**\_bct\_bitmaps\_per\_file**

云和恩墨(北京)信息技术有限公司

技术顾问 燕鑫

http://www.enmotech.com

**文档控制：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序** | **版本号** | **更改人** | **日期** | **备注** |
| 1 | 1.0版 | 燕鑫 | 2019-04-25 | 初始版本 |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1. 参数介绍 - 4 -](#_Toc7095130)

[2. 官文依据 - 5 -](#_Toc7095131)

[3. 环境准备 - 7 -](#_Toc7095132)

[4. 实验 - 8 -](#_Toc7095133)

[4.1 0级备份 - 8 -](#_Toc7095134)

[4.2 做2次differential备份，做1次cumulative备份 - 9 -](#_Toc7095135)

[4.3 再做几次cumulative备份 - 10 -](#_Toc7095136)

[4.4 换个表空间做一次0级备份，并做1次cumulative - 11 -](#_Toc7095137)

[4.5 再对sysaux做几次cumulative - 11 -](#_Toc7095138)

[4.6 再对users做一次differential备份 - 12 -](#_Toc7095139)

[4.7 总结 - 12 -](#_Toc7095140)

[5. bct文件的大小 - 15 -](#_Toc7095141)

# 参数介绍

SQL> @yx/par

Enter value for par: \_bct\_bitmaps\_per\_file

old 6: AND x.ksppinm LIKE '%&par%'

new 6: AND x.ksppinm LIKE '%\_bct\_bitmaps\_per\_file%'

NAME VALUE DESCRIB

---------------------------- ------------- ------------------

\_bct\_bitmaps\_per\_file 8 number of bitmaps to store for each datafile

来翻译一下这个解释：

为每一个数据文件而存放的位图的的数量：

1. 首先这个参数是一个数量
2. 其次这个数量是位图的数量
3. 最后，上述两点限定针对的是每一个数据文件，是each，而不是all。

说白了，就是bct文件中，在默认值的情况下，1个datafile最多有8个位图，那2个datafile，加起来最多自然就是16个位图。

那这个位图是什么呢？我们知道bct中记录的是变更的块，那么是以哪个为基准变更的块呢？显然这就取决于增量备份是cumulative还是differential，那么说白了，我们这里可以简单的推断一个位图对应该datafile的一次backup。

但是隐藏参数的这个解释有时候是不对的，所以我们不能全信，要试了才知道。

# 官文依据

下面再来贴一个官文的解释：

这是对于bct的介绍：

If block change tracking is enabled on a primary or standby database, then RMAN uses a block change tracking file to identify changed blocks for incremental backups. By reading this small bitmap file to determine which blocks changed, RMAN avoids having to scan every block in the data file that it is backing up.

The first level 0 incremental backup scans the entire data file. Subsequent incremental backups use the block change tracking file to scan only the blocks that have been marked as changed since the last backup.

如果bct被启用了，那么RMAN会使用一个快跟踪文件来确认需要进行增量备份的变化了的block。这个bct文件是个很小的位图文件（后面说它到底有多小）。有了这个文件，RMAN在做增量备份的时候就不用再把数据文件里的每个block扫一次来确定到底哪些是跟上次相比变化了的block。

0级备份是要全扫整个数据文件的，后面的增量备份有了bct文件，就不用再这么费事儿了。

下面是说bct文件的具体工作原理：

Oracle Database automatically manages space in the change tracking file to retain block change data that covers the eight most recent backups. After the maximum of eight bitmaps is reached, the oldest bitmap is overwritten by the bitmap that tracks the current changes.

The block change tracking file maintains bitmaps that mark changes in the data files between backups. The database performs a bitmap switch before each backup.

Oracle数据库自动管理bct文件的空间（不要曲解第一句的意思，它说的是单个文件，而不是说就能维护8个备份集）。当达到8个bitmap的上限时，最老的bitmap会被覆盖（再说白点儿，超过了\_bct\_bitmaps\_per\_file，老的bitmap就会被覆盖）。

bct文件维护的bitmap是什么呢？！是这个数据文件俩次备份之间变化的block。每次备份前，oracle会像切redo一样切到新的bitmap，然后再开始备份。

这是对于bitmap的举例说明：

Consider the eight-bitmap limit when developing your incremental backup strategy. For example, if you make a level 0 database backup followed by seven differential incremental backups, then the block change tracking file now includes eight bitmaps. If you then make a cumulative level 1 incremental backup, then RMAN cannot optimize the backup, because the bitmap corresponding to the parent level 0 backup is overwritten with the bitmap that tracks the current changes.

