Reducing Downtime

Using incremental backups and x-platform TTS

Martin Bach, Enkitec

Executive summary

A little known note on My Oracle Support explains how to perform a cross-platform migration using incremental backups and cross-platform transportable tablespaces. This talk explains how it works in detail.

The novelty in this approach is that the whole process is scripted. It resembles the established process of transportable tablespaces, namely the conversion of the data files to the destination platform, and the end result will be the same. The downtime required will be a lot less though.

This paper explains migration strategies to Exadata from a big endian platform.

Table of Contents

Why should you care about the process?

The IT industry is undergoing profound changes. A service-oriented architecture based on suitable deployment mechanisms and private clouds coupled with a high degree of automation is gradually replacing proprietary and expensive solutions.

The following trends have been identified:

- Traditional RISC based architecture is replaced by Intel x86-64
- Linux is becoming the operating system of choice for many deployments

The price advantage Intel architecture has over RISC based hardware is undeniable. Over the last 3 years Intel Xeon processors based on the Sandy Bridge architecture provided the most cost efficient processors, especially when considering their single thread performance.

AMD has gone a similar way as Sun/Oracle did with the unsuccessful T3-series processor: they all expected massively parallelized applications to be the norm when they presented the 6200 series Opterons. Unfortunately single thread performance is needed more, not the least because of the licensing model of many Enterprise Software Packages.

Why are RISC based systems losing market share?

The vast majority of workloads do not require the benefits offered by the traditional RISC based vendors. A two socket system based on Sandy Bridge Xeon processors such as offered by the (Exadata) X3-2 servers packs a dual socket E5-2690 processor offering 16 cores which are hyper-threaded. The E5 Sandy Bridge processors can address 768GB of memory.

Intel has just released the Ivy Bridge EP platform which replaces the current Sandy Bridge generation. The top of the range dual socket processor increased the core count to 12 (up from 8) for a total of 24 hyper-threaded cores. Addressable memory increased to 1.5 TB. A few generations ago, Xeon 5500-based systems had dual socket capability with 6 cores each.

The number of cores per socket cannot be the only reason for the dominance of Intel based systems though. Intel benefits greatly from economies of scale: their server processors are based on a micro-architecture that is already out in the field in consumer products. Roughly a year later Intel can release the corresponding server package. The massive scale of Intel's production must lead to cost advantages over vendors creating enterprise only processors.

Implications

Many large organisations are reviewing their hardware platform strategy. The review is most likely driven by cost constraints. Instead of having the luxury of supporting three or more platforms at the same time for a multitude of storage solutions as well as applications engineering can focus on a much smaller subset. The whole discussion around Database as a Service (DBaaS) revolves around these.

Engineering can focus on building blocks, the equivalent of Oracle's "engineered systems". If resources are scarce for such a project Engineered Systems could be directly ordered from Oracle, freeing resources to focus to work on the more pressing problems.

Possible building blocks if Exadata is not selected as the primary consolidation target:

- Gold: 8-way Xeon, flash storage?
- Silver: Four-way x86-64, maybe with flash storage
- Bronze: Two-way x86-64

The remainder of this paper assumes that Exadata is the migration target for the RISC based system.

Migrating to Linux

Migration to Linux can be simple or complex, depending on your source platform. Endianness is certainly the key factor. Migrating from little endianness platforms to Exadata is simpler than the same migration from big endianness platforms.

Important platforms employing the big endianness are the traditional UNIX servers based on RISC technology such as AIX, HP-UX and Solaris. Exadata is based on Intel Xeon processors, and these are little endian.

Endianness

Endianness has to do with the byte order. Interestingly the term has been taken from Jonathan Swift's "Gullivers Travels" according to an entertaining Wikipedia article.

A system using Big Endian Store the most significant byte in the smallest address. In a little endian system the least significant byte is stored in the smallest address. Files created on big endianness platforms cannot be read on little endian platforms, the contents are effectively "reversed". This causes problem not only exchanging files but also network problems. Network traffic is automatically converted actually.

Data Guard

Data Guard is one of the most versatile tools in Oracle. Amongst many things it can be used to migrate to a different platform. Staying on the same endianness platform is easy. If you are on Windows (not Itaninum) for example you could create a standby database on Linux, switch over and decommission the Windows server. Converting between operating systems is a little more difficult.

