A Complete Cross Platform Database Migration Guide Using Import and Export Utility

(HP-UX on PA-RISC to Linux on x86 Architecture)

Prepared by:
ViSolve Migration Team
4010, Moorpark Avenue
#205, San Jose,
California 95117

+1- 408-850-2243

migration_engg@visolve.com
www.visolve.com

November 15, 2011

Software Development / Support Lab: # 1, Rukmani Nagar, Ramanathapuram, Coimbatore - 641 045, TN, INDIA.

Disclaimers of Liability:

Whilst every effort has been made to ensure the accuracy of all information and statements contained in this document, the functionality, the service levels, performance and capabilities referred to are best estimates and best practices only, based on our understanding.

The information in the document is believed to be precise in all respects to the best of ViSolve's knowledge at the time of publication and is subject to change without prior notice.

ViSolve is not legally liable for any errors or opinions that appear in this document.

All trademarks and logos used in this document are the property of their respective owners.

Table of Contents

Introduction

A migration is required when you wish to move an existing application system to another technology and/or platform. Migration includes moving objects, data and application code from one device to another—preferably without disrupting or disabling the active applications and then redirecting all input/output (I/O) activity to the new device.

There are a variety of circumstances that might cause an organization to undertake a data migration, including:

1. Server or storage technology replacement or upgrade
2. Server or storage consolidation
3. Relocation of the data center
4. Server or storage equipment maintenance, including workload balancing or other performance-related maintenance.

Document Overview

This document explains the steps to migrate an Oracle database from a big-endian platform like HP-UX on PA-RISC to a little-endian platform like Linux on X86. This document also provides the complete cross platform database migration scenarios using Data Pump utility (Expdp/Impdp) for small organizations.

Key benefits of migration Fast Performance

1. Improved Management Restart
2. Fine-Grained Object Selection
3. Monitoring and Estimating Capability
4. Network Mode

Challenges to consider when migrating data

1. Performance
2. Primary volume/source data protection
3. Different storage medium
4. Different hardware platforms
5. Application downtime
6. Data corruption, missing data or data loss
7. Technical compatibility issues.

Assumptions Made

Oracle version on source and target

1. Oracle database used in both the platforms is Oracle 11g.
2. Oracle database is installed in both the source and the target platforms.

Source and target platforms

1. Source machine is big-endian platform (HPUX on PA-RISC)
2. Target machine is little-endian platform (Linux on X86)

Compatibility criteria

Different versions of the import utility are upwards compatible. This means that one can take an export file created from an old export version, and import it using a later version of the import utility. This is quite an effective way of upgrading a database from one release of Oracle to the next.

Service status of the platform

The status of the source and target is assumed to be in offline or maintenance mode and not doing any live transaction during the migration.

Data Pump Export and Import

Prior to oracle 10G, the logical backup was taken through EXP/IMP utility, which was the basic export/import utility till oracle 9i. Oracle has come with more powerful logical backup tool in oracle 10G. As we know about datapump export/import utility.

Data Pump takes the old export and import utilities one step further, you can have total control over the job running (stop it, pause it, check it, restart it). Data pump is a server side technology and it can transfer large amounts of data very quickly using parallel streams to achieve maximum throughput, they can be 15-45% faster than the older import/export utilities.

Advantages of using data pump

1. ability to estimate jobs times
2. ability to restart failed jobs
3. perform fine-grained object selection
4. monitor running jobs
5. directly load a database from a remote instance via the network
6. remapping capabilities
7. improved performance using parallel executions

Uses of Data Pump Utility

1. migrating databases
2. copying databases
3. transferring oracle databases between different operating systems
4. backing up important tables before you change them
5. moving database objects from one tablespace to another
6. transporting tablespace's between databases
7. reorganizing fragmented table data
8. extracting the DDL for tables and other objects such as stored procedures and packages

Data pump working procedure

The Master Control Process (MCP), has the process name DMnn, only one master job runs per job which controls the whole Data Pump job, it performs the following

1. create jobs and controls them
2. creates and manages the worker processes
3. monitors the jobs and logs the process
4. maintains the job state and restart information in the master table (create in the users schema running the job)
5. manages the necessary files including the dump file set

The master process creates a master table which contains job details (state, restart info), this table is created in the users schema who is running the Data Pump job. Once the job has finished it dumps the table contents into the data pump file and deletes the table. When you import the data pump file it re-creates the table and reads it to verify the correct sequence in which it should import the various database objects.

