenario, these tablespaces are PSAPSR3, PSAPSR3740, and PSAPSR3USR. To speed up the export, it might make sense to execute several exports in parallel. To decide how to reasonably allocate the number of export tasks, look at the number of tables in the SAP-specific tablespaces.

```
SQL> select tablespace_name, count(*) from dba_tables group by
tablespace_name order by tablespace_name;

TABLESPACE_NAME                  COUNT(*)
------------------------------ ----------
PSAPSR3                             82474
PSAPSR3740                           5361
PSAPSR3USR                             26
SYSAUX                                461
SYSTEM                                823
```

Listing 14: Checking the number of tables in the SAP-specific tablespaces.

Given the table counts shown in Listing 14, there's no significant benefit from exporting tablespaces in parallel. In this case, all tablespaces can be exported in a single export task:

```
$ cat tts_export_tablespaces.par
directory=migrate_pr1
dumpfile=tbs_all.dmp
transport_tablespaces=PSAPSR3,PSAPSR3740,PSAPSR3USR
transport_full_check=Y
logfile=tts_export_tablespaces.log
```

Listing 15: Creating a single export task.

A single `expdp` job then exports the required tablespaces:

```
$ expdp parfile=tts_export_tablespaces.par
```

Listing 16: Starting a single **`expdp`** job to export the tablespaces.

Listing 17 shows another example.

```
SQL> select tablespace_name, count(*) from dba_tables group by
tablespace_name order by tablespace_name;

TABLESPACE_NAME                  COUNT(*)
------------------------------ ----------
PSAPSR3                             47474
PSAPSR3740                          40361
PSAPSR3USR                             26
SYSAUX                                461
SYSTEM                                823
```

Listing 17: Another example of checking the number of tables.

Given the table counts in Listing 17, it makes sense to do two exports: one for PSAPSR3, and a second for PSAPSR3740 and PSAPSR3USR combined. This approach distributes the load fairly equally. Modify the parameter files as necessary to specify the tablespaces to be exported, as in the following:

```
$ cat tts_export_tablespaces_1.par
directory=migrate_pr1
dumpfile=tbs_PSAPSR3.dmp
transport_tablespaces=PSAPSR3
transport_full_check=Y
logfile=tts_export_tablespaces_1.log

$ cat tts_export_tablespaces_2.par
directory=migrate_pr1
dumpfile=tbs_PSAPSR3740_USR.dmp
transport_tablespaces=PSAPSR3740,PSAPSR3USR
transport_full_check=Y
logfile=tts_export_tablespaces_2.log
```

Listing 18: Creating two export tasks.

Next, start an `expdp` job for each set of tablespaces that needs to be exported, either using separate shells or executing one command after another using the EXIT_CLIENT command:

```
[shell1] $ expdp parfile=tts_export_tablespaces_1.par

[shell2] $ expdp parfile=tts_export_tablespaces_2.par
```

Listing 19: Starting an **expdp** job for two sets of tablespaces.

While the Data Pump Export jobs are running, it's possible to check their status either by attaching to the `expdp` client again, or by issuing an SQL statement in the database:

```
SQL> SELECT owner_name, job_name, rtrim(operation) "OPERATION",
rtrim(job_mode) "JOB_MODE", state, attached_sessions
    FROM dba_datapump_jobs
  ORDER BY 1,2;
```

Listing 20: Checking the status of an export using an SQL statement.

For the tablespace export process, follow these general recommendations:

- Use an SAP-conforming SPFILE during the export and set an SAP-specific EVENT in the SPFILE.
- Oracle Database instances for SAP systems tend to have quite a large number of objects although they are small in size. Because the number of objects determines how long a single export job runs, make sure to distribute the load by defining multiple `expdp` jobs. If distributing the load is not an option (for example, if there are very few tablespaces or a very uneven distribution of tables across tablespaces), long runtimes can result.
- Data Pump Export applies compression to all metadata by default. On older CPUs, this can especially lead to longer runtimes for all `expdp` jobs. To turn off compression, add COMPRESSION=NONE to all parameter files.
- Data Pump Export jobs can be killed with the KILL_JOB command on the command line. This might be necessary if a job does not finish properly. (See My Oracle Support Note 336014.1.)

When using transportable tablespaces to create a system copy, the source system can be put online again after the export processes complete (for all of the dictionary data and tablespace exports). Switch the tablespaces back to read-write mode:

```
SQL> alter tablespace PSAPSR3 read write;
Tablespace altered.
SQL> alter tablespace PSAPSR3740 read write;
Tablespace altered.
SQL> alter tablespace PSAPSR3USR read write;
Tablespace altered.
```

Listing 21: Switching the tablespaces back to read-write mode.

