Identify the cause of SQL Server blocking

By: Basit Farooq | Read Comments (10) | Related Tips: More > Locking and Blocking

Problem

In my previous article (Different techniques to identify blocking in SQL Server), I discussed locks and blocks, and presented you with an overview on how to troubleshoot and resolve blocks using dynamic management views and Activity Monitor. After I wrote this article, I received several emails from readers asking how they can use the information returned by these dynamic management views (DMVs) to identify SPIDs and other useful information about the processes that are actually causing blocking on a SQL Server instance. In this tip, I will share the query which I have written using these dynamic management views (DMVs) that will help you to quickly identify SPIDs and other useful information about the processes that are causing blocking on SQL Server instance.

Solution

As discussed in my previous article, SQL Server has a rich set of dynamic management views (quickly identify locking and blocking in SQL Server. That is why, writing such a query is quite the following dynamic management views (DMVs) for my query.

- sys.dm_os_waiting_tasks – Returns information about blocked and blocking proces
- sys.dm_exec_sessions – Returns information about authenticated sessions on SQ
- sys.dm_exec_requests – Returns the detailed information about the requests cu
- sys.dm_tran_locks - Returns the information about the current locks and the pr
- sys.dm_exec_query_plan – Returns the showplan for the query in XML format.
- sys.dm_exec_sql_text – Returns the text of T-SQL batch.

The following is the query, which I have written using these dynamic management views (DMVs) that will help you to quickly identify the SPIDs and other information about the processes that are causing the blocking on SQL Server instance. This query returns the comprehensive information about the blocking and waiting processes, which is useful for troubleshooting SQL Server locking and blocking issues. This query is also a good way to analyze detailed information about locks, and help you to identify the cause of a large number of blocks.

