λ Efficient, hot & automatic oracle database cloning

"Discover how to clone for your production database without disruption, in a totally automated & efficient way".

By Josep Vidal (josep.vidal@uv.es)

A database clone is a complete and separate copy of a database system that includes the business data, applications and the DBMS software. Here I will show you a script that automates the whole process without disrupting production systems in an efficient and reliable manner. No special hardware nor software is needed.

This procedure is especially useful for the DBA or system administrators who wants to give his developers a full-sized TEST and DEV instance by cloning the PROD instance into the development server system. The whole procedure can be scheduled as a nightly batch crontab job and doesn't need any human intervention. In fact, we use it, in a daily basis in our organization to clone/update test instances from production systems in a wide range of platforms (Linux/x86, Solaris/SPARC, AIX/Power). The script also can run in Windows by installing CYGWIN Linux-like environment for Windows .

In order to reduce the amount of resource needed to clone/update the database, only parts (blocks) of database physical objects (datafiles, redo & archived logs) that change from source to target are updated using rsync open source tool. To reduce the amount of time needed, parallel programing techniques are applied in the script. **The whole process is triggered by target system, which queries source database catalog in order to issue a remote hotbackup and automatically recover it on target system**.

?Background / Overview

The full script can be downloaded from http://www.uv.es/vijo/pclone.sh. In

order to execute it, place it on your target (DEV) system and execute it using an oracle user account by passing the remote database you want to clone and a user/password with grants to access to catalog tables: pclone.sh ORACLE_SID user/password.

The main steps are automatized using bash scripting and are executed/triggered from the target system. They consists off:

1 Querying source database catalog to determine physical database objects forming part of the source database.

2 Move each database object from source to target systems while maintaining database consistency.

3 Automatic recovery of the source database on the target system.

?Step 0: Requirements

Before you get started some system tuning is needed in order to easy the database cloning procedure. Both source and target systems accomplish with some system requirements.

- λ Both systems have the same operating system, oracle version and system architecture.
- λ Both systems have ssh & rsync tools installed and configured so you can login without password.
- λ To have both system date & time synchronized . One easy way to achieve multiple server time synchronization is using NTP (Network Time protocol).
- λ Source database is running in archive log mode and can be reached from target system with tnsping utility.

Most systems will accomplish requirements without additional system tuning. For example to have both date and time synchronized is a good system administration practice, that is used in nowadays systems. Similarly, rsync is installed in major part of

nowadays UNIX-like operating systems). Also, it is a good practice to use the same system architecture, operating system and oracle version for both production and dev & test systems.

- **Step 1: Determine physical database objects forming part of the source database**

First step involves connecting to the source database with the username/password supplied, and query the catalog tables to make a complete list of all physical database objects forming part of the source database: tablespaces, datafiles, redo logs, archive logs, init.ora, etc ... These will be the objects that will need to be automatically moved/updated from the source to the target system.

Information provided by tnsping is used to automatically determine both the source database HOST and the instance (SERVICE_NAME). In order to automatically determine ORACLE_HOME, the database to be cloned should be cataloged at oratab file.

Before physical objects can be copied, an underlying directory structure must be created. Querying source database catalog, the underlying database filesystem directory structure (mainly datafiles, dump, redo & archived log directories) is collected on source system and replicated to the target system. In order to do so, for each physical object file belonging to the source directory, the base directory is created with the following command: `mkdir -p `dirname $f``

?**Step 2: Moving physical database objects between systems**

The whole objective of this step is to move each type of database object from the source to the target system maintaining database consistency. Among heaviest database to be moved are datafiles, redo logs and archived logs. In order to move it we use an open source tool called rsync which is capable off moving files between remote computers by coping only file differences. The idea is to issue a remote hotbackup from the target system. This means, physical database objects from the source database are

copied while maintaining database consistency to the target system, in the same location they reside in the source system.

Next, the main code necessary to accomplish the remote hotbackup its listed:

Code Listing 2. Moving database physical objects.

```
remote_backup(){
              STATUS=`target_db_status`
              if [ "$STATUS" == "OK" ]; then
              shutdown_db "IMMEDIATE"
       fi
         sync_dump_dirs
                    sync_initora
                    sync_temporary_datafiles
                    sync_db_datafiles
                    sync_db_ctrl_and_log_files
}
```

Let me comment very briefly the code. First of all, the status of the target database is checked. If the database its running, it means that you are trying to update it. So the first step, is to shut it down, to avoid copying database objects while the target database is running. Then, each type of database file is synchronized between source and target system. Source database catalog is queried to see the file location and files belonging to database physical objects are moved to the same location at the target system.

Efficient data movement with rsync

Database files movement is carried out efficiently by rsync synchronization tool. Rsync does a block level comparison of 2 files and transfers only the parts that have changed which is a huge benefit if you are transferring large files like dafafiles over

a network link. As rsync man pages states ” rsync is a program that behaves in much the same way that rcp does, but has many more options and uses the rsync remote-update protocol to greatly speed up file transfers when the destination file already exists. The rsync remote-update protocol allows rsync to **transfer just the differences between two sets of files** across the network link, using an efficient checksum-search “.