At this point, the SAP system on the source can be restarted using the `startSAP` script.

Import Phase

The next steps take place on the destination system. For the import phase, the destination system must be able to access the data files (outlined in Listing 1 in the "Pre-Export Checks" section) and the metadata dump files that were created by the Data Pump Export jobs.

An NFS file system is a simple way of making file systems containing SAP data available from a source to a destination system. Using NFS requires reasonably fast network connections between the servers. Of course, a storage filer between both systems (or other shared storage accessible by both sides) can also be used. Other methods (such as copying using `scp`, mounting a CIFS file system, or `rsync`) can also accomplish the same goal.

Critical tasks in the import phase require that the following tasks be done:

- Source data files are first converted to destination data files that reside on storage cells on the Oracle SuperCluster.
- A PFILE is created that is suitable for the import operation and for running the database under SAP.
- A new Oracle Database instance is built on the destination system and tablespaces that have not been moved from the source (specifically the SYSTEM, SYSAUX, UNDO, and TEMP tablespaces) are created in the new database.

After constructing the new database, additional steps are necessary to create catalog entries and users, and to perform other post-database-creation steps. The remainder of this article describes the steps for completing these tasks.

Step 1. Copy and convert data files for tablespaces using the modified Oracle RMAN data file conversion script.

The first step of the import phase can be performed even when the tablespaces on the source database are still in read-only mode. In this example, the data files of the source system are made available on the destination system, and then they are imported into the Oracle ASM file system using Oracle RMAN (see Listing 22). The data files are renamed as they are moved into Oracle ASM.

```
RMAN> @conv.rman
Starting conversion at target at 21-MAR-16
using channel ORA_DISK_1
using channel ORA_DISK_2
using channel ORA_DISK_3
using channel ORA_DISK_4
using channel ORA_DISK_5
using channel ORA_DISK_6
using channel ORA_DISK_7
using channel ORA_DISK_8
using channel ORA_DISK_9
using channel ORA_DISK_10
using channel ORA_DISK_11
using channel ORA_DISK_12
channel ORA_DISK_1: starting datafile conversion
input file name=/export/MIGRATION/TTS/PR1/sapdata3/sr3_11/sr3.data11
[...]
converted datafile=+DATAC1/PR3/DATAFILE/psapsr3740.403.907077927
channel ORA_DISK_7: datafile conversion complete, elapsed time: 00:00:07
Finished conversion at target at 21-MAR-16
```

Listing 22: Output from Oracle RMAN data file conversion.

Extract the new data filenames and Oracle ASM grid names of the data files from the output of the conversion process. These names are required later during the metadata import process for the tablespaces.

Step 2. Create a PFILE.