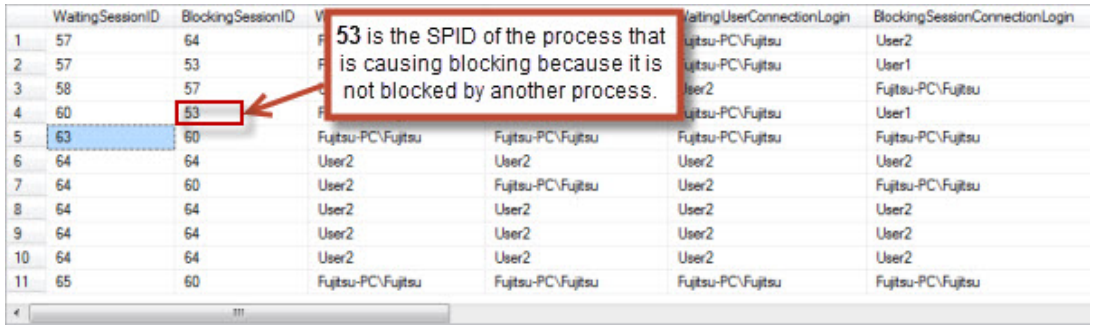
```
WITH [Blocking]
AS (SELECT w.[session_id]
   ,s.[original_login_name]
```

```
   ,s.[login_name]
   ,w.[wait_duration_ms]
   ,w.[wait_type]
   ,r.[status]
   ,r.[wait_resource]
   ,w.[resource_description]
   ,s.[program_name]
   ,w.[blocking_session_id]
   ,s.[host_name]
   ,r.[command]
   ,r.[percent_complete]
   ,r.[cpu_time]
   ,r.[total_elapsed_time]
   ,r.[reads]
   ,r.[writes]
   ,r.[logical_reads]
   ,r.[row_count]
   ,q.[text]
   ,q.[dbid]
   ,p.[query_plan]
   ,r.[plan_handle]
 FROM [sys].[dm_os_waiting_tasks] w
 INNER JOIN [sys].[dm_exec_sessions] s ON w.[session_id] = s.[session_id]
 INNER JOIN [sys].[dm_exec_requests] r ON s.[session_id] = r.[session_id]
 CROSS APPLY [sys].[dm_exec_sql_text](r.[plan_handle]) q
 CROSS APPLY [sys].[dm_exec_query_plan](r.[plan_handle]) p
 WHERE w.[session_id] > 50
  AND w.[wait_type] NOT IN ('DBMIRROR_DBM_EVENT'
      ,'ASYNC_NETWORK_IO'))
SELECT b.[session_id] AS [WaitingSessionID]
      ,b.[blocking_session_id] AS [BlockingSessionID]
      ,b.[login_name] AS [WaitingUserSessionLogin]
      ,s1.[login_name] AS [BlockingUserSessionLogin]
      ,b.[original_login_name] AS [WaitingUserConnectionLogin]
      ,s1.[original_login_name] AS [BlockingSessionConnectionLogin]
      ,b.[wait_duration_ms] AS [WaitDuration]
      ,b.[wait_type] AS [WaitType]
      ,t.[request_mode] AS [WaitRequestMode]
      ,UPPER(b.[status]) AS [WaitingProcessStatus]
      ,UPPER(s1.[status]) AS [BlockingSessionStatus]
      ,b.[wait_resource] AS [WaitResource]
      ,t.[resource_type] AS [WaitResourceType]
      ,t.[resource_database_id] AS [WaitResourceDatabaseID]
      ,DB_NAME(t.[resource_database_id]) AS [WaitResourceDatabaseName]
      ,b.[resource_description] AS [WaitResourceDescription]
      ,b.[program_name] AS [WaitingSessionProgramName]
      ,s1.[program_name] AS [BlockingSessionProgramName]
      ,b.[host_name] AS [WaitingHost]
      ,s1.[host_name] AS [BlockingHost]
      ,b.[command] AS [WaitingCommandType]
      ,b.[text] AS [WaitingCommandText]
      ,b.[row_count] AS [WaitingCommandRowCount]
      ,b.[percent_complete] AS [WaitingCommandPercentComplete]
      ,b.[cpu_time] AS [WaitingCommandCPUTime]
      ,b.[total_elapsed_time] AS [WaitingCommandTotalElapsedTime]
      ,b.[reads] AS [WaitingCommandReads]
      ,b.[writes] AS [WaitingCommandWrites]
      ,b.[logical_reads] AS [WaitingCommandLogicalReads]
      ,b.[query_plan] AS [WaitingCommandQueryPlan]
      ,b.[plan_handle] AS [WaitingCommandPlanHandle]
```

```
FROM [Blocking] b
INNER JOIN [sys].[dm_exec_sessions] s1
ON b.[blocking_session_id] = s1.[session_id]
INNER JOIN [sys].[dm_tran_locks] t
ON t.[request_session_id] = b.[session_id]
WHERE t.[request_status] = 'WAIT'
GO
```

Sample Output

To examine the results of this query, run it on SQL Server where you are experiencing blocks. For example, when I executed this query on my test SQL Server where I'm deliberately running some code to cause blocking, it brings the following results (*Note: To fit the resultset, I've split the resultset into seven images*):



	WaitingSessionID	BlockingSessionID	W		WaitingUserConnectionLogin	BlockingSessionConnectionLogin
1	57	64	F		ujitsu-PC\Fujitsu	User2
2	57	53	F		ujitsu-PC\Fujitsu	User1
3	58	57	C		ser2	Fujitsu-PC\Fujitsu
4	60	53	F		ujitsu-PC\Fujitsu	User1
5	63	60	Fujitsu-PC\Fujitsu	Fujitsu-PC\Fujitsu	Fujitsu-PC\Fujitsu	Fujitsu-PC\Fujitsu
6	64	64	User2	User2	User2	User2
7	64	60	User2	Fujitsu-PC\Fujitsu	User2	Fujitsu-PC\Fujitsu
8	64	64	User2	User2	User2	User2
9	64	64	User2	User2	User2	User2
10	64	64	User2	User2	User2	User2
11	65	60	Fujitsu-PC\Fujitsu	Fujitsu-PC\Fujitsu	Fujitsu-PC\Fujitsu	Fujitsu-PC\Fujitsu