The worker process is named DWnn and is the process that actually performs the work, you can have a number of worker process running on the same job (parallelism). The work process updates the master table with the various job statuses.

The shadow process is created when the client logs in to the oracle server it services data pump API requests, it creates the job consisting of the master table and the master process.

The client processes are the expdp and impdp commands

Steps for export the data
Export data from the source machine

Connect to the database

1. Login as oracle user on the source machine
2. Export the environment variables like ORACLE_HOME, ORACLE_SID, PATH appropriately.

   ```
   $ export ORACLE_HOME=/u02/product/11.1.0/db_1
   $ export ORACLE_SID=sample
   $ export PATH=$PATH:$ORACLE_HOME/bin
   ```

3. Login to oracle as dba user

   ```
   $ sqlplus / as sysdba
   ```

4. Start the database

   ```
   SQL> startup
   ```

5. Find the required space needed for creating the database and importing the data in the target machine using the following query.

   ```
   SQL> select a.data_size+b.temp_size+c.redo_size+d.controlfile_size “total_size in MB” from ( select
   sum(bytes)/1024/1024 data_size
   from dba_data_files ) a,
   ( select nvl(sum(bytes),0)/1024/1024 temp_size
   from dba_temp_files ) b,
   ( select sum(bytes)/1024/1024 redo_size
   from sys.v_$log ) c,
   ( select sum(BLOCK_SIZE*FILE_SIZE_BLKS)/1024/1024 controlfile_size
   from v$controlfile) d;
   ```

6. Exit from the database

   ```
   SQL> exit
   ```

7. Check the availability of free space on the disks, t hold the data that will be exported.

   ```
   $ bdf
   ```

8. Now create a folder 'backup' in the disk that have sufficient space and store the backup files.

Note: Here we have the directory '/u02' with sufficient space to store the exported data. Hence we have created a folder 'backup' in /u02 owned by oracle

Login as root and create the folder and give the permission and ownership appropriately

```
$ su – root
# mkdir /u02/backup
# chmod 700 /u02/backup
# chown oracle:dba /u02/backup
```

Setup the export database directory

1. Login to oracle as dba user

   ```
   $ sqlplus / as sysdba
   ```

2. Create a directory in the source database

   ```
   SQL> create directory expdp_dir as '/u02/backup';
   ```

3. Grant permission to that directory

   ```
   SQL> grant read, write on directory expdp_dir to system;
   ```

4. Exit the Database

There are four sets of files/values that need to be extracted to ease the migration procedure. They are,

1. Datadump file
2. Configuration parameter file
3. Initialization Parameters file

expdp/impdp utilities

These utilities are used to transfer data from one oracle database to another oracle database. The Export tool is used to export data from source database to a data dump file and the Import tool is used to load data into the target database from the data dump file.

The exported data dump file contains objects in the following order:

1. Type definitions
2. Table definitions
3. Table data
4. Table indexes
5. Integrity constraints, views, procedures, and triggers
6. Bitmap, function-based, and domain indexes

Obtaining data dump file

The data dump file can be obtained at three levels based on the requirement.

1. Full database level export
2. Schema level export
3. Table level export

In this document we are going to concentrate only full database export method.

Full database export

Full database export extracts all the object definitions and data from a complete database. Import may then reload a complete database or selectively load objects or sets of objects from the full database export file.

A full export is specified using the FULL parameter. In a full database export, the entire database is unloaded. This mode requires that you have the EXP_FULL_DATABASE role or DBA

privileges $ cd $ORACLE_HOME/bin
$ ./expdp system/system DIRECTORY=expdp_dir
DUMPFILE=data.dmp logfile=data.log full=y

Result:

Job "SYSTEM"."SYS_EXPORT_FULL_01" successfully completed

In some cases where the Database is in Terabytes the above command will not feasible since the dump file size will be larger than the operating system limit, and hence export will fail. In this situation you can create multiple dump files by using the following command

$ ./expdp system/sys DIRECTORY=expdp_dir DUMPFILE=expdp_dir:data%U.dmp parallel=4 filesize=40M logfile=expnew.log full=y

This will create multiple dump files named data01.dmp, data02.dmp, data03.dmp and so on. The FILESIZE parameter specifies how much larger the dump file should be.