这个例子就是说，默认是一个文件8个位图，如果我们有一个0级备份和7个差异备份，那么就把8个bitmap都占了。那如果再来一个累计增量备份，怎么办？！我们这里要注意，虽然说0级备份的bitmap还在，但是前面刚说的，备份前要switch，这一次累计备份就要先对这个0级备份的bitmap覆盖，那覆盖了不就没了吗？！对！所以这次累计备份就用不到bct文件。

我们就跟着这个延伸思考一下，那么再来一次累计备份呢，会用到bct文件么？显然不会，原因就是0级备份的bitmap没了，你后面再做多少次累计备份都用不到bct文件了。

那么要是我这时候做一次差异备份呢？显然又能用到bct文件了，因为差异备份只要求上一个备份的bitmap在就行。

官文里有一个信息不要漏掉！0级备份也有自己的bitmap！

官文的也有写错的时候，所以我们还是要实验。

所以我从小到大会提醒自己的其中一件事就是书本和先辈只是指路人，它们说的对不对要先思考再测试，如果我确实没条件证明它们错，那我才迫不得已选择信。

很简单一个道理，大家都是人，文明是人建立的，凭什么他们不会错。

不过随着不断的成长，我现在会再附带提醒自己一句：不要杠！

# 环境准备

12c中，bct哪儿开，显然是cdb开，pdb虽然怎么看怎么像个单独的数据库，但是我一定清楚，真正的database是CDB，pdb只是下面一个组件罢了。

pdb下开就是这个效果：

SQL> alter database enable block change tracking using file '/home/oracle/yx/aa.trf';

alter database enable block change tracking using file '/home/oracle/yx/aa.trf'

\*

ERROR at line 1:

ORA-65040: operation not allowed from within a pluggable database

还有就是，我能确定的是11gR2以后开这个open状态直接开就行了，官文里当然也说了的（mount或这open状态）。

SQL> alter database enable block change tracking using file '/home/oracle/yx/test\_bct.trf';

Database altered.

SQL> COL STATUS FORMAT A8

SQL> COL FILENAME FORMAT A60

SQL> SELECT STATUS, FILENAME FROM V$BLOCK\_CHANGE\_TRACKING;

STATUS FILENAME

-------- ------------------------------------------------------------

ENABLED /home/oracle/yx/test\_bct.trf

开了就能用。

那我们接着别跑题，我们就是要看看这个\_bct\_bitmaps\_per\_file参数的真实效果。

如果如官文所说，那么我们这里把它改成2更好实验。

SQL> alter system set "\_bct\_bitmaps\_per\_file"=2;

System altered.

# 实验

## 0级备份

我们先做一个0级备份。在一个pdb下备份pdb的users表空间：

[oracle@host01 ~]$ rman target sys/oracle@pdbprod1

RMAN> backup incremental level 0 tablespace users;

Starting backup at 25-APR-19

using channel ORA\_DISK\_1

channel ORA\_DISK\_1: starting incremental level 0 datafile backup set

channel ORA\_DISK\_1: specifying datafile(s) in backup set

input datafile file number=00012 name=/u01/app/oracle/oradata/PRODCDB/PDBPROD1/users01.dbf

channel ORA\_DISK\_1: starting piece 1 at 25-APR-19

channel ORA\_DISK\_1: finished piece 1 at 25-APR-19

piece handle=/home/oracle/flash/PRODCDB/622F3C480F3F3E2FE0530A0CA8C0ED75/backupset/2019\_04\_25/o1\_mf\_nnnd0\_TAG20190425T094508\_gd2494d6\_.bkp tag=TAG20190425T094508 comment=NONE

channel ORA\_DISK\_1: backup set complete, elapsed time: 00:00:01

Finished backup at 25-APR-19

后面就不会这么全部贴了。

我们来看视图：

SQL> select file#,to\_char(COMPLETION\_TIME,'hh24:mi:ss') time,USED\_CHANGE\_TRACKING,INCREMENTAL\_LEVEL,con\_id from v$backup\_datafile where COMPLETION\_TIME>trunc(sysdate);

FILE# TIME USE INCREMENTAL\_LEVEL CON\_ID

---------- -------- --- ----------------- ----------

12 09:45:08 YES 0 3

可以看到，0级备份用到了bct，那也就是说它确实占一个bitmap。

## 做2次differential备份，做1次cumulative备份

我们做俩次1级差异备份，让12号文件的bitmap超过2个：

RMAN> backup incremental level 1 tablespace users;

…

tag=TAG20190425T094751

…

RMAN> backup incremental level 1 tablespace users;

tag=TAG20190425T094910

我们来查询：

SQL> select file#,to\_char(COMPLETION\_TIME,'hh24:mi:ss') time,USED\_CHANGE\_TRACKING,INCREMENTAL\_LEVEL,con\_id from v$backup\_datafile where COMPLETION\_TIME>trunc(sysdate);