	WaitDuration	WaitType	WaitRequestMode	WaitingProcessStatus	BlockingSessionStatus	WaitResource	WaitResourceType
1	497188	LCK_M_S	S	SUSPENDED	RUNNING	PAGE: 6:1:1472	PAGE
2	497188	LCK_M_S	S	SUSPENDED	SLEEPING	PAGE: 6:1:1472	PAGE
3	372270	LCK_M_IX	IX	SUSPENDED	RUNNING	PAGE: 6:1:1472	PAGE
4	338030	LCK_M_U	U	SUSPENDED	SLEEPING	KEY: 6:72057594045595648 (8194443284a0)	KEY
5	279378	LCK_M_S	S	SUSPENDED	RUNNING	KEY: 6:72057594045595648 (8194443284a0)	KEY
6	249741	CXPACKET	U	SUSPENDED	RUNNING	KEY: 6:72057594045595648 (8194443284a0)	KEY
7	249742	LCK_M_U	U	SUSPENDED	RUNNING	KEY: 6:72057594045595648 (8194443284a0)	KEY
8	249741	CXPACKET	U	SUSPENDED	RUNNING	KEY: 6:72057594045595648 (8194443284a0)	KEY
9	249742	CXPACKET	U	SUSPENDED	RUNNING	KEY: 6:72057594045595648 (8194443284a0)	KEY
10	249743	CXPACKET	U	SUSPENDED	RUNNING	KEY: 6:72057594045595648 (8194443284a0)	KEY
11	218525	LCK_M_U	U	SUSPENDED	RUNNING	KEY: 6:72057594045595648 (8194443284a0)	KEY

	WaitResourceDatabaseID	WaitResourceDatabaseName	WaitResourceDescription
1	6	AdventureWorks2012	pagelock fileid=1 pageid=1472 dbid=6 subresource=FULL id=lock26a9c4900 mode=IX associatedObjectId=72057594045595648
2	6	AdventureWorks2012	pagelock fileid=1 pageid=1472 dbid=6 subresource=FULL id=lock26a9c4900 mode=IX associatedObjectId=72057594045595648
3	6	AdventureWorks2012	pagelock fileid=1 pageid=1472 dbid=6 subresource=FULL id=lock26a9c4900 mode=IX associatedObjectId=72057594045595648
4	6	AdventureWorks2012	keylock hobtid=72057594045595648 dbid=6 id=lock26a9c4d80 mode=X associatedObjectId=72057594045595648
5	6	AdventureWorks2012	keylock hobtid=72057594045595648 dbid=6 id=lock26a9c4d80 mode=X associatedObjectId=72057594045595648
6	6	AdventureWorks2012	exchangeEvent id=Pipe264ad2b90 WaitType=e_waitPipeNewRow nodeId=5
7	6	AdventureWorks2012	keylock hobtid=72057594045595648 dbid=6 id=lock26a9c4d80 mode=X associatedObjectId=72057594045595648
8	6	AdventureWorks2012	exchangeEvent id=Pipe264b241a0 WaitType=e_waitPipeNewRow nodeId=5
9	6	AdventureWorks2012	exchangeEvent id=Pipe264ad3670 WaitType=e_waitPipeNewRow nodeId=5
10	6	AdventureWorks2012	exchangeEvent id=Pipe264ad20c0 WaitType=e_waitPipeNewRow nodeId=5
11	6	AdventureWorks2012	keylock hobtid=72057594045595648 dbid=6 id=lock26a9c4d80 mode=X associatedObjectId=72057594045595648