%U is used when the one of the files reached its specified size, oracle creates another dumpfile in the file system, provided you have enough disk space.

%U specified along with the PARALLEL parameter, then one file for each is initially created. More files are created as they are needed based on how much data is being exported and how many parallel processes are given work to perform during the job.

The `FILESIZE` parameter has a maximum value equal to the maximum value that can be stored in 64 bits.

The maximum size of the FILESIZE value is listed here.

Operating System	Release of Oracle Server	Maximum Size
Any	Prior to 8.1.5	2 gigabytes
32-bit	8.1.5	2 gigabytes
64-bit	8.1.5 and later	Unlimited
32-bit with 32-bit files	Any	2 gigabytes
32-bit with 32-bit files	8.1.5 and later	Unlimited

Note:

The maximum value that can be stored in a file is dependent on your operating system. You should verify this maximum value in your Oracle operating system-specific documentation before specifying FILESIZE. You should also ensure that the file size you specify for Export is supported on the system on which Import will run

Collect the required Configuration parameters

A few configuration parameters need to be collected from the source machine database to create the database on the target.

The required configuration parameters are

1. Characterset
2. List of tablespaces,
3. Size of tablespaces,
4. Default tablespaces,
5. DB block size,
6. Undo management,
7. Undo tablespaces,

8. Users info,
9. Profile details,
10. Temporary tablespaces

Given below are the set of queries to collect the configuration parameters and store them on a file configuration.txt under /u02/backup.

```
SQL> spool /u02/backup/configuration.txt
SQL> select value from NLS_DATABASE_PARAMETERS
where parameter = 'NLS_CHARACTERSET';
 SQL> select * from v$tablespace
SQL> select BLOCK_SIZE from dba_tablespaces where
tablespace_name = 'tablespacename'
SQL> select PROPERTY_VALUE from database_properties
where property_name = 'DEFAULT_PERMANENT_TABLESPACE'
SQL> select value from v$parameter where name =
'db_block_size'
SQL> select value from v$parameter where name =
'undo_management'
SQL> select value from v$parameter where name =
'undo_tablespace'
SQL> select BLOCK_SIZE from dba_tablespaces where
tablespace_name = 'undo_tspacename'
SQL> select FILE_NAME, BYTES, AUTOEXTENSIBLE, MAXBYTES,
INCREMENT_BY from dba_data_files where tablespace_name='undo_tspace name'
SQL> select TABLESPACE_NAME, INITIAL_EXTENT, NEXT_EXTENT,
MIN_EXTENTS, MAX_EXTENTS, PCT_INCREASE, MIN_EXTLEN, STATUS, CONTENTS, LOGGING,
EXTENT_MANAGEMENT, ALLOCATION_TYPE, BLOCK_SIZE, FORCE_LOGGING, SEGMENT_SPACE_MANAGEMENT
from dba_tablespaces
SQL> select USERNAME, USER_ID, PASSWORD, ACCOUNT_STATUS,
LOCK_DATE, EXPIRY_DATE, DEFAULT_TABLESPACE, TEMPORARY_TABLESPACE, CREATED, PROFILE,
INITIAL_RSRC_CONSUMER_GROUP, EXTERNAL_NAME INITIAL_RSRC_CONSUMER_GROUP from dba_users
SQL> select PROFILE from dba_profiles group by profile
SQL> select RESOURCE_NAME, RESOURCE_TYPE, LIMIT from
dba_profiles where profile=’profile_name from above query'
SQL>spool off
```

Copy the initialization parameters

The default location of the oracle initialization parameter files is $ORACLE_HOME/dbs . Copy the required parameter file to the backup directory on the source machine.

```
$ cp $ORACLE_HOME/dbs/initSAMPLE.ora ~/u02/backup
```

Import data on the target machine

Transfer backup directory to target

Copy the backup directory (~/u02/backup) from source machine to target over the network or removable media using the following steps.

1. Login as oracle user on the target machine.
2. Inorder to restore the backup of the database taken, check for the availability of free space on the disks using the below command.

