Ora-600[4511]

# 起因

Oracle在试图修改一行数据时，发现块上对应行的lb不为0，但是在对应的itl槽上没有发现活动的事务。

对应的文档是：ORA-600 [4511] "Row locked by inactive ITL" (Doc ID 39553.1)

准备实验表：

SQL> create table t1(id number);

Table created.

SQL> insert into t1 values(2);

1 row created.

SQL> @yx/rowid

Enter value for owner: yx

Enter value for tname: t1

OBJ# FILE# BLOCK# ROW#

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

77277 7 143 0

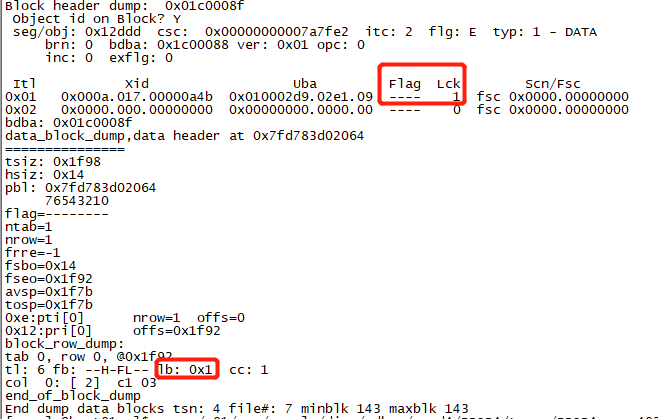
我们将7号文件的143号block dump出来（一定要记得刷块）：

SQL> alter system flush buffer\_cache;

System altered.

SQL> alter system dump datafile 7 block 143;

System altered.



正常的是这样的，lb有值，对应的itl槽要么就是活动事务，要么就是flag是U，也就是做的是fast cleanout。

然后我们提交该事务，再来dump：

SQL> commit;

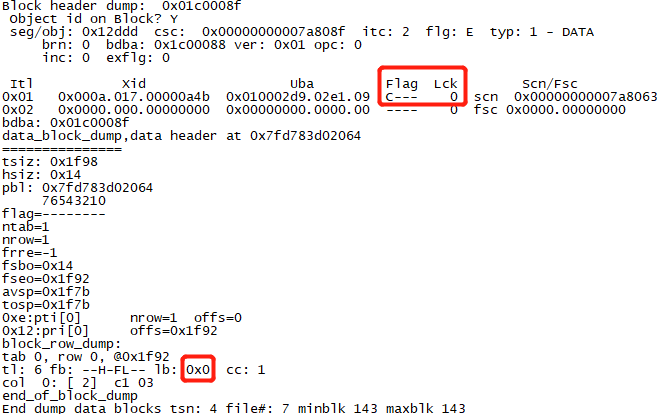
Commit complete.

SQL> alter system flush buffer\_cache;

System altered.

SQL> alter system dump datafile 7 block 143;

System altered.



我们看到，正常的效果应该是这个样子的。当然这是已经做了块延迟清除后的效果。

然后我们连续做几个事务，把事务槽重用几次：

SQL> insert into t1 values(3);

1 row created.

SQL> commit;

Commit complete.

SQL> insert into t1 values(3);

1 row created.

SQL> commit;

Commit complete.

SQL> insert into t1 values(3);

1 row created.

SQL> commit;

Commit complete.

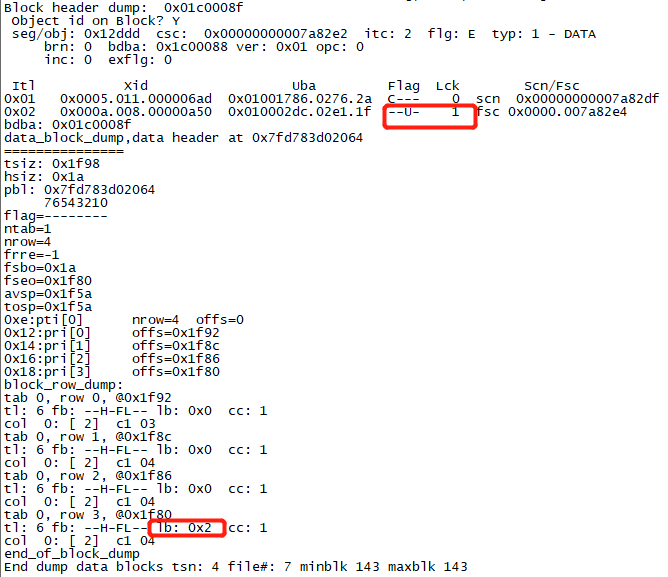
我们再来dump：

SQL> alter system flush buffer\_cache;

System altered.

