Erreur d'I/O sur un datafile: crash d'instance ou datafile offline?

Franck Pachot, dbi-services

L'arrivée du multitenant rend encore plus critique la disponibilité de l'instance, puisqu'elle peut maintenant faire tourner plusieurs bases. Que pensez-vous qu'il se passe lorsqu'il y a une erreur d'écriture sur un datafile ? Il y a eu un changement majeur en 11.2.0.2 qui est un peu passé inaperçu car on ne supprime pas souvent un datafile. Sauf bien sûr lorsque nous installons une nouvelle infrastructure Oracle chez un client, car nous testons toujours les scénarios de backup/recovery avant de la mettre en production. Voyons donc ce qu'il se passe en supprimant un datafile en RAC et surtout en 12c multitenant…

Pour rappel, nous avions l'habitude de rencontrer les scenarios suivants:

- Perte d'un membre du controlfile = crash d'instance.
- Perte d'un membre de redo log = message dans alert.log. S'il n'y a pas d'autre membre, on devra redémarrer l'instance (avec perte de transactions).
- Perte d'un datafile système =crash d'instance
- Perte d'un datafile non-système = le datafile passe offline

Le dernier cas n'est plus vrai depuis la 11.2.0.2, voici un exemple sur un 12c:

```
RMAN> host "rm -f /u02/oradata/CDB/PDB1/PDB1_users01.dbf";
host command complete


RMAN> alter system checkpoint;
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-00601: fatal error in recovery manager
RMAN-03004: fatal error during execution of command
ORA-01092: ORACLE instance terminated. Disconnection forced
RMAN-03002: failure of sql statement command at 02/19/2015 22:51:55
ORA-03113: end-of-file on communication channel
Process ID: 19135
Session ID: 357 Serial number: 41977
ORACLE error from target database:
ORA-03114: not connected to ORACLE
```

J'ai lançé le 'alter system' à partir de RMAN car en 12c on peut exécuter toute requête SQL à partir de RMAN. Mais l'important ici est que mon instance est stoppée. Tous les services (de toutes les PDB) sont arrêtés, même si personne n'a besoin du tablespace USERS de la PDB1.

Avant, depuis les plus vieilles versions d'Oracle, ce scenario aurait simplement mis le tablespace USERS offline mais la plupart des utilisateurs pouvaient continuer à travailler.

Pourquoi mettre le datafile offline?

On est tellement habitués au comportement des anciennes versions que l'on trouve souvent normal de mettre le datafile offline s'il n'y a pas besoin de stopper l'instance. Mais pourquoi continuer à laisser l'instance tourner, en mettant seulement le datafile offline, lorsqu'on ne peut pas écrire les 'dirty buffers' lors du checkpoint ? Pourquoi ne pas stopper l'instance directement, comme ce qu'il se passe lorsque c'est un datafile système qui manque ? Et pourquoi ce comportement seulement lorsqu'on est en archivelog mode ?

La raison, c'est que pour un tablespace non-système, et lorsqu'on est en archivelog mode, et si le problème vient d'une panne disque, alors vous pouvez faire un 'restore datafile' , 'recover datafile' et 'alter datafile online' sans avoir à stopper/redémarrer l'instance.

S'il s'agit d'un tablespace système, la base ne peut pas rester ouverte. Si vous n'êtes pas en archivelog mode, le datafile ne peut pas être récupéré. Et si tout votre storage n'est plus disponible, alors de toute façon plus personne ene peut travailler – autant arrêter l'instance.

Mais si vous avez consolidé plusieurs applications sur une base de données, les applications qui n'utilisent pas le tablespace perdu peuvent continuer à être utilisés. Donc ce choix de mettre le datafile offline au lieu de stopper l'instance est guidé par le choix de laisser disponible une partie de l'application ouverte.

Ce choix était valable pour beaucoup d'installations par le passé, mais aujourd'hui :

- Quelle est la probabilité de perdre un seul datafile plutôt que tout le storage? Aujourd'hui tout est strippé. Et en ASM, on ne touche pas aux datafiles donc pas de risque d'en supprimer un par erreur.
- Avez-vous encore beaucoup d'utilisateurs qui peuvent continuer à travailler lorsqu'un datafile est offline ? On a souvent une seule application par base. Nous verrons le cas du multitenant plus loin.
- N'avez-vous pas des mécanismes de failover qui permettent de rendre la totalité du service en quelques minutes – donc beaucoup moins que le temps de récupération d'un fichier ? Si l'instance ne voit plus le fichier mais qu'il est toujours là, RAC maintient l'accès sur une autre instance. Si le fichier est perdu, DataGuard (ou Dbvisit standby lorsqu'on est en Standard Edition)

permet d'ouvrir une copie de la base rapidement, et de manière automatique si elle est en fast-start failover avec un observeur.

