How to capture a Query Command Timeout in TSQL

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:28 AM PST

Contents

How to capture a Query Command Timeout in Transact-SQL

Issue

In some cases, the customer is seeing the query timeout (command timeout) only in the application, but cannot tell what exact statement is causing the timeout.

In the telemetry, the timeout is exposed in Kusto's `MonWiQdsExecStats` column `exec_type` with the value "3". But how can the customer see this for themselves?

Mitigation

The customer can capture query timeouts (command timeouts) through Extended events in Azure SQL Database. Detailed steps are available through the Code samples using either the ring buffer for ad-hoc capture or a file target for persisted capture in Azure storage.

Here is a sample script using the ring buffer, following the steps from Ring Buffer target code for extended events in Azure SQL Database. The ring buffer is easier to use because you simply have to run the script; with a file target, you also have to deal with storage connectivity and permissions, which might not be easy or possible for the customer contact you are working with.

Extended Events using a Ring Buffer target

The sample script below captures the minimum information related to query timeouts:

- Attention (timeout) event
- Blocked process report
- Error_Reported event

- RPC_Completed and SQL_Batch_Completed events

The script also has commented additional events that might help you with narrowing down further, e.g. RPC_Statement_Starting and SQL_Statement_completed. Uncommented them as needed to get further details, at the cost of bloated capture data.