SQL> alter system dump datafile 7 block 143;

System altered.



我们看到这里最后一行的事务虽然已经提交，但是依然持有锁，原因就是这里是做了fast cleanout，我们看到对应的flag中也是U，fsc没有释放，lck也有值。

但是，如果lb有值，但是flag是C并且lck为0，fsc被释放已经变成了SCN，说白了就是，从事务槽上看，这个事务槽已经被释放可以重用了，但是从行上看，这个行还在被锁着。那么就会有问题，但是我认为ORACLE发现这种情况时，应当去主动清除该行上的lock。但是事实如何呢？

# 制造问题

很简单，我们把id为2的那行数据中的lb修改为01。

## 寻找对应的offset

### 自己算

这里我们需要确定该数据的offset。我们看到dump中记录的是0x1f92，换算成10进制就是8082。那么因为该表是ASSM，且该block有2个itl，所以真实的offset应当是加100，也就是8182。

### Bbed查

BBED> set file 7 block 143

FILE# 7

BLOCK# 143

BBED> p kdbr

sb2 kdbr[0] @118 8082

sb2 kdbr[1] @120 8076

sb2 kdbr[2] @122 8070

sb2 kdbr[3] @124 8064

然后这里我们可以使用上面加100的方式算。

那么我们想要直接得到正确的位置，应当使用p \*kdbr[0]：

BBED> p \*kdbr[0]

rowdata[18]

-----------

ub1 rowdata[18] @8182 0x2c

这里我们可以看下rowdata[18]的内容：

BBED> p rowdata[18]

ub1 rowdata[18] @8182 0x2c

BBED> x /rn

rowdata[18] @8182

-----------

flag@8182: 0x2c (KDRHFL, KDRHFF, KDRHFH)

lock@8183: 0x00

cols@8184: 1

col 0[2] @8185: 2

现在我们来就行修改：

BBED> dump offset 8182 count 32

File: /u01/app/oracle/oradata/PROD4/users01.dbf (7)

Block: 143 Offsets: 8182 to 8191 Dba:0x01c0008f

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

2c000102 c1030106 e482

<32 bytes per line>

BBED> modify /x 01 offset 8183

Warning: contents of previous BIFILE will be lost. Proceed? (Y/N) y

File: /u01/app/oracle/oradata/PROD4/users01.dbf (7)

Block: 143 Offsets: 8183 to 8191 Dba:0x01c0008f

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

010102c1 030106e4 82

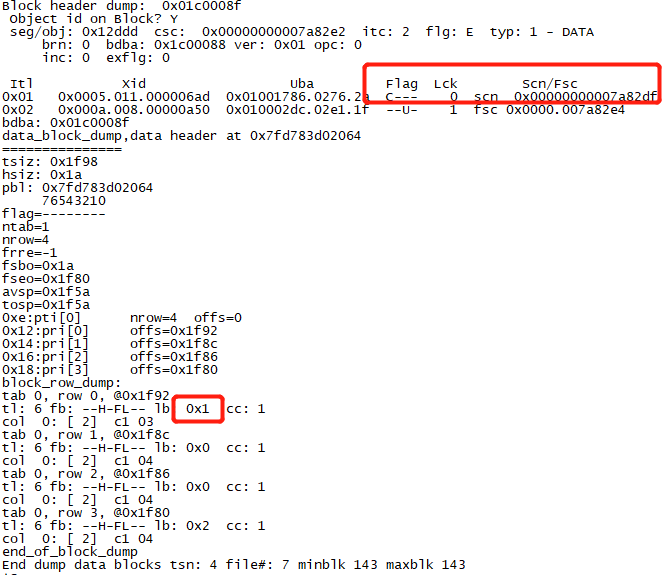
<32 bytes per line>

BBED> sum apply

Check value for File 7, Block 143:

current = 0xef5a, required = 0xef5a

现在我们再来dump看下：



我们看到我们已经把情况构造出来了。

## 对id=2做dml

我们现在select：

SQL> select \* from t1 where id=2;