Two MOS notes are relevant in this context:

- 413484.1 for physical standby databases
- 108568.1 for logical standby databases

Review the notes to see if they can help you in your case.

Migration without using Data Guard

Migrating to a different platform is of course still possible without Data Guard, but involves a little more work. Many, many options exist, amongst them you find:

- RMAN based transportable data files or convert database
- DBMS_FILE_TRANSFER will perform endianness conversion on the fly
- Export/Import with or without Data Pump
- Replication using Golden Gate, Streams, other replication users

The first option will be further investigated in this paper.

Using Transportable Tablespaces for a platform migration

Transportable Tablespaces (TTS) have been an Oracle feature for quite some time. Over its existence it has been greatly enhanced, the most important enhancement in our case is the support to convert data files from big endian to a little endian platform.

A possible high level process when using transportable tablespaces to migrate from a RISC platform to Linux/Exadata is as follows:

1. Set tablespace to read-only on source
2. Export tablespace metadata from source
3. Transport files to destination
4. Create necessary metadata (users, grants)
5. Convert all data files to target platform
6. Import tablespace metadata to destination
7. Make tablespace available for reading and writing

This process unfortunately requires a substantial amount of downtime. But:

We do not use downtime.

The migration process can be shortened depending on your application by using the method described next.

Using incremental backups plus x-platform TTS

This complex sounding tongue twister is quite a revelation actually. Described in MOS note 1389592.1 it automates the migration process to a high degree. It uses RMAN incremental backups of image copies. If you used Oracle 10.2 then you will most likely have seen this process as part of the OEM suggested backup strategy.

From a high level, this is what happens with an incremental backup/restore on image copies

1. You create an RMAN level 0 backup "as copy" of your data file. It can be your whole database or a subset.

   Time passes…

2. Create an incremental backup of your datafiles.
3. Apply incremental backup on *datafile copy*, effectively rolling it forward
4. Repeat.

This process has initially be designed with "switchover to copy" in mind. The switchover to copy command updates the control file and replaces the active data files with the image copy. Following the switchover to the database copy you apply any archived redo logs. If the incremental backups are applied regularly to the image copies you can greatly reduce the time it takes before the system becomes available again. First of all you do not need to perform a restore since the data files are

already on disk. Secondly you save time since there is less redo to be applied to the data files. Imagine a situation where you take a level 0 backup on Sunday and incremental level 1 backups on Tuesday, Thursday, and Saturday. If you incur an outage on Wednesday you have to execute an elaborate and lengthy recovery scenario.

Incremental backups and TTS together

The MOS note just mentioned describes the migration process in great detail, and it is made up of multiple stages:

1. Initial setup
2. Prepare phase
3. Roll forward phase
4. Transport phase
5. Cut over

Let's discuss the process in more detail. When reading this section please be aware that the new process automates a great many manual tasks. It cannot overcome limitations of TTS. The restrictions are listed in the documentation set in the section on TTS.

Prerequisites

The procedure is currently supported for migrations to Exadata only.

On Exadata you have to create what is referred to as an "Incremental Convert Home". At the time of writing this new Oracle home needs to be an 11.2.0.2 home with Bundle Patch 12 applied. There is no need to worry about the 11.2.0.2 home and its support status. You will not need to run a database out of this Oracle home. The Incremental Convert Home requires the following patches:

- Patch 13340675 for core functionality
- Patch 1459711 if source contains IOTs

There might be others-check the MOS note for more information.

The destination database home will most likely 11.2.0.3 with the latest QDPE installed.

The source database has to be compatible set to >= 10.2, must operate in archivelog mode. RMAN preferences should not be set to create compressed backupsets and the database must not contain offline or read only data files.