Create Event Session

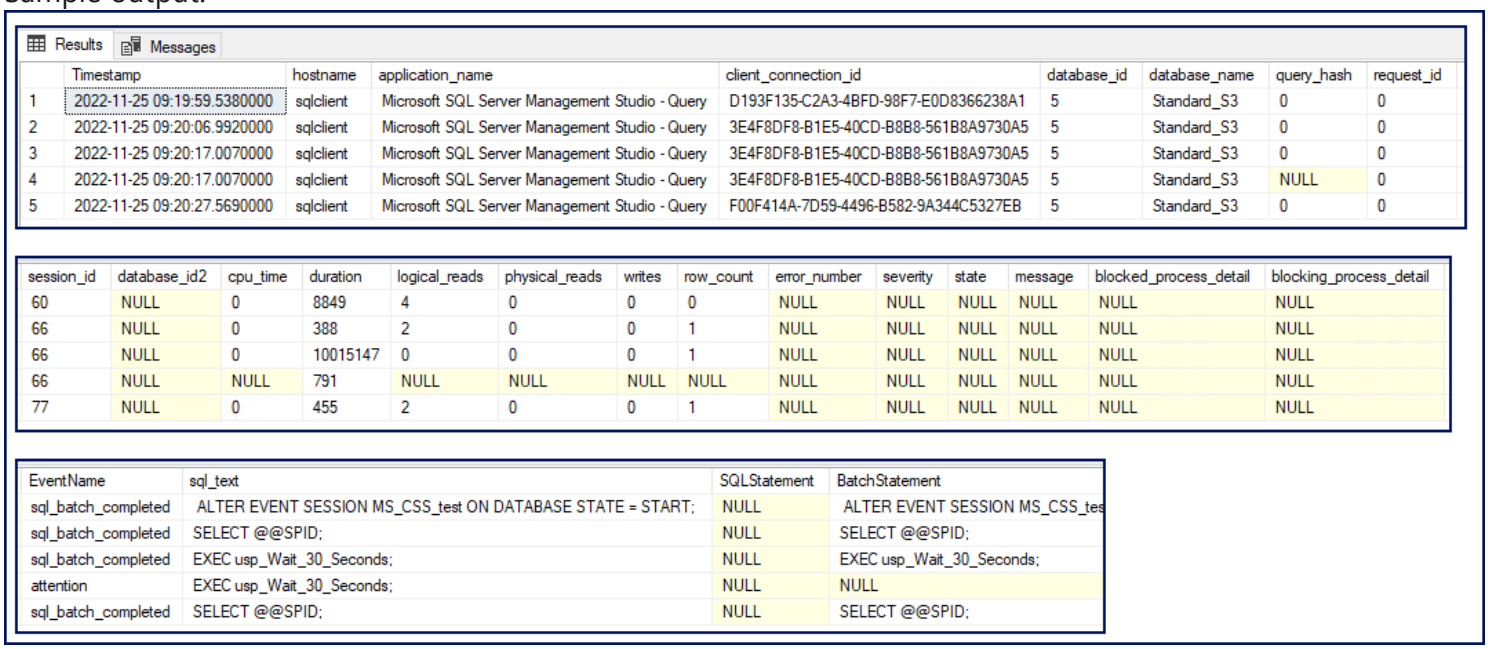
```
CREATE EVENT SESSION [MS_attention_timeout] ON DATABASE

ADD EVENT sqlserver.blocked_process_report(
ACTION(sqlserver.client_app_name,sqlserver.client_connection_id,sqlserver.client_hostname,sqlserver.database_i

ADD EVENT sqlserver.error_reported(
ACTION(sqlserver.client_app_name,sqlserver.client_connection_id,sqlserver.client_hostname,sqlserver.database_i
    WHERE ([package0].[greater_than_uint64]([sqlserver].[database_id],(4)) AND [package0].[equal_boolean]([sql
/*
ADD EVENT sqlserver.login(
ACTION(sqlserver.client_app_name,sqlserver.client_connection_id,sqlserver.client_hostname,sqlserver.database_i

ADD EVENT sqlserver.logout(
ACTION(sqlserver.client_app_name,sqlserver.client_connection_id,sqlserver.client_hostname,sqlserver.database_i

ADD EVENT sqlserver.rpc_starting(
ACTION(sqlserver.client_app_name,sqlserver.client_connection_id,sqlserver.client_hostname,sqlserver.database_i
*/
ADD EVENT sqlserver.rpc_completed(
ACTION(sqlserver.client_app_name,sqlserver.client_connection_id,sqlserver.client_hostname,sqlserver.database_i
    WHERE ([package0].[greater_than_uint64]([sqlserver].[database_id],(4)) AND [package0].[equal_boolean]([sql
/*
ADD EVENT sqlserver.sql_batch_starting(
ACTION(mdmtargetpkg.mdmget_TimeStampUTC,sqlserver.client_app_name,sqlserver.client_connection_id,sqlserver.cli
*/
ADD EVENT sqlserver.sql_batch_completed(
ACTION(sqlserver.client_app_name,sqlserver.client_connection_id,sqlserver.client_hostname,sqlserver.database_i
    WHERE ([package0].[greater_than_uint64]([sqlserver].[database_id],(4)) AND [package0].[equal_boolean]([sql
/*
ADD EVENT sqlserver.sp_statement_starting(
ACTION(sqlserver.client_app_name,sqlserver.client_connection_id,sqlserver.client_hostname,sqlserver.database_i

ADD EVENT sqlserver.sp_statement_completed(SET collect_object_name=(1)
ACTION(sqlserver.client_app_name,sqlserver.client_connection_id,sqlserver.client_hostname,sqlserver.database_i
    WHERE ([package0].[greater_than_uint64]([sqlserver].[database_id],(4)) AND [package0].[equal_boolean]([sql

ADD EVENT sqlserver.sql_statement_starting(
ACTION(mdmtargetpkg.mdmget_TimeStampUTC,sqlserver.client_app_name,sqlserver.client_connection_id,sqlserver.cli

ADD EVENT sqlserver.sql_statement_completed(
ACTION(sqlserver.client_app_name,sqlserver.client_connection_id,sqlserver.client_hostname,sqlserver.database_i
    WHERE ([package0].[greater_than_uint64]([sqlserver].[database_id],(4)) AND [package0].[equal_boolean]([sql
*/

ADD EVENT sqlserver.attention(
ACTION(sqlserver.client_app_name,sqlserver.client_connection_id,sqlserver.client_hostname,sqlserver.database_i

ADD TARGET package0.ring_buffer(SET max_events_limit=(10000),max_memory=(102400))
WITH (MAX_MEMORY=4096 KB,EVENT_RETENTION_MODE=ALLOW_SINGLE_EVENT_LOSS,MAX_DISPATCH_LATENCY=30 SECONDS,MAX_EVEN
GO

ALTER EVENT SESSION MS_attention_timeout ON DATABASE STATE = START;

-- (wait until the timeout occurs)
```