ID

----------

2

正常！

然后再做dml：

SQL> delete from t1 where id=2;

delete from t1 where id=2

\*

ERROR at line 1:

ORA-00600: internal error code, arguments: [4511], [], [], [], [], [], [], [], [], [], [], []

我们看到，确实是4511的错误。

## 检测问题

我们首先按mos所说尝试analyze：

SQL> analyze table t1 VALIDATE STRUCTURE CASCADE;

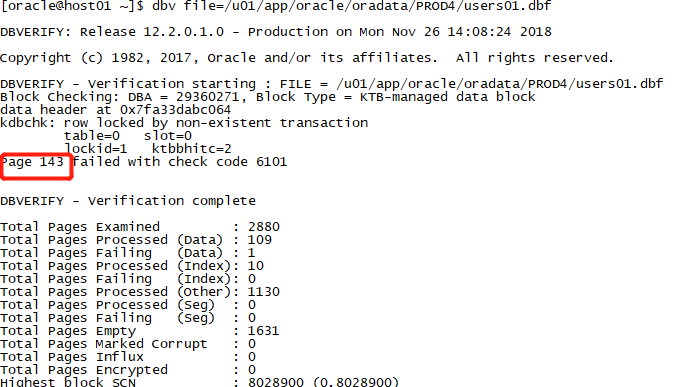
analyze table t1 VALIDATE STRUCTURE CASCADE

\*

ERROR at line 1:

ORA-01498: block check failure - see trace file

然后我们使用dbv去查：



我们看到，确实查到了。

那么rman会不会发现呢？！

这个问题最初是在人社的生产环境上发现的。但是备份一直在正常进行，并没有出问题，而且我们select依然能正常进行。所以我认为RMAN并不会发现该坏块：

RMAN> validate datafile 7;

Starting validate at 26-NOV-18

using target database control file instead of recovery catalog

allocated channel: ORA\_DISK\_1

channel ORA\_DISK\_1: SID=748 device type=DISK

channel ORA\_DISK\_1: starting validation of datafile

channel ORA\_DISK\_1: specifying datafile(s) for validation

input datafile file number=00007 name=/u01/app/oracle/oradata/PROD4/users01.dbf

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

List of Datafiles

=================

File Status Marked Corrupt Empty Blocks Blocks Examined High SCN

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

7 OK 0 1631 2889 8028900

File Name: /u01/app/oracle/oradata/PROD4/users01.dbf

Block Type Blocks Failing Blocks Processed

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

Data 0 109

Index 0 10

Other 0 1130

Finished validate at 26-NOV-18

并且可以正常备份。如果rman发现不了，那么也就是说V$DATABASE\_BLOCK\_CORRUPTION中就不会有记录：

SQL> select \* from V$DATABASE\_BLOCK\_CORRUPTION;

no rows selected

# 解决问题

## 修改block

网上有说法称重启库或者flush buffer cache可解决。理论上我并不认为可行，经过尝试，也确实不可行。

那么我们就把id=2那一行的lb再改成0：

BBED> set file 7 block 143;

FILE# 7

BLOCK# 143

BBED> set offset 8182

OFFSET 8182

BBED> dump count 32

File: /u01/app/oracle/oradata/PROD4/users01.dbf (7)

Block: 143 Offsets: 8182 to 8191 Dba:0x01c0008f

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

2c010102 c1030106 e482

<32 bytes per line>

BBED> modify /x 00 offset 8183

Warning: contents of previous BIFILE will be lost. Proceed? (Y/N) y

File: /u01/app/oracle/oradata/PROD4/users01.dbf (7)

Block: 143 Offsets: 8183 to 8191 Dba:0x01c0008f

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

000102c1 030106e4 82

<32 bytes per line>

BBED> sum apply

Check value for File 7, Block 143:

current = 0xee5a, required = 0xee5a

然后我们再来删：

SQL> delete from t1 where id=2;

delete from t1 where id=2

\*

ERROR at line 1:

ORA-00600: internal error code, arguments: [4511], [], [], [], [], [], [], [],

[], [], [], []

这是因为块是从内存中读到的。

我之前有个错误的认识，就是flush buffer cache会用buffer中的block把磁盘上的block覆盖掉，也就是说我白改了。当然这个错误的认识是之前确实遇到过这种情况，但我觉得按道理也应该是这样的。但是这次就不是。

我们来flush一遍buffer cache，然后再delete：

SQL> delete from t1 where id=2;

1 row deleted.