Plutôt que de mettre le datafile offline – et donc de rendre l'application non disponible même sur tous les nœuds d'un RAC – en attendant une opération manuelle de restore, ne vaut-il pas mieux stopper l'instance et rendre la base disponible ailleurs si un autre serveur peut voir le fichier ?

11gR2 – Patchset 2

C'est donc l'idée du 'bug 7691270', plutôt considéré comme 'enhancement request', apparu en 11.2.0.2 et mettant fin à ce comportement particulier qu'on connaissait depuis longtemps: toute erreur d'écriture au checkpoint provoque maintenant l'arrêt immédiat de l'instance. Et c'est bien sûr toujours le cas en 12c.

Evidemment, c'est un 'shutdown abort' qui est fait puisqu'on ne peut plus écrire les 'dirty buffers'. Mais les modifications sont protégées par le redo. C'est aussi la raison pour laquelle le comportement précédent – datafile offline – n'était implémenté qu'en archivelog mode, afin d'être sûr de pouvoir faire le recovery du fichier une fois récupéré. En noarchivelog mode, le redo serait prochainement écrasé lors d'un futur log switch, donc le crash de l'instance est indispensable pour éviter de générer plus de redo. On en reparle plus bas à propos des pluggable databases.

A noter que ceci ne concerne que les écritures de 'dirty buffers', lors d'un checkpoint. Les erreurs de lectures, ou d'écriture en direct-path, sont écrites dans l'alert.log mais ne changent rien au status du datafile ni de l'instance.

_datafile_write_errors_crash_instance

Ce nouveau comportement est activé par le paramètre :

```
_datafile_write_errors_crash_instance = true
```

True est la valeur par défaut. On peut revenir au mode précédent avec:

```
alter system set "_datafile_write_errors_crash_instance"=false
```

Le nouveau comportement correspond d'avantage à ce qu'on souhaite aujourd'hui, mais si vous avez consolidé plusieurs applications dans une base, que vous n'avez pas de mécanisme automatique de failover, et que vous estimez qu'il est possible de perdre un datafile sans les perdre tous, alors il peut être souhaitable de mettre ce paramètre à false.

Deux rappels :

- C'est un paramètre non-documenté, c'est une bonne idée de demander au support oracle s'il n'y a pas de contre-indication dans votre contexte.
- Si vous n'êtes pas en ARCHIVELOG mode, alors peu importe, puisque de toute façon vous ne tenez pas à vos données.

RAC

Ce changement a principalement été motivé par le fait que de plus en plus de bases sont en haute disponibilité. Mettre un datafile offline en RAC le rend inaccessible à toutes les instances. Or souvent, l'erreur d'écriture ne vient pas de l'absence du fichier, mais d'un problème d'accès à partir d'un nœud. Il est donc préférable de laisser les autres instances tourner.

Si en RAC on préfère mettre le datafile offline, avec "_datafile_write_errors_crash_instance" à false, alors il faut s'assurer d'être en 11.2.0.4 ou 12c car sinon, à cause du bug 13745317, l'agent du cluster stoppe les instances lorsqu'un datafile est offline. C'est donc pire : plus aucune instance n'est disponible dans le cluster.

Multitenant 12c

Nous avons vu plus haut qu'une raison de mettre le datafile offline était la consolidation de plusieurs applications dans une même base, ce qui n'est plus très courant aujourd'hui, d'où la décision d'arrêter en 11.2.0.2.

Mais la 12c a amené une nouvelle manière de consolider avec le multitenant. Si l'on fait du multitenant pour consolider plusieurs applications sur une même instance, veut-on toujours stopper toute l'instance (donc toute la CDB) à la première erreur d'écriture sur un datafile user d'une pluggable database (PDB) ?

Si la Haute Disponibilité est assurée par DataGuard en fast-start failover, alors oui, il est peut-être préférable de basculer tous les utilisateurs vers le site qui n'a pas de problèmes d'I/O. Si on est en RAC par contre, il est possible de ne basculer que les services associés à cette pluggable database. Tout dépend aussi si l'on doit assurer le niveau de service (SLA) par PDB ou pour toute la CDB.