Read the event session

Run this after the timeout has occurred - check for the "Attention" event. If you notice blocking, you can get further details from the blocking report by un-commenting the additional blocking columns.

```
WITH CTE AS (
    SELECT CAST(xet.target_data AS XML) AS [target_data_XML], xe.name as SessionTargetName
        FROM sys.dm_xe_database_session_targets AS xet
        INNER JOIN sys.dm_xe_database_sessions AS xe ON (xe.address = xet.event_session_address)
--      WHERE xe.name = 'MS_attention_timeout' -- only add if there are several active ringbuffers
)
, CTE2 AS (
       SELECT T2.EventData.query('.').value('(/event/@timestamp)[1]', 'DateTime2') AS Timestamp
       , T2.EventData.query('.').value('(/event/action[@name=''client_hostname''])[1]', 'sysname') AS hostname
       , T2.EventData.query('.').value('(/event/action[@name=''client_app_name''])[1]', 'sysname') AS applicat
       , T2.EventData.query('.').value('(/event/action[@name=''client_connection_id''])[1]', 'sysname') AS cli
       , T2.EventData.query('.').value('(/event/action[@name=''database_id''])[1]', 'int') AS database_id
       , T2.EventData.query('.').value('(/event/action[@name=''database_name''])[1]', 'sysname') AS database_n
       , T2.EventData.query('.').value('(/event/action[@name=''query_hash''])[1]', 'nvarchar(50)') AS query_ha
       , T2.EventData.query('.').value('(/event/action[@name=''request_id''])[1]', 'sysname') AS request_id
       , T2.EventData.query('.').value('(/event/action[@name=''session_id''])[1]', 'int') AS session_id
       , T2.EventData.query('.').value('(/event/data[@name=''source_database_id''])[1]', 'bigint') AS database
       , T2.EventData.query('.').value('(/event/data[@name=''cpu_time''])[1]', 'bigint') AS cpu_time
       , T2.EventData.query('.').value('(/event/data[@name=''duration''])[1]', 'bigint') AS duration
       , T2.EventData.query('.').value('(/event/data[@name=''logical_reads''])[1]', 'bigint') AS logical_reads
       , T2.EventData.query('.').value('(/event/data[@name=''physical_reads''])[1]', 'bigint') AS physical_rea
       , T2.EventData.query('.').value('(/event/data[@name=''writes''])[1]', 'bigint') AS writes
       , T2.EventData.query('.').value('(/event/data[@name=''row_count''])[1]', 'bigint') AS row_count
       , T2.EventData.query('.').value('(/event/data[@name=''error_number''])[1]', 'bigint') AS error_number
       , T2.EventData.query('.').value('(/event/data[@name=''severity''])[1]', 'bigint') AS severity
       , T2.EventData.query('.').value('(/event/data[@name=''state''])[1]', 'nvarchar(50)') AS state
       , T2.EventData.query('.').value('(/event/data[@name=''message''])[1]', 'nvarchar(2000)') AS message
       , T2.EventData.query('.').value('(/event/data[@name=''blocked_process'']/value/blocked-process-report/b
       , T2.EventData.query('.').value('(/event/data[@name=''blocked_process'']/value/blocked-process-report/b
/*
       , T2.EventData.query('.').value('(/event/data[@name=''blocked_process'']/value/blocked-process-report/b
       , T2.EventData.query('.').value('(/event/data[@name=''blocked_process'']/value/blocked-process-report/b
       , T2.EventData.query('.').value('(/event/data[@name=''blocked_process'']/value/blocked-process-report/b
       , T2.EventData.query('.').value('(/event/data[@name=''blocked_process'']/value/blocked-process-report/b
       , T2.EventData.query('.').value('(/event/data[@name=''blocked_process'']/value/blocked-process-report/b
*/
       , T2.EventData.query('.').value('(/event/@name)[1]', 'varchar(50)') AS EventName
       , T2.EventData.query('.').value('(/event/action[@name=''sql_text''])[1]', 'nvarchar(2000)') AS sql_text
       , T2.EventData.query('.').value('(/event/data[@name=''statement''])[1]', 'nvarchar(2000)') AS SQLStatem
       , T2.EventData.query('.').value('(/event/data[@name=''batch_text''])[1]', 'nvarchar(2000)') AS BatchSta
       FROM CTE
       CROSS Apply [target_data_XML].nodes('/RingBufferTarget/event') AS T2(EventData)
--       CROSS Apply [target_data_XML].nodes('/RingBufferTarget/event[@name=''blocked_process_report'']') AS T
--       CROSS Apply [target_data_XML].nodes('/RingBufferTarget/event[@name=''sql_statement_starting'']') AS T
--       CROSS Apply [target_data_XML].nodes('/RingBufferTarget/event[@name=''sql_statement_completed'']') AS
--       CROSS Apply [target_data_XML].nodes('/RingBufferTarget/event[@name=''sp_statement_starting'']') AS T2
--       CROSS Apply [target_data_XML].nodes('/RingBufferTarget/event[@name=''sp_statement_completed'']') AS T
)
SELECT * FROM CTE2
```