FILE# TIME USE INCREMENTAL\_LEVEL CON\_ID

---------- -------- --- ----------------- ----------

12 09:45:08 YES 0 3

12 09:47:52 YES 1 3

12 09:49:10 YES 1 3

我们看到，都用到bct了的，其实这里已经佐证了只要\_bct\_bitmaps\_per\_file大于等于2，那么差异增量备份就一定能用的上bct。

然后我们做一次累计增量备份：

RMAN> backup incremental level 1 CUMULATIVE tablespace users;

tag=TAG20190425T095134

按前面官文所说，这次增量备份是用不到bct的，其实上一次增量备份如果是cumulative的，就已经用不到bct了。我们来看看是不是：

SQL> select file#,to\_char(COMPLETION\_TIME,'hh24:mi:ss') time,USED\_CHANGE\_TRACKING,INCREMENTAL\_LEVEL,con\_id from v$backup\_datafile where COMPLETION\_TIME>trunc(sysdate);

FILE# TIME USE INCREMENTAL\_LEVEL CON\_ID

---------- -------- --- ----------------- ----------

12 09:45:08 YES 0 3

12 09:47:52 YES 1 3

12 09:49:10 YES 1 3

12 09:51:34 NO 1 3

## 再做几次cumulative备份

我们再做俩次，看看是不是，cumulative方式的增量备份再也用不到bct了（因为0级的bitmap被覆盖了），我们直接看结果：

SQL> select file#,to\_char(COMPLETION\_TIME,'hh24:mi:ss') time,USED\_CHANGE\_TRACKING,INCREMENTAL\_LEVEL,con\_id from v$backup\_datafile where COMPLETION\_TIME>trunc(sysdate);

FILE# TIME USE INCREMENTAL\_LEVEL CON\_ID

---------- -------- --- ----------------- ----------

12 09:45:08 YES 0 3

12 09:47:52 YES 1 3

12 09:49:10 YES 1 3

12 09:51:34 NO 1 3

12 09:52:45 NO 1 3

12 09:53:09 NO 1 3

6 rows selected.

确实如此。

## 换个表空间做一次0级备份，并做1次cumulative

RMAN> backup incremental level 0 tablespace sysaux;

tag=TAG20190425T095400

RMAN> backup incremental level 1 CUMULATIVE tablespace sysaux;

tag=TAG20190425T095429

这一次，因为\_bct\_bitmaps\_per\_file是2，所以0级bitmap在，那么第一次cumulative一定是可以用的到bct的：

FILE# TIME USE INCREMENTAL\_LEVEL CON\_ID

---------- -------- --- ----------------- ----------

12 09:45:08 YES 0 3

12 09:47:52 YES 1 3

12 09:49:10 YES 1 3

12 09:51:34 NO 1 3

12 09:52:45 NO 1 3

12 09:53:09 NO 1 3

10 09:54:02 YES 0 3

10 09:54:29 YES 1 3

确实如此。

## 再对sysaux做几次cumulative

接着我有做了俩次sysaux的cumulative备份，我们想下，到底能不能用到bct呢？

这完全取决于oracle会覆盖哪一条bitmap，如果覆盖上一条cumulative的bitmap，那么就能接着用到bct，如果覆盖了0级备份的bitmap，那么就用不到了。

显然Oracle不会这么蠢：

FILE# TIME USE INCREMENTAL\_LEVEL CON\_ID

---------- -------- --- ----------------- ----------

12 09:45:08 YES 0 3

12 09:47:52 YES 1 3

12 09:49:10 YES 1 3

12 09:51:34 NO 1 3

12 09:52:45 NO 1 3

12 09:53:09 NO 1 3

10 09:54:02 YES 0 3

10 09:54:29 YES 1 3

10 09:54:52 YES 1 3

10 09:55:12 YES 1 3

## 再对users做一次differential备份

这里我们想一下，如果\_bct\_bitmaps\_per\_file是一个bct文件中所有的bitmap数，那么此时bct文件中应当已经没有users的数据文件的bitmap了。那么这一次差异备份就不会用到bct。

如果如我们的理解以及官文的叙述以及参数的解释，那么本次差异备份就可以用到bct：

RMAN> backup incremental level 1 tablespace users;

tag=TAG20190425T095554

FILE# TIME USE INCREMENTAL\_LEVEL CON\_ID

---------- -------- --- ----------------- ----------

12 09:55:54 YES 1 3

## 总结

\_bct\_bitmaps\_per\_file只要大于等于2，对于单纯的differential备份或单纯的cumulative备份都没影响，但是对于混合的---就一个0级备份，一会儿differential一会儿cumulative，那么对cumulative可能会有影响（说可能，是因为如果0级备份的bitmap不被覆盖，那么也不会有影响）