In the sample migration described in this article, a PFILE must be created that is both suitable for the import and later for running the database under SAP. The example PFILE in Listing 23 shows several optimizations that are commonly defined for SAP environments. In particular, settings specified in SAP Note 1888485 are implemented for the import process because they help to optimize performance.

```
PR3.__data_transfer_cache_size=0
PR3.__db_cache_size=28521267200
PR3.__java_pool_size=469762048
PR3.__large_pool_size=335544320
PR3.__oracle_base='/oracle/PR3/121'#ORACLE_BASE set from environment
PR3.__pga_aggregate_target=8589934592
```

```
PR3.__sga_target=32212254720
PR3.__shared_io_pool_size=536870912
PR3.__shared_pool_size=2147483648
PR3.__streams_pool_size=134217728
*._awr_mmon_deep_purge_all_expired=TRUE#SAP RECOMMENDED SETTINGS - SAP NOTE
1888485
*._fix_control='5099019:ON','5705630:ON','6055658:OFF','6120483:OFF','63995
97:ON','6430500:ON','6440977:ON','6626018:ON','6972291:ON','7168184:OFF','7
658097:ON','8937971:ON','9196440:ON','9495669:ON','13077335:ON','13627489:O
N','14255600:ON','14595273:ON','18405517:2'#SAP RECOMMENDED SETTINGS - SAP
NOTE 1888485
*._ktb_debug_flags=8#SAP RECOMMENDED SETTINGS - SAP NOTE 1888485
*._mutex_wait_scheme=1#SAP RECOMMENDED SETTINGS - SAP NOTE 1888485
*._mutex_wait_time=10#SAP RECOMMENDED SETTINGS - SAP NOTE 1888485
*._optim_peek_user_binds=FALSE#SAP RECOMMENDED SETTINGS - SAP NOTE 1888485
*._optimizer_adaptive_cursor_sharing=FALSE#SAP RECOMMENDED SETTINGS - SAP
NOTE 1888485
*._optimizer_aggr_groupby_elim=FALSE#SAP RECOMMENDED SETTINGS - SAP NOTE
1888485
*._optimizer_compute_index_stats=FALSE
*._optimizer_extended_cursor_sharing_rel='NONE'#SAP RECOMMENDED SETTINGS -
SAP NOTE 1888485
*._optimizer_reduce_groupby_key=FALSE
*._optimizer_use_feedback=FALSE#SAP RECOMMENDED SETTINGS - SAP NOTE 1888485
*._securefiles_concurrency_estimate=50#SAP RECOMMENDED SETTINGS - SAP NOTE
1888485
*._suppress_identifiers_on_dupkey=TRUE#SAP RECOMMENDED SETTINGS - SAP NOTE
1888485
*._use_single_log_writer='TRUE'#SAP RECOMMENDED SETTINGS - SAP NOTE 1888485
*.compatible='12.1.0.2.0'
*.control_file_record_keep_time=30
*.control_files='+DATAC1/PR3/cntrlPR3_1.dbf','+RECOC1/PR3/cntrlPR3_2.dbf'
*.db_block_size=8192
*.db_create_file_dest='+DATAC1'
*.db_create_online_log_dest_1='+RECOC1'
*.db_files=1500
*.db_name='PR3'
*.DB_RECOVERY_FILE_DEST='+RECOC1'
*.DB_RECOVERY_FILE_DEST_SIZE=40G
*.db_unique_name='PR3'
*.event='10027','10028','10142','10183','10191','10995 level 2','38068
level 100','38085','38087','44951 level 1024'#SAP RECOMMENDED SETTINGS -
SAP NOTE 1888485
*.filesystemio_options='SETALL'
*.java_pool_size=0
*.log_archive_format='PR3ARC%t_%s_%r.dbf'
*.log_archive_start=FALSE
*.log_buffer=14303232
*.log_checkpoints_to_alert=TRUE
*.open_cursors=800
*.optimizer_adaptive_features=false
*.optimizer_capture_sql_plan_baselines=false
*.optimizer_index_cost_adj=20
*.os_authent_prefix='ops$'
*.parallel_degree_policy='AUTO'
*.PARALLEL_EXECUTION_MESSAGE_SIZE=16384
```

```
*.PARALLEL_MAX_SERVERS=80
*.parallel_min_servers=10
*.pga_aggregate_target=8G
*.processes=850
*.QUERY_REWRITE_ENABLED='FALSE'
*.RECYCLEBIN='OFF'
*.remote_login_passwordfile='exclusive'
*.remote_os_authent=true
*.REPLICATION_DEPENDENCY_TRACKING=FALSE
*.sessions=1700
*.sga_target=30G
*.shared_pool_size=2G
*.star_transformation_enabled='true'
PR3.thread=1
*.timed_statistics=true
*.undo_management='AUTO'
PR3.undo_tablespace='PSAPUNDO0001'
```

Listing 23: Example PFILE with SAP-recommended settings that help to optimize performance.

Step 3. Create the Oracle Database instance on the destination.

After importing and converting data files on the Oracle SuperCluster storage cells and constructing a PFILE, it is time to build a new Oracle Database instance on the destination system. There are several methods that can be used to create the database instance, including scripts and `dbca` (as long as the created database fulfills SAP database requirements); discussion of these two methods follows. Alternatively, if an administrator is more familiar with SAP tools, then the SAP Software Provisioning Manager (SWPM) can be used to create the database instance.