	WaitingSessionProgramName	BlockingSessionProgramName	WaitingHost	BlockingHost	WaitingCommandType
1	Microsoft SQL Server Management Studio - Query	Microsoft SQL Server Management Studio - Query	FUJITSU-PC	FUJITSU-PC	SELECT
2	Microsoft SQL Server Management Studio - Query	Microsoft SQL Server Management Studio - Query	FUJITSU-PC	FUJITSU-PC	SELECT
3	Microsoft SQL Server Management Studio - Query	Microsoft SQL Server Management Studio - Query	FUJITSU-PC	FUJITSU-PC	UPDATE
4	Microsoft SQL Server Management Studio - Query	Microsoft SQL Server Management Studio - Query	FUJITSU-PC	FUJITSU-PC	DELETE
5	Microsoft SQL Server Management Studio - Query	Microsoft SQL Server Management Studio - Query	FUJITSU-PC	FUJITSU-PC	SELECT
6	Microsoft SQL Server Management Studio - Query	Microsoft SQL Server Management Studio - Query	FUJITSU-PC	FUJITSU-PC	UPDATE
7	Microsoft SQL Server Management Studio - Query	Microsoft SQL Server Management Studio - Query	FUJITSU-PC	FUJITSU-PC	UPDATE
8	Microsoft SQL Server Management Studio - Query	Microsoft SQL Server Management Studio - Query	FUJITSU-PC	FUJITSU-PC	UPDATE
9	Microsoft SQL Server Management Studio - Query	Microsoft SQL Server Management Studio - Query	FUJITSU-PC	FUJITSU-PC	UPDATE
10	Microsoft SQL Server Management Studio - Query	Microsoft SQL Server Management Studio - Query	FUJITSU-PC	FUJITSU-PC	UPDATE
11	Microsoft SQL Server Management Studio - Query	Microsoft SQL Server Management Studio - Query	FUJITSU-PC	FUJITSU-PC	UPDATE

	WaitingCommandText	WaitingCommandRowCount	WaitingCommandPercentComplete
1	BEGIN TRANSACTION SELECT * FROM [AdventureWorks2012].[Person].[Person]	0	0
2	BEGIN TRANSACTION SELECT * FROM [AdventureWorks2012].[Person].[Person]	0	0
3	(@1 int,@2 tinyint)UPDATE [AdventureWorks2012].[Person].[Person] set [EmailPromotion] = @1 WHER...	0	0
4	BEGIN TRANSACTION DELETE FROM [AdventureWorks2012].[Person].[Person] WHERE EmailPr...	0	0
5	/****** Script for SelectTopNRows command from SSMS ******/ SELECT TOP 1000 [BusinessEntityID] ...	0	0
6	BEGIN TRANSACTION UPDATE [AdventureWorks2012].[Person].[Person] SET [FirstName] = 'J...	0	0
7	BEGIN TRANSACTION UPDATE [AdventureWorks2012].[Person].[Person] SET [FirstName] = 'J...	0	0
8	BEGIN TRANSACTION UPDATE [AdventureWorks2012].[Person].[Person] SET [FirstName] = 'J...	0	0
9	BEGIN TRANSACTION UPDATE [AdventureWorks2012].[Person].[Person] SET [FirstName] = 'J...	0	0
10	BEGIN TRANSACTION UPDATE [AdventureWorks2012].[Person].[Person] SET [FirstName] = 'J...	0	0
11	BEGIN TRANSACTION UPDATE [AdventureWorks2012].[Person].[Person] SET [FirstName] = '...	0	0

	WaitingCommandCPUTime	WaitingCommandTotalElapsedTime	WaitingCommandReads	WaitingCommandWrites	WaitingCommandLogicalReads
1	15	497191	491	0	5
2	15	497191	491	0	5
3	0	372302	0	0	9
4	94	339044	3108	5	2435
5	0	279414	0	0	5
6	0	249902	0	0	10
7	0	249902	0	0	10
8	0	249902	0	0	10
9	0	249902	0	0	10
10	0	249902	0	0	10
11	15	218572	0	0	13