Initial setup

The initial setup is to be completed on the source and destination host. The first step is to download rman_xttconvert_1.3.zip from MOS. You could store it in the source host's $HOME/xtt. In the next step you edit the configuration file, named xtt.properties. In the example the following settings were used:

```
oracle@solaris:~/xtt$ sed -e '/^#/d' -e '/^$/d' xtt.properties
tablespaces=MBH1,MBH2
platformid=20
dfcopydir=/u01/xtt/stage_source
backupformat=/u01/xtt/stage_source
stageondest=/u01/xtt/stage_dest
```

```
storageondest=/u01/oradata/little/
backupondest=/u01/reco
cnvinst_home=/u01/app/oracle/product/11.2.0.2/xtt_home
cnvinst_sid=xtt
parallel=1
```

Tablespaces MBH1 and MBH2 are about to be exported from platform 20 (Solaris). The default directory containing the image copies will be /u01/xtt/stage_source, and this is the same directory to contain the backups.

On the destination host you create the incremental backup home with the patches described. In this home you start the incremental recovery instance in nomount mode. If the destination database does not yet exist create it in the 11.2.0.3 home. You also create the necessary directories as referenced in the xtt.properties file. Finally you copy $HOME/xtt from the source to the destination host.

In the example the following 2 databases will be on the destination host:

```
[oracle@linux dbs]$ tail -n2 /etc/oratab
xtt:/u01/app/oracle/product/11.2.0.2/xtt_home:N
little:/u01/app/oracle/product/11.2.0/dbhome_1:N
```

The XTT instance is the incremental recovery instance. Its pfile is really simple, and remember that it will only ever be started in nomount mode:

```
[oracle@linux dbs]$ cat initxtt.ora
db_name=xtt
compatible=11.2.0.2.0
```

The next step is to prepare the initial backup

Prepare Phase

With the initial setup completed it is time to take the initial backup. In this step the data file copies are created in the defcopydir. Other files created are xttplan.txt and rmanconvert.cmd. Those will be needed in the next steps.

```
oracle@solaris:~/xtt$ . oraenv
ORACLE_SID = [bigendia] ?
The Oracle base remains unchanged with value /u01/app/oracle
oracle@solaris:~/xtt$ $ORACLE_HOME/perl/bin/perl xttdriver.pl -p
Prepare source for Tablespaces:
                  'MBH1','MBH2'  /u01/xtt/stage_dest
xttpreparesrc.sql for 'MBH1','MBH2' started at Wed Jun 19 08:50:14 2013
xttpreparesrc.sql for  ended at Wed Jun 19 08:50:15 2013
oracle@solaris:~/xtt$
```

With the files created you need to get them to the destination host. SCP might be a good choice to do so.

```
oracle@solaris:~/xtt$ cd /u01/xtt/stage_source/
oracle@solaris:/u01/xtt/stage_source$ ls -l
total 205432
-rw-r-----   1 oracle   oinstall 52436992 Jun 19 08:50 MBH1_5.tf
-rw-r-----   1 oracle   oinstall 52436992 Jun 19 08:50 MBH2_6.tf
oracle@solaris:/u01/xtt/stage_source$

oracle@solaris:/u01/xtt/stage_source$ scp * oracle@linux:/u01/xtt/stage_dest/
oracle@linux's password:
MBH1_5.tf            100% |*********************************| 51208 KB    00:03
```

```
MBH2_6.tf             100% |*********************************| 51208 KB    00:03
oracle@solaris:/u01/xtt/stage_source$
```

Following this operation copy the rmanconvert.cmd and xttplan.txt files to the destination, then execute the xttdriver with the –c (“convert”) option:

```
[oracle@linux xtt]$ /u01/app/oracle/product/11.2.0/dbhome_1/perl/bin/perl
xttdriver.pl -c
Converted datafiles listed in: /tmp//xttnewdatafiles.txt
[oracle@linux xtt]$

[oracle@linux xtt]$ cat /tmp//xttnewdatafiles.txt
##MBH1
5,/u01/oradata/little/MBH1_5.xtf
##MBH2
6,/u01/oradata/little/MBH2_6.xtf
[oracle@linux xtt]$
```

This concludes the prepare phase. The rmanconvert.cmd’s contents is shown here for completeness:

```
oracle@solaris:/tmp$ cat rmanconvert.cmd
host 'echo ts##MBH1';
  convert from platform 'Solaris Operating System (x86-64)'
  datafile
   '/u01/xtt/stage_dest/MBH1_5.tf'
  format '/u01/oradata/little//%N_%f.xtf'
 parallelism 1;
host 'echo ts##MBH2';
  convert from platform 'Solaris Operating System (x86-64)'
  datafile
   '/u01/xtt/stage_dest/MBH2_6.tf'
  format '/u01/oradata/little//%N_%f.xtf'
 parallelism 1;
oracle@solaris:/tmp$
```

Nothing really special.