Database Creation Method 1: Using Scripts

The example script in Listing 24 shows how to create the database instance:

```
$ cat createDBscripts/1_createdb.sql
connect / as sysdba
shutdown immediate;
startup nomount;
CREATE DATABASE "PR3"
MAXINSTANCES 8
MAXLOGHISTORY 1
MAXLOGFILES 100
MAXLOGMEMBERS 3
MAXDATA FILES 1000
DATAFILE '+DATAC1' SIZE 700M AUTOEXTEND ON NEXT 10240K MAXSIZE UNLIMITED
EXTENT MANAGEMENT LOCAL
SYSAUX DATAFILE '+DATAC1' SIZE 600M AUTOEXTEND ON NEXT 10240K MAXSIZE
UNLIMITED
```

```
DEFAULT TEMPORARY TABLESPACE "PSAPTEMP" TEMPFILE '+DATAC1' SIZE 1000M
AUTOEXTEND ON NEXT 640K MAXSIZE UNLIMITED
UNDO TABLESPACE "PSAPUNDO0001" DATAFILE '+DATAC1' SIZE 200M AUTOEXTEND ON
NEXT 5120K MAXSIZE UNLIMITED
CHARACTER SET UTF8
NATIONAL CHARACTER SET UTF8
LOGFILE GROUP 1 ('+RECOC1') SIZE 4G, GROUP 2 ('+RECOC1') SIZE 4G, GROUP 3
('+RECOC1') SIZE 4G, GROUP 5 ('+RECOC1') SIZE 4G, GROUP 6 ('+RECOC1') SIZE
4G
USER SYS IDENTIFIED BY "examplepw" USER SYSTEM IDENTIFIED BY "examplepw";
```

Listing 24. First example script, which creates the database instance.

The database layout that the script in Listing 24 defines can be adjusted to specific requirements and is only an example. Make sure that the character set and national character set have the same values identified in the "Pre-Export Checks" phase of the migration.

The second example script, shown in Listing 25, creates the catalog entries for the objects that the first script (Listing 24) creates:

```
$ cat createDBscripts/2_createcatalog.sql
connect sys/examplepw as SYSDBA
set echo on
spool 2_createcatalog.log append
@?/rdbms/admin/catalog.sql;
@?/rdbms/admin/catblock.sql;
@?/rdbms/admin/catproc.sql;
@?/rdbms/admin/catoctk.sql;
@?/rdbms/admin/owminst.plb;
connect system/examplepw
@?/sqlplus/admin/pupbld.sql;
spool off
```

Listing 25. Second example script, which creates the catalog entries.

The third example script, shown in Listing 26, locks all accounts that are not used by SAP, making the installation more secure:

```
$ cat createDBscripts/3_lockaccounts.sql
connect / as sysdba
shutdown immediate;
startup;
SET VERIFY OFF
set echo on
spool lockaccounts.log append
BEGIN
FOR item IN ( SELECT USERNAME FROM DBA_USERS WHERE ACCOUNT_STATUS IN
('OPEN', 'LOCKED', 'EXPIRED') AND USERNAME NOT IN ( 'SYS','SYSTEM') ) LOOP
dbms_output.put_line('Locking and Expiring: ' || item.USERNAME);
```

```
execute immediate 'alter user ' || sys.dbms_assert.enquote_name(
sys.dbms_assert.schema_name( item.USERNAME),false) || ' password expire
account lock' ;
END LOOP;
END;
/
spool off
```

Listing 26. Third example script, which locks all accounts.

The fourth script, shown in Listing 27, looks for the latest installed Patch Set Update (PSU) and applies it to the newly created database:

```
$ cat createDBscripts/4_postcreate.sql
SET VERIFY OFF
connect sys/examplepw as sysdba
shutdown immediate;
startup;
set echo on
spool postDBCreation.log append
@?/rdbms/admin/catbundle.sql psu apply;
select 'utl_recomp_begin: ' || to_char(sysdate, 'HH:MI:SS') from dual;
execute utl_recomp.recomp_serial();
select 'utl_recomp_end: ' || to_char(sysdate, 'HH:MI:SS') from dual;
shutdown immediate;
connect sys/examplepw as sysdba
startup mount;
alter database archivelog;
alter database open;
connect sys/secret as sysdba
set echo on
create spfile='spfilePR3.ora' FROM pfile;
shutdown immediate;
spool off
```

Listing 27. Fourth example script, which looks for the latest PSU.

Database Creation Method 2: Using `dbca`

For many database administrators, `dbca` is the tool of choice for creating new databases. It is generally possible to use `dbca` to construct Oracle Database instances for SAP, although it is not the method recommended in SAP Notes. When using `dbca`, it is important to define the character sets the same as they are defined in the source database. The database creation script in Listing 24 can be used as a starting point for the layout of the data files, but must be adjusted to specific needs.

Step 4. Create the roles SAPCONN and SAPDBA.

After creating the Oracle Database instance, the SAP-specific roles SAPCONN and SAPDBA must be created. SAP Notes 834917 and 134592 include the latest bug fixes, as well as two ready-to-use scripts for creating these roles.