	WaitingCommandQueryPlan	WaitingCommandPlanHandle
1	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.2" Build="11.0.2100...	0x06000100EC56073160EB8166020000000100000000000000...
2	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.2" Build="11.0.2100...	0x06000100EC56073160EB8166020000000100000000000000...
3	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.2" Build="11.0.2100...	0x06000100AA812218F06B0266020000000100000000000000...
4	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.2" Build="11.0.2100...	0x060006007DF3362DA0F2816602000000010000000000000...
5	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.2" Build="11.0.2100...	0x06000100AF8DFD29A0686664020000000100000000000...
6	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.2" Build="11.0.2100...	0x060001003EDFD422407D026602000000010000000000000...
7	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.2" Build="11.0.2100...	0x060001003EDFD422407D026602000000010000000000000...
8	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.2" Build="11.0.2100...	0x060001003EDFD422407D026602000000010000000000000...
9	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.2" Build="11.0.2100...	0x060001003EDFD422407D026602000000010000000000000...
10	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.2" Build="11.0.2100...	0x060001003EDFD422407D026602000000010000000000000...
11	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.2" Build="11.0.2100...	0x06000100D6F4A412E06F666402000000010000000000000...

The following are the columns returned by this query:

- **WaitingSessionID** – The SPID of the waiting session.
- **BlockingSessionID** – The SPID of the blocking session.
- **WaitingSessionUserLogin** – The user session login name under which waiting session is currently executing.
- **BlockingSessionUserLogin** – The user session login name under which blocking session is currently executing.
- **WaitingUserConnectionLogin** – The login name that the user used to create waiting session.
- **BlockingSessionConnectionLogin** – The login name that the user used to create waiting session.

- **WaitDuration** – Waiting process wait time in milliseconds.
- **WaitType** – Type of wait.
- **WaitRequestMode** – Mode of the wait request.
- **WaitingProcessStatus** – The status of waiting process.
- **BlockingSessionStatus** – The status of blocking process.
- **WaitResource** – The name of the resource request is waiting for.
- **WaitResourceType** – The type of the resource request is waiting for.
- **WaitResourceDatabaseID** – The database id of the database in which the requested resource exists.
- **WaitResourceDatabaseName** – The name of the database in which the requested resource exists.
- **WaitResourceDescription** – The detailed description of the waiting resource.
- **WaitingSessionProgramName** – The name of the program that initiated the waiting session.
- **BlockingSessionProgramName** – The name of the program that initiated the blocking session.
- **WaitingHost** – The name of the workstation that is specific to waiting session.
- **BlockingHost** – The name of the workstation that is specific to blocking session.
- **WaitingCommandType** – The type of waiting session command.
- **WaitingCommandText** – The text of waiting session command.
- **WaitingCommandRowCount** – Expected number of rows return by the waiting session.
- **WaitingCommandPercentComplete** – Percentage of the waiting request client.
- **WaitingCommandCPUTime** – CPU time used by waiting session.
- **WaitingCommandTotalElapsedTime** – The total time elapsed in milliseconds since the waiting request arrived.
- **WaitingCommandReads** – The number of reads performed by the waiting session request.
- **WaitingCommandWrites** – The number of writes performed by the waiting session request.
- **WaitingCommandLogicalReads** – The number of logical reads performed by the waiting session request.
- **WaitingCommandQueryPlan** – Waiting command execution plan.
- **WaitingCommandPlanHandle** – Plan handle of the waiting session command.

As you can see from above resultset, that process 53 listed BlockingSessionID column of row 4 is not blocked by another process, hence identified as the SPID that is the cause of the blocking on my test SQL Server instance.

Next Steps

- Transaction locks are the most common cause of blocked processes. The stronger (least concurrent) the isolation level, the more likely it is to cause a blocked process.
- Try to use less granular lock for your queries, as the less granular the lock, the more likely a blocked process or deadlock will not occur. For example, if the entire table is locked, there is a higher likelihood of blocks than if only a single row is locked.
- Revisit your database design because bad database design could be potential reason for excessive locking and blocking.
- Check out these tips to learn more about locking and blocking:
  - Understanding SQL Server Locking
  - Understanding SQL Server Blocking
  - Locking and Blocking Tips category

Last Update: 4/18/2013

About the author

Basit is a Senior Database Administrator and has worked in the IT industry for 11+ years.