In this step, rsync tool for remote transfer uses SSH for its communications. Configuring SSH for public key authentication allows for passphrase-free logins. Password free logins benefit remote access and automation. This is very useful for moving physical objects from source to target system without being asked to enter a password each time.

Cloning database datafiles

Among different files composing source database that need to be copied are dump files, init.ora, datafiles, control and redo and archived log files. Some files (dump, archives, init.ora) can be copied without taking care of database consistency. Others like datafiles should be copied in a consistent state. In order to copy datafiles in a consistent state, source database tablespace should be put in backup mode:

```
alter tablespace $TBS begin backup
```

The algorithm can be summarized as; for each Tablespace in the source database do: First put the tablespace in backup mode, then move each datafile from source to the target system in parallel, and finally end the tablespace backup mode. The code necessary to implement it, is:

Code Listing 3. Moving database datafiles.

```
sync_db_datafiles(){
        TBS=`get_tablespaces_name  ${SOURCE_DB_CONNECT_STRING}`
                switch_db_logfile
                _ini # Parallelism related variables & locks initialization
                for T in $TBS; do
```

```
        DATAFILES=`get_tbs_datafiles ${SOURCE_DB_CONNECT_STRING} ${T}`
    begin_tbs_backup ${SOURCE_DB_CONNECT_STRING} ${T}
                    for d in $DATAFILES ; do
                _maximum_parallelism_barrier
                                mkdir -p `dirname $d`
                    rsync -e 'ssh -c blowfish' -tapogL $HOST:$d $d;_sub;exit;)&
            done
          _wait_for_all_children_to_finish_barrier
          end_tbs_backup ${SOURCE_DB_CONNECT_STRING} ${T}
                done
}
```

Parallel programing

As you can imagine, moving large databases could take a lot of time. In order to improve database cloning time, parallel programing techniques are applied. The main parallel technique is to divide the amount of objects needed to move among different tasks. In order to do so, we create a process for each datafile to be moved until a maximum level of parallelism is reached (`_maximum_parallelism_barrier`). Once the maximum parallelism degree is reached, only a new process is created when a running task finishes. Finally, the program flow must wait for ending the backup state for a determinate tablespace until all tasks created for moving datafiles finishes its work (`_wait_for_all_children_to_finish_barrier`). The maximum parallelism degree, this is the maximum number of tasks created to move physical database objects, it is a configurable parameter. A similar approach is used to move both control and redo logs files.

?Step 3: Recovering database

After a hotbackup whatever it is remote or local, always media recovery – also called datafile media recovery - is needed. The goal of datafile media recovery is to restore **database integrity**. So, what it is needed to do at this step is to recover source database physical copy on target system. In order to do so, source database catalog is queried to determine source database datafiles. For each source database datafile a media recovery is issued on the physical copy at target system. The procedure can be automated using the following code and consists in 3 steps: First the database is mounted, then media recovery is issued on each datafile and finally the database is opened:

Code Listing 3. Database recovery code:

```
recover_db(){
                export ORACLE_SID=$SOURCE_DB
        startup_db "mount"
                TBS=`get_tablespaces_name  ${SOURCE_DB_CONNECT_STRING}`
                        for T in $TBS; do
                         DATAFILES=`get_tbs_datafiles ${SOURCE_DB_CONNECT_STRING}
        ${T}`
                                for d in $DATAFILES ; do
                                recover_datafile $d "AUTOMATIC"
                        done
                         done
                 open_db
}
        recover_datafile() {
        DATAFILE=$1
        MODE=$2
        sqlplus /nolog <<EOF
```

```
connect / as sysdba ;

recover $MODE datafile '$DATAFILE' ;

exit

EOF

}
```

?Conclusion

Database cloning can be used for different purposes like testing (functional & load), developing, database maintenance tests or DBA training. Here, I have provided you with a tool that automates the whole process without disrupting production systems in an easy, efficient and reliable manner, without needing any special hardware or software. The whole process and the major steps and code necessary to implement it, has been discussed in the article.

Of course the amount of time needed to complete the whole process depends on database size, available computational resources (CPUs & discs) and database modification rate as well as the updating frequency. For example, our organization has 150 GB accounting production database running in a p-570 server uncapped LPAR (6 power 5 CPUs). After an initial cloning to test system - remote development p570 LPAR - databases were running independently during 6 months. After that, the test database was updated from production, and the whole process ended in less than an hour (58 minutes).

The whole procedure can be scheduled as a nightly batch crontab job and doesn't need any human intervention. I am currently working in a similar automation tool to clone databases between systems with different oracle versions, operating systems & system architectures.

- Josep Vidal (josep.vidal@uv.es) is a system analyst & DBA at UV, (http://www.uv.es) a Spanish university located at Valencia City. Josep has been an active open source developer on RT-Linux kernel extensions, including Posix Signals, Timers & Application defined scheduling.

References:

- SSH login without password: http://linuxproblem.org/art_9.html
- Basic NTP configuration: http://tldp.org/LDP/sag/html/basic-ntp-config.html
- Oracle hotbackup: http://www.adp-gmbh.ch/ora/admin/backup_recovery/hot_backup.html