SQL> commit;

Commit complete.

成功删除。

## Move tablespace

我们这次先创建一个索引再创建一个存储过程：

SQL> create index idx\_t1 on t1(id);

Index created.

SQL> create or replace procedure test\_t1

2 as

3 begin

4 for i in (select id from t1) loop

5 dbms\_output.put\_line(i.id);

6 end loop;

7 end;

8 /

Procedure created.

SQL> exec test\_t1;

3

3

3

PL/SQL procedure successfully completed.

我们再次制造该错误：

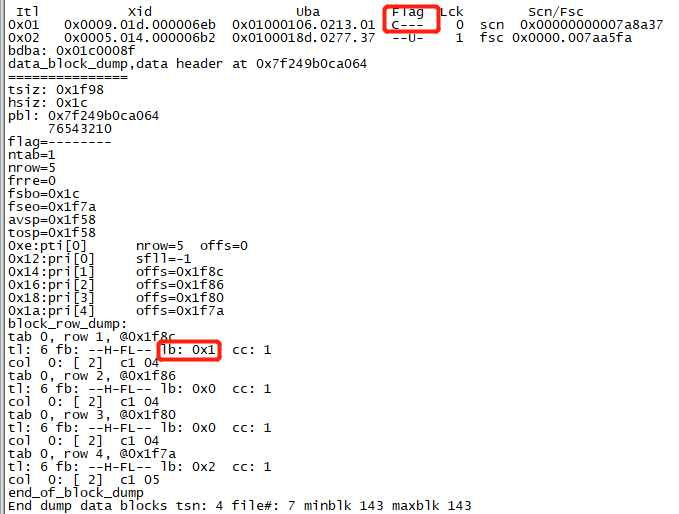
SQL> delete from t1 where id=3;

delete from t1 where id=3

\*

ERROR at line 1:

ORA-00600: internal error code, arguments: [4511], [], [], [], [], [], [], [], [], [], [], []



现在我们来move该表：

SQL> alter table t1 move tablespace users;

Table altered.

再次删除：

SQL> delete from t1 where id=3;

3 rows deleted.

SQL> rollback;

Rollback complete.

可以看到，问题已经得到了解决。但是我们一定要记得重建索引。其实alert日志中是有提示的：

2018-11-27T15:41:53.062086+08:00

Some indexes or index [sub]partitions of table YX.T1 have been marked unusable

所以我们重建索引：

SQL> select 'alter index '||owner||'.'||index\_name||' rebuild online;' rebiud\_index\_sql from dba\_indexes where table\_name='T1' and owner='YX';

REBIUD\_INDEX\_SQL

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

alter index YX.IDX\_T1 rebuild online;

SQL> alter index YX.IDX\_T1 rebuild online;

Index altered.

SQL> set serveroutput on

SQL> exec test\_t1;

3

3

3

4

存储过程也没问题。现在表已经移动到了新的block上：

SQL> @yx/rowid.sql

Enter value for owner: yx

Enter value for tname: t1

OBJ# FILE# BLOCK# ROW#

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

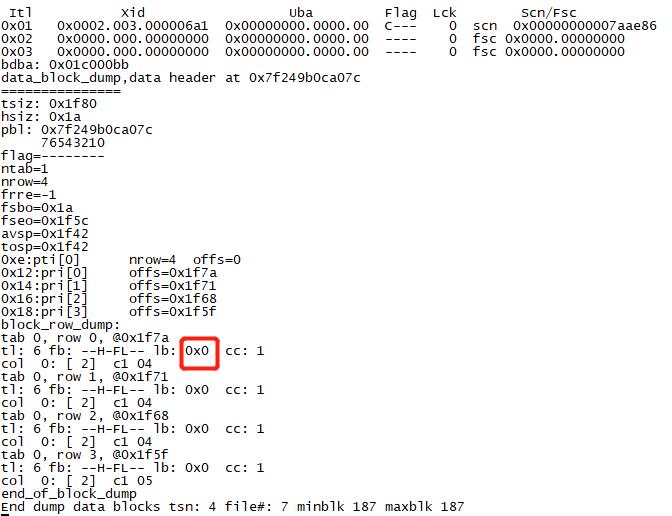
77285 7 187 0

77285 7 187 1

77285 7 187 2

77285 7 187 3

然后我们继续看一下新的block中的内容：



我们看，不知道为啥多了个itl槽，借机正好验证一下offset的公式，这次应当是加124：

SQL> select to\_number('1f7a','xxxxxx') from dual;

