

# **Locking conflicts resolution in Oracle RAC environments.**

**A Technical White Paper**

**By**

**David Gornshtein**

**Boris Tamarkin**

**[WisdomForce Technologies, Inc](http://www.wisdomforce.com)**

**<http://www.wisdomforce.com>**

**Revision 1.0**

**Note\*: The latest copy of the document is available at**  
<http://www.wisdomforce.com/dweb/resources/docs/locking.rac.pdf>

*This topic is about resolving locking conflicts in Oracle RAC environment with example and hands on.*

## Introduction.

The distributed locking is one of the most complicated issues in using Oracle RAC. Its complexity did not reduce at whole since the OPS times.

Most of the issues may be resolved by running sql statements via gv\$ and x\$ tables. However in order to resolve the most complicated problems you or oracle support should use system state dumps and oradebug.

In this short topic we will explain relatively simple case of distributed lock problem. Shortly we will present two additional topics regarding usage of "oradebug lkdebug" statement and another topic about reading results of "oradebug -g all hanganalyze 3" and "oradebug -g all dump systemstate 10" respectively.

## Basic definitions

Blocking lock defined as *distributed* if session on instance A trying to lock row and waiting for another session locking this row on instance B.

Blocking lock defined as *local* if session on instance A trying to lock row and waiting for another session locking this row on the same instance A.

## Locking resolution

In usual scenario you as a DBA will be paged after system is HALT and everything stopped to work. CPUs on all nodes are on 0% and it seems like a hang.

Hang is possible, however more likely there is a distributed locking problem when dealing with RAC.

First step is connect to some instance as a DBA and try to identify distributed locking problem. The following statement will return list of all blocking locks that may be distributed or local.

```
SELECT lh.inst_id Locking_Inst, lh.sid Locking_Sid,
       lw.inst_id Waiting_Inst, lw.sid Waiter_Sid,
       decode ( lh.type, 'MR', 'Media_recovery',
                  'RT', 'Redo_thread',
                  'UN', 'User_name',
                  'TX', 'Transaction',
                  'TM', 'Dml',
                  'UL', 'PLSQL User_lock',
                  'DX', 'Distrted Transaxion',
                  'CF', 'Control_file',
                  'IS', 'Instance_state',
                  'FS', 'File_set',
                  'IR', 'Instance_recovery',
                  'ST', 'Diskspace Transaction',
```

```

        'IV', 'Libcache_invalidation',
        'LS', 'LogStaartORswitch',
        'RW', 'Row_wait',
        'SQ', 'Sequence_no',
        'TE', 'Extend_table',
        'TT', 'Temp_table',
        'Nothing-' ) Waiter_Lock_Type,
    decode ( lw.request, 0, 'None',
              1, 'NoLock',
              2, 'Row-Share',
              3, 'Row-Exclusive',
              4, 'Share-Table',
              5, 'Share-Row-Exclusive',
              6, 'Exclusive',
              'Nothing-' ) Waiter_Mode_Req
FROM gv$lock lw, gv$lock lh
WHERE lh.id1=lw.id1
      AND lh.id2=lw.id2
      AND lh.request=0
      AND lw.lmode=0
      AND (lh.id1,lh.id2) in (
          SELECT id1,id2 FROM gv$lock WHERE request=0
          INTERSECT
          SELECT id1,id2 FROM gv$lock WHERE lmode=0 );

```

This example has been tested on a 4 node sun cluster with 4 node RAC 9.2, therefore instances are numbered from 1 to 4 respectively.

Output:

```

LOCKING_INST LOCKING_SID WAITING_INST WAITER_SID WAITER_LOCK_TYPE
WAITER_MODE_REQ
-----
      1      41          2      37   Transaction Exclusive
      4      22          1      41   Transaction Exclusive

```

Now you see that session 22 on instance 4 is locking, session 41 on instance 1 is waiting and session 37 on instance 2 waiting for something locked by session 41 on instance 1

```

4.22 --
-
-- 1.41 --
-
-- 2.37

```

Next step would be investigating what are the blocking sessions username and what object has been locked.

Look at the objects locked by sessions: STA == ACT means waiting, STA == INA means locked

```

SELECT a.inst_id, a.session_id, b.serial#, b.status, a.oracle_username,
       a.os_user_name, a.process, c.name
FROM   sys.obj$ c, gv$session b, gv$locked_object a
WHERE  a.session_id=b.sid
AND    c.obj#=a.object_id

```

Output:

```
INST_ID SID  SER# STA ORACLE_USERNAME  OS_USER_NAME  Os Proc  Locked  
Object
```

```
-----  
1 41 9123 ACT RECOVER      oracle      8279  LOCK_TEST  
1 41  59 INA RECOVER      oracle      8279  LOCK_TEST  
1 41 4316 INA RECOVER      oracle      8279  LOCK_TEST  
2 37 952 ACT RECOVER      oracle      9189  LOCK_TEST  
2 37 216 INA RECOVER      oracle      9189  LOCK_TEST  
2 37 188 INA RECOVER      oracle      9189  LOCK_TEST  
4 22  15 INA RECOVER      oracle     15149  LOCK_TEST  
4 22  32 INA RECOVER      oracle     15149  LOCK_TEST  
4 22 2703 INA RECOVER      oracle     15149  LOCK_TEST
```

This time you may kill deepest blocking session (in our example it is sid 37 on instance 2). Very likely that it will resolve the problem, but you may still want investigate what the blocked / blocking queries are before.

```
select s.inst_id, substr(u.username,1,12) user_name, s.sql_text  
from gv$sql s, gv$session u  
where s.hash_value = u.sql_hash_value  
and s.inst_id = u.inst_id  
and lower(sql_text) not like '%from v$sql s, v$session u%'  
and sid in (37, 41, 22);
```

Output:

```
INST_ID USER_NAME  SQL_TEXT  
-----  
1 RECOVER  select OBJECT_NAME from lock_test where OBJECT_NAME =  
:"SYS_B_0" for update  
2 RECOVER  select OBJECT_NAME from lock_test where OBJECT_NAME =  
:"SYS_B_0" for update
```