Step 5. Import the transported tablespaces into the newly created database.

After setting up the database, it is time to make it aware of the imported data files. (Currently, the database cannot do anything with these files, because the tablespaces they belong to are not yet defined.)

It's recommended to import the tablespaces first and then to import the dictionary data. If the dictionary data is imported first, there will be errors, particularly when importing views. These errors are invalid because they refer to tables that do not yet exist. It's safe to ignore them, but they can make log files difficult to read, making "real" errors much harder to detect.

First, create a directory in the database that points to the location of the Data Pump Export `dmp` files:

```
SQL> create directory pr1migration as '/export/software/migrationpr1/';
Directory created
```

Listing 28: Creating a directory that points to the **dmp** files.

Next, create the SAPSR3 user manually. This step is necessary because the tablespaces are being imported prior to the data dictionary. It is possible to import the dictionary first and then the tablespaces, but there are some serious disadvantages in doing so.

```
SQL> create user SAPSR3 identified by <password>;
SQL> grant connect, resource, sapdba, sapconn, unlimited tablespace to
sapsr3;
Grant succeeded
```

Listing 29: Creating the SAPSR3 user manually.

After creating the SAPSR3 user and granting it the SAPDBA and SAPCONN roles, prepare the parameter file for importing the tablespaces. This requires a list of the previously created data filenames on the destination. Using the output of the Oracle RMAN conversion script (refer back to Listing 22), this list can be created by sorting the names of the converted data files. It might be more practical to use `asmcmd` to sort the list of the data files to create the parameter file, as in this sequence:

```
EXPORT ORACLE_SID=+ASM1
ORACLE_HOME=/u01/app/12.1.0.2/grid
asmcmd ls +DATAC1/PR3/DATAFILE | grep PSAPS | \
nawk -v q="'"  -v s="+DATAC1/PR3/DATAFILE/"
'BEGIN { printf "%s","transport_data files=" } { i=k; if (k!="") print q s
i q","; k=$0 } END
{ print q s k q }
```

Listing 30: Using **asmcmd** to sort the data file list.

The parameter file can be quite long, particularly when importing a large database with a huge number of small data files. An excerpt of the parameter file might look like this:

```
$ cat ttsimptbs.par
directory=pr1migration
dumpfile=tbs_nocompress.dmp
transport_data files='+DATAC1/PR3/DATAFILE/PSAPSR3.361.907077867',
'+DATAC1/PR3/DATAFILE/PSAPSR3.362.907077867',
'+DATAC1/PR3/DATAFILE/PSAPSR3.363.907077867',
[...]
'+DATAC1/PR3/DATAFILE/PSAPSR3USR.404.907077927'
logfile=ttsimptbs_nocompress.log
```

Listing 31: Excerpt of the parameter file.

After creating a parameter file, start the actual import of the tablespaces:

```
$ impdp PARFILE=ttsimptbs.par
```

Listing 32: Starting the tablespace import.

Step 6. Import the dictionary data.

The second import is the import of the dictionary data. It is done with a very simple and short parameter file:

```
$ cat ttsimpdict.par
directory=pr1migration
dumpfile=dict_nocompress.dmp
logfile=ttsimpdict.log

$ impdp PARFILE=ttsimpdict.par
```

Listing 33: Importing the dictionary data.

Final Steps

After the import, validate the database integrity and prepare the database for use.
Step 1. Check the created tablespaces.