Donc en multitenant, il est possible qu'on préfère stopper seulement la PDB.
Pour cela, il faudra accepter de mettre les datafiles offline :

```
alter system set "_datafile_write_errors_crash_instance"=false
```

Mais ce n'est pas suffisant. Cela ne concerne que les datafiles non-système. On ne veut pas non plus stopper toute la CDB si on perd le tablespace système d'une seule PDB. J'avais ouvert un bug là-dessus (19001390), qui est corrigé dans le dernier PSU.

Pluggable database

Bien, avec la correction de ce bug, l'instance CDB ne s'arrête pas. Mais ce n'est pas encore suffisant… si on a perdu un datafile système, il va falloir le restaurer. Et lorsqu'on perd un datafile système, on fait un shutdown abort puisqu'on ne peut pas mettre un datafile système offline. En non-CDB, pas de problème. L'instance étant arrêtée, on ne risque pas de perdre les online redo logs.

Mais faire un 'shutdown abort' d'une PDB est différent. Les online redo logs couvrent toute la CDB et vont être écrasés aux prochains log switches. Par défaut Oracle nous empêche de faire ça. Il faudra donc attendre de pouvoir stopper toute la CDB afin de faire le restore/recover du fichier manquant.

C'est un gros problème de disponibilité. Si on est en archivelog mode, on peut accepter de faire ce 'shutdown abort' de PDB puisqu'on garde le redo. On peut autoriser ça avec :

```
alter system set "_enable_pdb_close_abort"=true
```

Si l'on n'est pas en archivelog mode, alors c'est plus risqué. Donc même avec ce paramètre à true, la PDB ne sera pas mise offline. On peut malgré tout autoriser cela avec un paramètre équivalent – mais pour mode noarchivelog:

```
alter system set "_enable_pdb_close_noarchivelog"=true
```

Bien sûr, on risque alors de perdre la PDB dans ce cas. C'est acceptable seulement si on peut la recréer.

Ces deux paramètres peuvent se définir au niveau de chaque PDB, mais il faut les définir au niveau CDB si on ne veut pas que l'instance soit arrêtée au moment du checkpoint. C'est donc le comportement de toutes les PDB qu'il faut définir.

Exemple

Pour résumer, voici le comportement après avoir supprimé le datafile système d'une PDB lorsque la CDB (avec dernier PSU) a les paramètres suivants :

```
"_datafile_write_errors_crash_instance"=false
"_enable_pdb_close_abort"=true
```

Je vais utiliser le Recovery Advisor pour voir l'erreur :

```
RMAN> list failure;

Database Role: PRIMARY

List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
897        CRITICAL OPEN      23-MAR-15     System datafile 8:
 '/u02/oradata/CDB/PDB/datafile/o1_mf_system_bk04nchr_.dbf' is
 missing
```

Ce fichier est le datafile du tablespace système de la pluggable database PDB.

Grâce au paramétrage précédent, l'instance n'a pas été stoppée au checkpoint.

```
RMAN> advise failure;

Database Role: PRIMARY

List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
897        CRITICAL OPEN      23-MAR-15     System datafile 8:
 '/u02/oradata/CDB/PDB/datafile/o1_mf_system_bk04nchr_.dbf' is
 missing

analyzing automatic repair options; this may take some time
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=50 device type=DISK
analyzing automatic repair options complete


Mandatory Manual Actions
========================
no manual actions available


Optional Manual Actions
=======================
1. If file
 /u02/oradata/CDB/PDB/datafile/o1_mf_system_bk04nchr_.dbf was
 unintentionally renamed or moved, restore it
2. Automatic repairs may be available if you shutdown the
 database and restart it in mount mode


Automated Repair Options
========================
Option Repair Description
------ ------------------
1      Restore and recover datafile 8
  Strategy: The repair includes complete media recovery with
 no data loss
  Repair script:
 /u01/app/oracle/diag/rdbms/cdb/CDB/hm/reco_2804473397.hm
```

Le Recovery Advisor propose le restore/recover:

```
RMAN> repair failure;

Strategy: The repair includes complete media recovery with no
 data loss
Repair script:
 /u01/app/oracle/diag/rdbms/cdb/CDB/hm/reco_2804473397.hm

contents of repair script:
   # restore and recover datafile
   sql 'PDB' 'alter database datafile 8 offline';
   restore ( datafile 8 );
   recover datafile 8;
   sql 'PDB' 'alter database datafile 8 online';

Do you really want to execute the above repair (enter YES or
 NO)? YES
executing repair script

sql statement: alter database datafile 8 offline
```