The following actions were performed during the prepare phase:

- Initial image copy of tablespaces created
- Data file copies are available on destination
- Files have been converted to destination platform

Notice that the source system was ONLINE all the time. Now keep calm and carry on with the next crisis.

Roll Forward Phase

The roll forward phase requires you to create incremental backups on the source, which will be transferred and converted.

```
Create the backups using the following command:
oracle@solaris:~/xtt$ $ORACLE_HOME/perl/bin/perl xttdriver.pl -i
Prepare newscn for Tablespaces: 'MBH1','MBH2'
…
Recovery Manager complete.
```

The backup files have to be copied to the destination host next. The file incrbackups.txt contains the list of files to be copied.

In the next step the incremental backups are applied and rolled forward. You need to copy xttplan.txt tsbkupmap.txt from the source system to the destination host followed by a call to xttdriver -r. Your environment variables need to point to the xtt database instance.

```
[oracle@linux xtt]$ $ORACLE_HOME/perl/bin/perl xttdriver.pl -r
Start rollforward
Started database in nomount state
CONVERTED BACKUP PIECE/u01/reco/xtts_incr_backup

PL/SQL procedure successfully completed.

Entering RollForward
After applySetDataFile
Done: applyDataFileTo
Done: RestoreSetPiece
Done: RestoreBackupPiece

PL/SQL procedure successfully completed.
[output repeated for each]
```

You may subsequently create further incremental backups on the source. Each time you do so you have to copy the xttplan.txt and tsbkupmap.txt file to the destination host.

Finally as the last step in the procedure you need to determine the new SCN. Connect to the source database and run xttdriver -s:

```
oracle@solaris:~/xtt$ $ORACLE_HOME/perl/bin/perl xttdriver.pl -s
Prepare newscn for Tablespaces: 'MBH1','MBH2'
New /tmp//xttplan.txt with FROM SCN's generated
oracle@solaris:~/xtt$ cat /tmp/xttplan.txt
 MBH1####992746
 5
 MBH2####992764
 6
```

Repeating the roll forward phase

The roll forward phase can be repeated as many times as needed. The process is always the same, but you need to pay attention to creating the new from SCN (xttdriver -s) command.

Transport phase and cut over

After a number of iterations of the previous phase the target database on Exadata should be reasonably close to the maximum SCN of the source database. At the Big Weekend you can then cut over to the new environment.

All steps in the process are known: create an incremental backup, and apply it on the target. Now you encounter an outage for the migrated tablespaces: just with any other TTS procedure you have to set the tablespace read only, export the metadata and plug the tablespaces into the target database.

Downtime begins!

The saving in downtime lies in the fact you don't have to convert the data files at this stage: they are already converted and ready.

The first step is to set the tablespaces in the source database read-only. Next you need to create a final incremental backup as demonstrated in the previous section. Use the xttdriver.pl script with option "-i" to create the backup. Do not forget to copy xttplan.txt tsbkupmap.txt as well as the new data files!

Next apply the backup on the destination using xttdriver.pl -r as shown before.

Your data files are now current and ready to be plugged in. Normally you would have to go through the somewhat painful process of creating the parameter file for expdp, but thankfully this is taken care off to:

```
[oracle@linux xtt]$ $ORACLE_HOME/perl/bin/perl xttdriver.pl -e
[oracle@linux xtt]$ cat /tmp/xttplugin.txt
impdp directory=<DATA_PUMP_DIR> logfile=<tts_imp.log> \
network_link=<ttslink> transport_full_check=no \
transport_tablespaces=MBH1,MBH2 \
transport_datafiles='/u01/oradata/little/MBH1_5.xtf', \
'/u01/oradata/little/MBH2_6.xtf';
[oracle@linux xtt]$
```

Adapt the template to match your needs and prepare the database's db link to the source database. After the impdp command has finished you are ready to go. The final task is to set the migrated tablespaces to read-write mode.