Check the integrity of the database after the import process by using Oracle RMAN:

```
$ rman target /
Recovery Manager: Release 12.1.0.2.0 - Production on Wed Mar 23 09:30:51
2016
```

```
Copyright (c) 1982, 2014, Oracle and/or its affiliates.  All rights
reserved.

connected to target database: PR3 (DBID=1089922202)

RMAN> validate database;

Starting validate at 23-MAR-16
using target database control file instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=1303 device type=DISK
channel ORA_DISK_1: starting validation of datafile
channel ORA_DISK_1: specifying datafile(s) for validation
input datafile file number=00018
name=+DATAC1/PR3/DATAFILE/psapsr3.363.907077867
[...]
input datafile file number=00047
name=+DATAC1/PR3/DATAFILE/psapsr3usr.404.907077927
channel ORA_DISK_1: validation complete, elapsed time: 00:01:05
List of Data files
=================
File Status Marked Corrupt Empty Blocks Blocks Examined High SCN
---- ------ -------------- ------------ --------------- ----------
1    OK     0              21322        423680          972516553
  File Name: +DATAC1/PR3/DATAFILE/system.406.907086685
  Block Type Blocks Failing Blocks Processed
  ---------- -------------- ----------------
  Data       0              78059
  Index      0              53323
  Other      0              270976

[...]

File Status Marked Corrupt Empty Blocks Blocks Examined High SCN
---- ------ -------------- ------------ --------------- ----------
47   OK     0              2393         2560            970497994
  File Name: +DATAC1/PR3/DATAFILE/psapsr3usr.404.907077927
  Block Type Blocks Failing Blocks Processed
  ---------- -------------- ----------------
  Data       0              18
  Index      0              2
  Other      0              147

channel ORA_DISK_1: starting validation of datafile
channel ORA_DISK_1: specifying datafile(s) for validation
including current control file for validation
including current SPFILE in backup set
channel ORA_DISK_1: validation complete, elapsed time: 00:00:01
List of Control File and SPFILE
===============================
File Type    Status Blocks Failing Blocks Examined
------------ ------ -------------- ---------------
SPFILE       OK     0              2
Control File OK     0              1328
Finished validate at 23-MAR-16
```

Listing 34: Checking the integrity of the database after the import.

Step 2. Switch the tablespaces into read-write mode.

```
SQL> alter tablespace PSAPSR3 read write;
Tablespace altered.
SQL> alter tablespace PSAPSR3740 read write;
Tablespace altered.
SQL> alter tablespace PSAPSR3USR read write;
Tablespace altered.
SQL> select tablespace_name,status from dba_tablespaces;
TABLESPACE_NAME    STATUS
---------------------------------------
SYSTEM          ONLINE
SYSAUX          ONLINE
PSAPUNDO0001    ONLINE
PSAPTEMP        ONLINE
PSAPSR3         ONLINE
PSAPSR3740      ONLINE
PSAPSR3USR      ONLINE
```

Listing 35: Switching the tablespaces to read-write mode.

Set the default tablespace and temporary tablespace for user SAPSR3:

```
SQL> alter user sapsr3 default tablespace psapsr3usr temporary tablespace
psaptemp;
User altered.
```

Listing 36: Setting the tablespaces for user SAPSR3.

On some SAP systems, the default tablespace for user SAPSR3 is PSAPSR3. If this is the case on your system, change it to PSAPSR3 instead of PSAPSR3USR.

The Oracle Database migration is now complete and the database instance is usable. There are, however, some additional tasks that are necessary to finalize the migration of the SAP system. Since these are „normal“ post-migration tasks, these are not discussed in this paper.

Performance Considerations

Both the export and import processes can take quite long if the system is not configured properly. In an internal test of an Oracle Database instance for SAP containing around 250,000 objects, export runtimes without optimizations decreased from about 17 hours to 25 minutes after applying the proper optimizations.

Critical factors that impact export performance include

- Settings in the PFILE of the source database. Check that all settings match the recommended settings in the SAP Notes. In particular, the lack of an event `38068 level 100` entry resulted in a substantial performance impact in testing.
- Fast file systems with the correct mount options. Check the mount options for the file system where the Data Pump Export dump (`*.dmp`) files are stored.

In addition, correctly configured parameter files speed up both the export and import processes:

- Compression should be turned off. Metadata cannot be compressed much at all, so a lot of time can be saved by disabling compression.
- Index statistics can be excluded. On the destination platform, database statistics must be re-created, so there is no reason to transport unneeded statistics.

See Also

Additional information relative to using the transportable tablespaces method of Oracle Database migration can be found in the relevant Oracle Database documentation, including

- Information about using Oracle Database Data Pump: *Oracle Database Utilities* (for Oracle Database 11*g*) or *Oracle Database Utilities* (for Oracle Database 12*c*)
- Information about using Oracle RMAN and transportable tablespaces: *Oracle Database Backup and Recovery User's Guide* (for Oracle Database 11*g*) or *Oracle Database Backup and Recovery User's Guide* (for Oracle Database 12*c*)

Kontaktadressen:

Andris Perkons
Oracle Deutschland B.V. & Co KG
Hamborner Str. 51
40472 Düsseldorf

Telefon:	+49 211 74839791
E-Mail	andris.perkons@oracle.com

Jan Brosowski
Oracle Deutschland B.V. & Co KG
Altrottstr. 31
69190 Walldorf

Telefon:	+49 6227 356201
E-Mail	jan.brosowski@oracle.com
Internet:	www.oracle.com