Mais:

```
RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of repair command at 03/23/2015 14:22:34
RMAN-03015: error occurred in stored script Repair Script
RMAN-03009: failure of sql command on default channel at 03/23/2015
 14:22:34
RMAN-11003: failure during parse/execution of SQL statement: alter database
 datafile 8 offline
ORA-01541: system tablespace cannot be brought offline; shut down if
 necessary
```

Mais malheureusement – encore un bug – la pluggable database n'est pas fermée et le Recovery Advisor a oublié cette étape. Il faut donc le faire à la main :

```
RMAN> alter pluggable database PDB close;

Statement processed
```

Ceci est possible grâce à `"_enable_pdb_close_abort"=true` (je suis en mode archivelog).

Le recovery peut maintenant se faire :

```
RMAN> repair failure;
…
media recovery complete, elapsed time: 00:00:01
Finished recover at 23-MAR-15

sql statement: alter database datafile 8 online
repair failure complete
```

Il nous reste à ouvrir la PDB.

Comme il y a eu un shutdown abort, il faut aussi faire un recover sur tous les fichiers :

```
RMAN> alter pluggable database PDB open;

RMAN-00571: ===========================================================
RMAN-00569: =============== ERROR MESSAGE STACK FOLLOWS ===============
RMAN-00571: ===========================================================
RMAN-03002: failure of sql statement command at 03/23/2015 14:40:27
ORA-01113: file 9 needs media recovery
ORA-01110: data file 9:
 '/u02/oradata/CDB/datafile/o1_mf_sysaux_bjdxygdj_.dbf'

RMAN> recover pluggable database PDB;

Starting recover at 23-MAR-15
using channel ORA_DISK_1

starting media recovery
media recovery complete, elapsed time: 00:00:00

Finished recover at 23-MAR-15

RMAN> alter pluggable database PDB open;

Statement processed
```

Ce n'est donc pas tellement automatisé, mais c'est possible sans impacter la disponibilité des autres PDB. Bien sûr, il faut avoir confiance dans le monitoring. On aura besoin des archivelogs, il est préférable de ne pas trop attendre.

Conclusion

J'ai détaillé les différents scenarios et paramètres car il faut bien comprendre les conséquences en termes de disponibilité. Et il n'y a pas de cas général : la haute disponibilité est assurée de manière différente sur chaque site (RAC, Dataguard avec ou sans observer,…), et les besoins de disponibilité du service et des données sont propres à chaque application.

Il est toujours préférable de garder les paramètres par défaut. Mais dans certains cas décrits ici, on peut préférer mettre le paramètre `"_datafile_write_errors_crash_instance"` à false. Et dans ce cas, si l'on est en multitenant archivelog, il est recommandé de passer le dernier PSU et de définir aussi `"_enable_pdb_close_abort"` à true afin de pouvoir restaurer le fichier manquant sans interrompre le service des autres PDB.

Il est par contre important dans tous les cas de s'assurer que le monitoring de l'alert.log envoie une alerte dès qu'il y a une erreur d'écriture sur un fichier. Plus on attend et plus la récupération sera longue et risquée.

Dernière remarque. Aujourd'hui, beaucoup considèrent que la disponibilité des fichiers est assurée par le stockage, avec les niveaux de RAID assurant le mirroring, les SAN synchronisés, etc. On oublie malheureusement trop souvent l'erreur humaine. La perte d'un fichier peut simplement venir d'une mauvaise manipulation, d'un bug dans un script, etc. Et dans ce cas, le responsable peut rapidement faire un

restore/recover et remettre le datafile online, sans devoir stopper toutes les applications.
Pour la même raison, par notre expérience chez les clients, nous conseillons toujours de multiplexer les controlfiles et redo log files même sont mirrorés au niveau des disques.

Contact:
Franck Pachot
franck.pachot@dbi-services.com

Franck Pachot is senior consultant at dbi services in Switzerland. He has 20 years of experience in Oracle databases, all areas from development, data modeling, performance, administration, and training. Oracle Certified Master and Oracle ACE, he tries to leverage knowledge sharing in forums, publications, presentations.