Tracing Deadlocks

The article here is to identify and track the source for the cause of deadlock. Typically in a system integration solution you wouldn’t encounter deadlock in your testing. This deadlock might creep in production or in live phase as your solution talks to different systems, which is connected, live with users. In a production environment turning the profiler and searching for the deadlock on is really painful and moreover the performance is also affected to a small extent.

Here are the steps to identify the source for the deadlock:

- Enable the Trace flag 3604 followed by 1204. The trace flag 3604 is not documented in SQL 2K books online. This option sends the trace to the client.
- The trace flag 1204 returns the type of locks participating in the deadlock and the current command affected. The deadlock information is automatically sent to the error log
- DBCC TRACEON (3604)
- DBCC TRACEON (1204)

Now any deadlocks happening in SQL server will be logged into the error log file, ERRORLOG. You can access the file either through enterprise manager or through explorer. In enterprise manager you can see it under Management / SQL Server Logs or under <Drive:>\\Program Files\Microsoft SQL Server\MSSQL\LOG.

The deadlock information appears in the file as

```
Deadlock encountered.... Printing deadlock information

2004-02-28 09:31:20.86 spid4

2004-02-28 09:31:20.86 spid4     Wait-for graph

2004-02-28 09:31:20.86 spid4

2004-02-28 09:31:20.86 spid4     Node:1

2004-02-28 09:31:20.86 spid4     RID: 7:1:35443:0            CleanCnt:1 Mode: U Flags: 0x2

2004-02-28 09:31:20.86 spid4      Grant List 0::

2004-02-28 09:31:20.86 spid4       Owner:0x5c53dfa0 Mode: U       Flg:0x0 Ref:0 Life:00000001 SPID:61 ECID:0

2004-02-28 09:31:20.86 spid4       SPID: 61 ECID: 0 Statement Type: UPDATE Line #: 167

2004-02-28 09:31:20.86 spid4       Input Buf: Language Event: Sproc_TPS_Reports


2004-02-28 09:31:20.86 spid4      Requested By:

2004-02-28 09:31:20.86 spid4       ResType:LockOwner Stype:'OR' Mode: U SPID:58 ECID:0 Ec:(0x6EF235F8) Value:0x658cb560
Cost:(0/297FC)

2004-02-28 09-: 31:20.86 spid4

2004-02-28 09:31:20.86 spid4     Node:2

2004-02-28 09:31:20.86 spid4     RID: 7:1:76320:0            CleanCnt:1 Mode: X Flags: 0x2

2004-02-28 09:31:20.86 spid4      Grant List 1::
```

```
2004-02-28 09:31:20.86 spid4      Owner:0xdabf560 Mode: X       Flg:0x0 Ref:0 Life:02000000 SPID:58 ECID:0
2004-02-28 09:31:20.86 spid4      SPID: 58 ECID: 0 Statement Type: DELETE Line #: 158
2004-02-28 09:31:20.86 spid4      Input Buf: Language Event: EXEC Sproc_TPS_CaptureMissingEvent
2004-02-28 09:31:20.86 spid4     Requested By:
2004-02-28 09:31:20.86 spid4      ResType:LockOwner Stype:'OR' Mode: U SPID:61 ECID:0 Ec:(0x5A507578) Value:0x8c32720 Cost:(0/0)
2004-02-28 09:31:20.86 spid4    Victim Resource Owner:
2004-02-28 09:31:20.86 spid4     ResType:LockOwner Stype:'OR' Mode: U SPID:61 ECID:0 Ec:(0x5A507578) Value:0x8c32720 Cost:(0/0)
```

Let’s analyze the log file. In the above case 2 nodes were involved in deadlock. Node :1 belongs to SPID :61 and Node :2 belongs to SPID :58. To know the source query that was under execution during deadlock, can be found from the input buffer. This line can be found from the line starting with “input Buf:” For Node:1 its “Sproc_TPS_reports” and for Node :2 its “Sproc_TPS_CaptureMissingEvent”.

Now that we know the query source we need to know what was the statement and the line no# that caused the deadlock. This can be retrieved from “Statement Type” present just above the “Input Buf” statement. In this case its “Update Line #167” for SPID:61 and “Delete Line #158” for SPID : 58.