View all my tips

We Recommend

- More SQL Server DBA Tips...

Comments and Feedback:

Thursday, April 18, 2013 - 12:20:25 AM - Gopalakrishnan Read The Tip

Excellent one to identify blocking. Thanks a lot Basit.

Thursday, April 18, 2013 - 8:18:45 AM - Chris W Read The Tip

That is great for instances when you are currently having an issue with blocking. How do I look at an issue that occurred overnight and is passed (i.e. that is no longer blocking but I have the timestamp when the blocking occurred).

Thursday, April 18, 2013 - 8:56:34 AM - Hassan Parthasarathy Read The Tip

Very nice article! Worth looking!
Thanks
Partha

Thursday, April 18, 2013 - 9:01:33 AM - Hossam Abdelwahab Read The Tip

Thanks very much ... very detailed informative helpfull one...

Thursday, April 18, 2013 - 7:55:14 PM - Cezar Read The Tip

Hi Basit, thanks by the information. I did a query, that i think can help us to find the spid blocker. If we will use the query how a table (between parenthesis) + alias. Then we wont need to find the spid blocker looking for visually.

```
Select BlockingSessionID From (query) q1 Where NotExists (

Select q2.WaitingSessionID From (query) q2 Where q2.WaitingSessionID = q1.BlockingSessionID

)
```

Wednesday, May 01, 2013 - 2:37:20 AM - Andrey Read The Tip

It is much easier to use sp_WhoIsActive to determine the blocking sequences. I use it.

Wednesday, May 01, 2013 - 9:06:36 AM - SeaQuill Read The Tip

Why are there rows such as 8, 9, and 10 where WaitingSessionID = BlockingSessionID?

Wednesday, May 01, 2013 - 9:56:35 AM - Tim Greenan Read The Tip

Hi Basit, thanks for sharing this is very helpful. I tested it on my dev server and I was able to create blocking that was not detected by you query. The request_status of the blocked process was 'CONVERT'. I think that if you change the WHERE clause to t.[request_status] <> 'GRANT' that should fix it.

Amit Bansal posted an article describing the CONVERT request status with an example of how to create a convert wait/block - http://www.sqlservergeeks.com/blogs/AmitBansal/sql-server-bi/298/sql-server-request_status-in-sys-dm_tran_locks-grant-wait-and-convert

Thanks,

Tim

Wednesday, May 01, 2013 - 11:36:56 AM - George Shouse Read The Tip

From your article:

> Try to use less granular lock for your queries, as the less granular the lock, the more likely a blocked process or deadlock will not occur. For example, if the entire table is locked, there is a higher likelihood of blocks than if only a single row is locked.

A row lock is more granular than a table lock. I think you want "Try to use a **more** granular lock for your queries, as the more granular the lock, the more likely a blocked process or deadlock will not occur."

http://en.wikipedia.org/wiki/Granularity

Wednesday, May 01, 2013 - 8:14:12 PM - Ann Read The Tip

Why on CTE_query_definition

```
CROSSAPPLY [sys].[dm_exec_sql_text](r.[plan_handle]) q
```

not r.sql_handle?

Thanks!

Post a Comment or Question

Keep it clean and stay on the subject or we may delete your comment.
Your email address is not published. Required fields are marked with an asterisk (*)

*Name *Email Notify for updates Signup for our newsletter

Comments

*** NOTE *** - If you want to include code from SQL Server Management Studio (SSMS) in your post, please copy the code from SSMS and paste the code into a text editor like NotePad before copying the code below to remove the SSMS formatting.

*Enter Code M747

Submit Reset

Sponsor Information