http://www.sqlservercentral.com/articles/deadlock/64315/
Printed 2009/02/03 06:08AM

A Deadlock Occurence and Resolution

By Praveen Voleti, 2008/09/10

Introduction

An upgrade was performed at my company recently to the application code on the SQL Server, which involved basically the addition of columns to few tables. After the upgrade, the customer started facing severe deadlocks. Around 8 out of 10 transactions were getting deadlocked and were failing.

Steps Taken in resolving the deadlocks:

I will walk through the steps we took to determine what was causing the deadlock and resolve the issue.

1. First I enabled the trace flag 1204 on the server to capture the deadlock details in the error log with the command listed below:

```
DBCC traceon(1204,-1)
```

Next I confirmed that the trace was enabled with the below command:

```
DBCC Tracestatus(-1)
```

The output shown below indicated that the trace flag 1204 was enabled.

```
TraceFlag Status Global Session
--------- ------ ------ -------
1204      1      1      0
```

2. I then gathered the deadlock details from the SQL error log with the command 'xp_readerrorlog'. One instance of deadlock occurred was given below:

```
Deadlock encountered .... Printing deadlock information
Wait-for graph
Node:1 PAG: 9:1:18061 CleanCnt:2 Mode: SIU Flags: 0x2
Grant List 1::
Grant List 2::
Owner:0x27c007e0 Mode: S Flg:0x0 Ref:0 Life:00000001 SPID:84 ECID:0
SPID: 84 ECID: 0 Statement Type: UPDATE Line #: 11
Input Buf: RPC Event: dbo.Example_Stored_proc
Requested By: 0
ResType:LockOwner Stype:'OR' Mode: IX SPID:78 ECID:0 Ec:(0x44AA55F0) Value:0x3affcd00 Cost:(0/0)
Node:2 PAG: 9:1:18134 CleanCnt:2 Mode: SIU Flags: 0x2
Grant List 1::
Owner:0x28e6f060 Mode: S Flg:0x0 Ref:0 Life:00000001 SPID:78 ECID:0
SPID: 78 ECID: 0 Statement Type: UPDATE Line #: 11
Input Buf: RPC Event: dbo. Example_Stored_proc
Grant List 2::
Requested By:
ResType:LockOwner Stype:'OR' Mode: IX SPID:84 ECID:0 Ec:(0x239955F0) Value:0x3affc940 Cost:(0/0)
Victim Resource Owner:
ResType:LockOwner Stype:'OR' Mode: IX SPID:84 ECID:0 Ec:(0x239955F0) Value:0x3affc940 Cost:(0/0)
```

3. From the captured details, it was found that the deadlocks were occurring due to UPDATE statements in the stored procedure 'Example_stored_proc' while trying to get an Intent exclusive lock on the pages.

4. My next step was to read the contents of the stored procedure 'Example_stored_proc' through sp_helptext.

5. The procedure was calling around 5 procedures and each of those were calling 4 more procedures.

6. As going through all the procedures to identify the deadlock statements was difficult, I was able to get to the correct update statement through the Page numbers in the deadlock details captured earlier. The deadlock details captured had the page numbers that are involved in the deadlock as shown below:

Node:1 PAG: 9:1:18061 CleanCnt:2 Mode: SIU Flags: 0x2

7. I gathered the Page numbers that were being deadlocked and used the undocumented DBCC command 'DBCC Page' to get the table name associated with the page. The details of the command syntax are in this link:http://support.microsoft.com/kb/83065

Though the article refers to earlier SQL versions, the command is still valid in SQL Server 2000 and provides the page information.

I then used this command to get the page details:

```
DBCC page(9,1,18061,0)
```

The parameters were taken from the deadlock details shown above. The output obtained was pasted below. It contains the object name associated with that page. Under the PAGE HEADER, the value m_objId gives the table name associated with the deadlock:

```
PAGE: (1:18061)
---------------
BUFFER:
-------
BUF @0x01665900
---------------
bpage = 0x1DF58000 bhash = 0x00000000 bpageno = (1:18061)
bdbid = 9 breferences = 1 bstat = 0xb
bspin = 0 bnext = 0x00000000
PAGE HEADER:
------------
Page @0x1DF58000
----------------
m_pageId = (1:18061) m_headerVersion = 1 m_type = 1
m_typeFlagBits = 0x0 m_level = 0 m_flagBits = 0x8000
m_objId = 1013578649 m_indexId = 0 m_prevPage = (0:0)
m_nextPage = (0:0) pminlen = 52 m_slotCnt = 82
m_freeCnt = 3075 m_freeData = 5009 m_reservedCnt = 0
m_lsn = (2689:87968:2) m_xactReserved = 0 m_xdesId = (0:0)
m_ghostRecCnt = 0 m_tornBits = 0
.................
```

Once the object id was obtained, table name was found out with the command:

```
Use DBNAME
Select object_name(OBJECT_ID)
```

8. I then reviewed the code of the dependent stored procedures on the table and found one update procedure among the list.

9. The code of the update procedure was similar to that shown below:

```
Update TABLE
 Set Column1 =@C1,
 Column2=@C2,
 Column2=@C3,
 Where ID = @id
```

10. I then verified the indexes on the table and found that the table did not have an index on the ID column. This was forcing the update to perform a table scan causing it to take an INTENT exclusive lock on all the pages thereby causing

the deadlock in the process.

11. As there was only one non-clustered index existing on that table, I created a clustered index on the table on the ID column with this command:

```
IF NOT EXISTS (SELECT name FROM sysindexes
 WHERE name = 'IX_Clustered_index_column')
CREATE CLUSTERED INDEX IX_Clustered_index_column
  ON dbo.TABLE(ID) ON [PRIMARY]
```

12. After the index creation, the intent exclusive lock was not requested on all the pages of the update statement and as a result the deadlocks got resolved.

Conclusion

On further discussion with the application team about the reason for the deadlock occurrence after the upgrade performed by them, I came to know the following:

- Before the upgrade, for every insertion, one record was inserted or updated in the table identified in the deadlock
- As part of the upgrade, the application team performed modifications to the stored procedure code such that every insertion may cause up to 10 updates in the table.

This confirmed the reason why the deadlocks started occurring after the upgrade. As the table access increased heavily due to table scans, transactions took a longer time for each update thereby causing the deadlock scenario.