Now that you know the line no# , you can attack the query to optimize it for avoiding deadlock. The problem lies if there are multiple calls to stored procedure within this stored procedure Sproc_TPS_Reports or Sproc_TPS_CaptureMissingEvent? Then this line no # and statement type can’t say much to pinpoint the object involved in the deadlock. The next step is to look at the line next to “Node :1” or “Node :2”. In the this example its RID: X: X: X: X. The RID indicates that the data is fetched from heap directly and it’s a row lock. The The SQL server can lock on the following items

Items	**Description**
RID	is a row identifier. It is used to individually lock a single row within a table. The format is RID: *db_id*:*file_id*:*page_no*:*row_no*
Key	is a row lock within an index. Used to protect key ranges in serializable transactions The format is KEY: *db_id*:*object_id*:*index_id*;
Page	is a lock, when entire 8-KB data page or index page will be locked. The format is PAG: *db_id*:*file_id*:*page_no*
Extent	is only used for allocation. When it's used, entire extent will be locked The format is EXT: *db_id*:*file_id*:*extent_no*
Table	lock is used when a large percentage of the table's rows are queried or updated. This lock includes all table's data and indexes The format is TAB: *db_id*:*object_id*;
Database	is used when you restore the database. The format is DB: *db_id*

From the above table its clear that except for Extent and Database Items you can trace the deadlock object with ease. Anyways extent and database will not be encountered in your custom code stored procedure. Other than the extent and database, key and table have the object_id in it. So by just executing the query "select object_name(<object_id>)". You can determine the object name easily. For other lock items RID and page you can determine the object name from the Page_no. This can be determined through an undocumented DBCC command DBCC PAGE. The syntax for this is:

DBCC PAGE ({db_id|dbname}, pagenum [,print option] [,cache] [,logical])

where:

db_id|dbname **-** Enter either the dbid or the name of the database

pagenum **-** Enter the page number of the SQL Server page that is to be examined

print option **-** (Optional) Print option can be either 0, 1, or 2

0 **-** (Default) This option causes DBCC PAGE to print out only the page header information.

1 **-** This option causes DBCC PAGE to print out the page header information, each row of information from the page, and the page's offset table. Each of the rows printed out will be separated from each other.

2 **-** This option is the same as option 1, except it prints the page rows as a single block of information rather than separating the individual rows. The offset and header will also be displayed.

cache **-** (Optional) This parameter allows either a 1 or a 0 to be entered

0 **-** This option causes DBCC PAGE to retrieve the page number from disk rather than checking to see if it is in cache.

1 **-** (Default) This option takes the page from cache if it is in cache rather than getting it from disk only.

logical **-** (Optional) This parameter is for use if the page number that is to be retrieved is a virtual page rather then a logical page. It can be either 0 or 1.

0 **-** If the page is to be a virtual page number.

1 **-** (Default) If the page is the logical page number.

So just pass the parameter that you get from RID / Page Lock items. In the above case it would be DBCC Page(7,1,35443,3)

The output for this will appear in client as below. For ease in understanding, only the portion that is of interest is copied here.

```
PAGE: (1:35443)

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

BUFFER:

-------

BUF @0x0196D440

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

bpage = 0x7BA22000        bhash = 0x00000000        bpageno = (1:35443)

bdbid = 7                 breferences = 0          bstat = 0x9

bspin = 0                 bnext = 0x00000000

PAGE HEADER:

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

Page @0x7BA22000

----------------
```

```
m_pageId = (1:35443)      m_headerVersion = 1       m_type = 1

m_typeFlagBits = 0x0      m_level = 0               m_flagBits = 0x0

m_objId = 2029457433      m_indexId = 0             m_prevPage = (1:37670)

m_nextPage = (1:106590)   pminlen = 5               m_slotCnt = 8

m_freeCnt = 7408          m_freeData = 7996         m_reservedCnt = 0

m_lsn = (77495:484:39)    m_xactReserved = 0        m_xdesId = (0:9175698)

m_ghostRecCnt = 0         m_tornBits = 0
```

of all these info , the data of m_objID is of interest(which is highlighted in bold). The value is 2029457433. Now that you know the object ID you can determine the object name by executing the query

```
SELECT  object_name(2029457433)
```

Now that you have traced the source, you can work out to find the solution for avoiding the deadlock.