   $ df

3. Now create a folder 'backup' in the disk that have sufficient space and store the backup file.

   Note: Here we have the directory '/u02' with sufficient space to store the exported data. Hence we have created a folder 'backup' in /u02 owned by oracle. In order to avoid confusion in restoring the database, use the same directory names as in the source machine.
   Login as root and create the folder and give the permission and ownership appropriately
   $ su – root
   # mkdir /u02/backup
   # chmod 700 /u02/backup
   # chown oracle:dba /u02/backup

4. Copy the source backup directory files (~/u02/backup) to the target directory /u02/backup.

Modify the Initialization parameter file

Copy the oracle initialization parameter file initSAMPLE.ora from the backup directory that was copied from the source machine(/u02/backup) to a file named initSAMPLE.ora in the $ORACLE_HOME/dbs/ directory on target machine and modify the required directives based on the source database. The correct values are available on the configuration.txt file on the /u02/backup directory

initSAMPLE.ora file

```
db_cache_size=281018368
java_pool_size=20971520
large_pool_size=4194304
pga_aggregate_target=209715200
sga_target=637534208 shared_pool_size=314572800
streams_pool_size=8388608
*.audit_file_dest='/u02/admin/SAMPLE/adump'
*.audit_trail='db'
*.compatible='11.1.0.0.0'
*.control_files='/u02/data/SAMPLE/control1.ctl','/u02/data/SAMPLE/control2.ctl','/u02/data/SAMPLE1/control
3.ctl'
*.db_block_size=8192 *.db_domain=''
*.db_name='SAMPLE'
*.diagnostic_dest='/u02/data'
*.dispatchers='(PROTOCOL=TCP) (SERVICE=RANXDB)'
*.memory_target=845152256
*.open_cursors=300
*.processes=150
```

Create the Database

Steps to create a database:

1. Export the environment variables like ORACLE_HOME, ORACLE_SID, PATH appropriately.

```
$ ORACLE_HOME=/u02/product/11.1.0/db_1
$ ORACLE_SID=SAMPLE
$ PATH=$PATH:$ORACLE_HOME/bin
$ export ORACLE_HOME ORACLE_SID PATH
$ cd $ORACLE_HOME/bin
$ sqlplus /nolog SQL> create spfile from
pfile='/u02/product/11.1.0/db_1/dbs/initSAMPLE.ora'
SQL> startup nomount
```

2. Create a database using following sql statement

```
SQL> CREATE DATABASE SAMPLE USER SYS IDENTIFIED BY sys USER
SYSTEM IDENTIFIED BY system LOGFILE GROUP 1 ('/u02/data/SAMPLE/redo01.log')SIZE 100M, GROUP 2
('/u02/data/SAMPLE/redo02.log')SIZE 100M, GROUP 3 ('/u02/data/SAMPLE/redo03.log')SIZE 100M
MAXLOGFILES 5 MAXLOGMEMBERS 5 MAXLOGHISTORY 1 MAXDATAFILES 100 MAXINSTANCES 1 CHARACTER
SET US7ASCII NATIONAL CHARACTER SET AL16UTF16 DATAFILE '/u02/data/SAMPLE/system01.dbf' SIZE 320M
REUSE SYSAUX DATAFILE '/u02/data/SAMPLE/sysaux01.dbf' SIZE 325M REUSE EXTENT MANAGEMENT LOCAL
DEFAULT TEMPORARY TABLESPACE tempts1 TEMPFILE '/u02/data/SAMPLE/temp01.dbf'SIZE 200M REUSE
UNDO TABLESPACE undotbs DATAFILE '/u02/data/SAMPLE/undotbs01.dbf'SIZE 200M REUSE AUTOEXTEND ON
MAXSIZE UNLIMITED;
```

Note: The temporary tablespace and undo tablespace names must be same as the source database tablespace names.

Post database creation steps

```
SQL> connect sys as sysdba
Run the following scripts to complete the database creation.
SQL> @/<ORACLE_HOME>/rdbms/admin/catalog.sql
SQL> @/<ORACLE_HOME>/rdbms/admin/catproc.sql
```

catalog.sql: *Creates the views of the data dictionary tables, the dynamic performance views, and public synonyms for many of the views. Grant PUBLIC access to the synonyms.*

catproc.sql: Runs all scripts required for or used with PL/SQL

```
SQL> connect system/manager
SQL> @/<ORACLE_HOME>/sqlplus/admin/pupbld.sql
```

The "system" user might also want to run pupbld.sql. pupbld.sql creates a table that allows to block someone from using sqlplus.