TO\_NUMBER('1F7A','XXXXXX')

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

8058

SQL> select 8058+124 from dual;

8058+124

----------

8182

BBED> p \*kdbr[0]

rowdata[30]

-----------

ub1 rowdata[30] @8182 0x2c

BBED> p kdbr[0]

sb2 kdbr[0] @142 8058

## 新建表rename的方法

这里在新表上重新把老表的索引创建就不说了。

我们主要说存储过程。

我们现在看存储过程的状态：

SQL> select status from user\_objects where object\_name='TEST\_T1';

STATUS

-------

VALID

然后rename一下再看：

SQL> alter table t1 rename to t2;

Table altered.

SQL> select status from user\_objects where object\_name='TEST\_T1';

STATUS

-------

INVALID

然后我们新建个t1：

SQL> create table t1(id varchar2(2));

Table created.

SQL> insert into t1 values('A');

1 row created.

SQL> commit;

Commit complete.

SQL> select status from user\_objects where object\_name='TEST\_T1';

STATUS

-------

INVALID

SQL> exec test\_t1;

A

PL/SQL procedure successfully completed.

SQL> select status from user\_objects where object\_name='TEST\_T1';

STATUS

-------

VALID

我们看到，只要这个表名存在了，这个存储过程就能继续使用。

虽然表刚新建的时候存储过程仍是invalid，但是只要调用一次，就会变为valid。

当然，我们直接执行脚本utlrp.sql也行：

SQL> select status from dba\_objects where object\_name='TEST\_T1';

STATUS

-------

INVALID

SQL> @?/rdbms/admin/utlrp

SQL> select status from dba\_objects where object\_name='TEST\_T1';

STATUS

-------

VALID

# Interblock and Intrablock Corruption

Interblock corruption是块与块之间的corruption，一定是logical corruption。

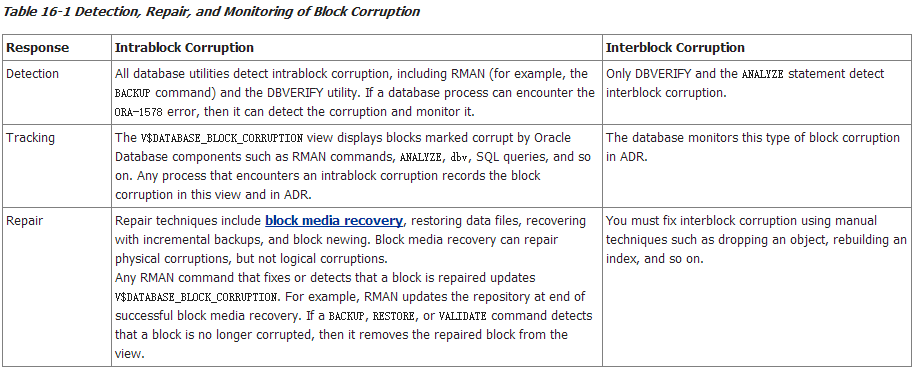
Intrablock corruption是块里面的的corruption，可以是physical和logical俩种情况。

用RMAN和DBV可以检测出Intrablock corruption，而interblock corruption只能靠dbv和analyze检测出来。

V$DATABASE\_BLOCK\_CORRUPTION会记录被标记为corrupt的intrablock corruption，这些被标记的block可能被rman、analyze、dbv或者sql查询等标记的。

对于intrablock corruption，使用介质恢复、restore等都可以进行恢复。同时，被恢复了的block，V$DATABASE\_BLOCK\_CORRUPTION中会删掉关于该block的记录。而对于interblock corruption，必须要手动恢复，比如重建索引，drop表等等。

如下是官文中的比对：



那么我们看，本次的块损坏，虽然不是块与块之间的损坏，但应当数据interblock corruption。所以只有dbv和analyze可以检测到。