我们顺带可以看下v$backup\_datafile别的列，这个视图还是挺有用的：

SQL> desc v$backup\_datafile;

Name Null? Type

----------------------------------------- -------- ----------------------------

RECID NUMBER

STAMP NUMBER

SET\_STAMP NUMBER

SET\_COUNT NUMBER

FILE# NUMBER

CREATION\_CHANGE# NUMBER

CREATION\_TIME DATE

RESETLOGS\_CHANGE# NUMBER

RESETLOGS\_TIME DATE

INCREMENTAL\_LEVEL NUMBER

INCREMENTAL\_CHANGE# NUMBER

CHECKPOINT\_CHANGE# NUMBER

CHECKPOINT\_TIME DATE

ABSOLUTE\_FUZZY\_CHANGE# NUMBER

MARKED\_CORRUPT NUMBER

MEDIA\_CORRUPT NUMBER

LOGICALLY\_CORRUPT NUMBER

DATAFILE\_BLOCKS NUMBER

BLOCKS NUMBER

BLOCK\_SIZE NUMBER

OLDEST\_OFFLINE\_RANGE NUMBER

COMPLETION\_TIME DATE

CONTROLFILE\_TYPE VARCHAR2(1)

USED\_CHANGE\_TRACKING VARCHAR2(3)

BLOCKS\_READ NUMBER

USED\_OPTIMIZATION VARCHAR2(3)

FOREIGN\_DBID NUMBER

PLUGGED\_READONLY VARCHAR2(3)

PLUGIN\_CHANGE# NUMBER

PLUGIN\_RESETLOGS\_CHANGE# NUMBER

PLUGIN\_RESETLOGS\_TIME DATE

SECTION\_SIZE NUMBER

UNDO\_OPTIMIZED VARCHAR2(3)

BLOCKS\_SKIPPED\_IN\_CELL NUMBER

CON\_ID NUMBER

BACKED\_BY\_PDB VARCHAR2(3)

SPARSE\_BACKUP VARCHAR2(3)

GUID RAW(16)

SQL> select file#,to\_char(COMPLETION\_TIME,'hh24:mi:ss') time,USED\_CHANGE\_TRACKING,INCREMENTAL\_LEVEL,BLOCKS,CHECKPOINT\_CHANGE#,BACKED\_BY\_PDB from v$backup\_datafile where COMPLETION\_TIME>trunc(sysdate);

FILE# TIME USE INCREMENTAL\_LEVEL BLOCKS CHECKPOINT\_CHANGE# BAC

---------- -------- --- ----------------- ---------- ------------------ ---

12 09:45:08 YES 0 741 12624004 YES

12 09:47:52 YES 1 11 12624446 YES

12 09:49:10 YES 1 2 12624655 YES

12 09:51:34 NO 1 11 12625086 YES

12 09:52:45 NO 1 11 12625286 YES

12 09:53:09 NO 1 11 12625357 YES

10 09:54:02 YES 0 53306 12625501 YES

10 09:54:29 YES 1 1 12625581 YES

10 09:54:52 YES 1 1 12625680 YES

10 09:55:12 YES 1 1 12625744 YES

# bct文件的大小

The size of the block change tracking file is proportional to the size of the database and the number of enabled threads of redo.

The size of the block change tracking file can increase and decrease as the database changes. The size is not related to the frequency of updates to the database. Typically, the space required for block change tracking for a single instance is approximately 1/30,000 the size of the data blocks to be tracked. For an Oracle RAC environment, it is 1/30,000 of the size of the database, times the number of enabled threads.

The following factors that may cause the file to be larger than this estimate suggests:

1. To avoid the overhead of allocating space as your database grows, the block change tracking file size starts at 10 megabytes. New space is allocated in 10 MB increments. Thus, for any database up to approximately 300 gigabytes, the file size is no smaller than 10 MB, for up to approximately 600 gigabytes the file size is no smaller than 20 megabytes, and so on.
2. For each data file, a minimum of 320 kilobytes of space is allocated in the block change tracking file, regardless of the size of the data file. Thus, if you have a large number of relatively small data files, the change tracking file is larger than for databases with a smaller number of larger data files containing the same data.

这个总结一下：

1. 大小跟数据库大小以及enabled redo threads数（也可以理解为rac的节点数）成比例
2. 跟数据库发生的更新频次无关！！！
3. 大小的估算公式是：size of database \* 1/30000 \* number of enabled threads
4. 300g以下的库bct文件不会小于10m，600g以下的库bct文件不会小于20m，依此类推
5. bct文件中，为每个datafile至少分配320k的空间。
6. 由第5条得出，就算是数据完全一样的数据库，数据文件多的database的bct文件要更大。