For OLAP Related issues, if source database is OLAP enabled then run the following sql to solve those issues.

```
SQL> connect sys as sysdba/sys
SQL> @/<ORACLE_HOME>/olap/admin/olapiboo.plb
```

Create a default tablespace other than the table spaces that we created while creating the database query.

```
SQL> connect sys as sysdba/sys
```

Create a user tablespace to be assigned as the default tablespace for users.

```
SQL> CREATE TABLESPACE users LOGGING DATAFILE
```

'/u02/data/SAMPLE/users01.dbf' SIZE 250M REUSE AUTOEXTEND ON NEXT 1280K MAXSIZE UNLIMITED EXTENT MANAGEMENT LOCAL;

Connect to the database

1. Login as oracle user on the target machine.
2. Export the environment variables like ORACLE_HOME, ORACLE_SID, PATH appropriately.

```
$ ORACLE_HOME=/u02/product/11.1.0/db_1
$ ORACLE_SID=SAMPLE $ PATH=
$PATH:$ORACLE_HOME/bin
$ export ORACLE_HOME ORACLE_SID PATH
```

3. Login to oracle as dba user

```
$ sqlplus / as sysdba
```

4. Start the database

```
SQL> startup
```

Setup the export database directory

1. Create a directory in target database

```
SQL> create directory expdp_dir as '/u02/backup';
```

2. Grant the permission to that directory

```
SQL> grant read, write on directory expdp_dir to system;
```

Steps for Import the data

Full database import

The full import mode loads the entire contents of the source (export) dump file to the target database. However, you must have been granted the IMP_FULL_DATABASE role on the target database. The data pump import is invoked using the impdp command in the command line with the FULL parameter specified in the same command line.

```
$ cd $ORACLE_HOME/bin
$ ./impdp system/system DIRECTORY=expdp_dir
DUMPFILE=data.dmp logfile=data.log full=y
```

Result:

```
Job "SYSTEM"."SYS_IMPORT_FULL_01" successfully completed
```

In some cases where the Database is in Terabytes and we exported the data as multiple dump files , then you can import the multiple dump files using the following command

```
$ ./impdp system/sys DIRECTORY=expdp_dir DUMPFILE=expdp_dir:data%U.dmp parallel=4 logfile=expnew.log
full=y
```

This will import multiple dump files in to the database.

Verification

Verify the target database whether the data imported or not. Use bellow queries to check the exported data and imported data in both source and target database.

```
SQL> select count(*) from dba_users;
SQL> select count(*) from dba_objects;
SQL> select count(*) from user_tables;
SQL> select a.data_size+b.temp_size+c.redo_size+d.controlfile_size "total_size in MB" from ( select
sum(bytes)/1024/1024 data_size
from dba_data_files ) a,
( select nvl(sum(bytes),0)/1024/1024 temp_size
from dba_temp_files ) b,
( select sum(bytes)/1024/1024 redo_size
from sys.v_$log )
c, ( select sum(BLOCK_SIZE*FILE_SIZE_BLKS)/1024/1024 controlfile_size from v$controlfile) d;
```

These queries will display the list of users, objects and size of the database.

Things to Lookout for

1. Volume-Time Relationship

   The time required to extract the data is dependent on the size of the data. Extracting and reinserting high volumes of data can take more time for large databases.

2. Ongoing Transactions

   Ongoing changes are not captured by Export, nor is there a way to demarcate the boundaries that allow the incremental data to be re-exported. This effectively implies application downtime during the export/import process.

3. Recoverability

   Failures during export and/or import are not easy to recover.

Conclusion

Data migration is a routine part of IT operations in today's business environment. Even so, it often causes major disruptions as a result of downtime or application performance problems, and it can severely impact budgets. To prevent these problems, organizations need a consistent and reliable methodology that enables them to plan, design, migrate and validate the migration. Further, they need migration procedures that support their specific migration requirements, including operating systems, storage platforms and performance. In addition, migration products that maintain continuous data availability during the migration without affecting performance are desirable.

The migration procedures mentioned in this document provide a server-side infrastructure and new high-speed, parallel Export and Import utilities for highly efficient bulk data and metadata movement between databases. We can now move data and metadata between databases between different platforms faster and easier than ever before, and this methodology helps you meet your various organization needs.