Sample output:



Results | Messages

	Timestamp	hostname	application_name	client_connection_id	database_id	database_name	query_hash	request_id
1	2022-11-25 09:19:59.5380000	sqlclient	Microsoft SQL Server Management Studio - Query	D193F135-C2A3-4BFD-98F7-E0D8366238A1	5	Standard_S3	0	0
2	2022-11-25 09:20:06.9920000	sqlclient	Microsoft SQL Server Management Studio - Query	3E4F8DF8-B1E5-40CD-B8B8-561B8A9730A5	5	Standard_S3	0	0
3	2022-11-25 09:20:17.0070000	sqlclient	Microsoft SQL Server Management Studio - Query	3E4F8DF8-B1E5-40CD-B8B8-561B8A9730A5	5	Standard_S3	0	0
4	2022-11-25 09:20:17.0070000	sqlclient	Microsoft SQL Server Management Studio - Query	3E4F8DF8-B1E5-40CD-B8B8-561B8A9730A5	5	Standard_S3	NULL	0
5	2022-11-25 09:20:27.5690000	sqlclient	Microsoft SQL Server Management Studio - Query	F00F414A-7D59-4496-B582-9A344C5327EB	5	Standard_S3	0	0

session_id	database_id2	cpu_time	duration	logical_reads	physical_reads	writes	row_count	error_number	severity	state	message	blocked_process_detail	blocking_process_detail
60	NULL	0	8849	4	0	0	0	NULL	NULL	NULL	NULL	NULL	NULL
66	NULL	0	388	2	0	0	1	NULL	NULL	NULL	NULL	NULL	NULL
66	NULL	0	10015147	0	0	0	1	NULL	NULL	NULL	NULL	NULL	NULL
66	NULL	NULL	791	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
77	NULL	0	455	2	0	0	1	NULL	NULL	NULL	NULL	NULL	NULL

EventName	sql_text	SQLStatement	BatchStatement
sql_batch_completed	ALTER EVENT SESSION MS_CSS_test ON DATABASE STATE = START;	NULL	ALTER EVENT SESSION MS_CSS_tes
sql_batch_completed	SELECT @@SPID;	NULL	SELECT @@SPID;
sql_batch_completed	EXEC usp_Wait_30_Seconds;	NULL	EXEC usp_Wait_30_Seconds;
attention	EXEC usp_Wait_30_Seconds;	NULL	NULL
sql_batch_completed	SELECT @@SPID;	NULL	SELECT @@SPID;

Stop the event session

```
-- stop the capture
ALTER EVENT SESSION MS_attention_timeout ON DATABASE STATE = STOP;
```

Clean up the resources

```
-- clean up the resources
ALTER EVENT SESSION MS_attention_timeout ON DATABASE DROP TARGET package0.ring_buffer;

DROP EVENT SESSION MS_attention_timeout ON DATABASE;
```

More information

To reproduce a T-SQL command timeout, you can create a stored procedure that takes more time than the configured query timeout:

```
CREATE OR ALTER PROCEDURE usp_Wait_30_Seconds
AS
   SELECT 1
   WAITFOR DELAY '00:00:30'
;
EXEC usp_Wait_30_Seconds;
```

In SQL Server Management Studio (SSMS), you can set the timeout through Query -> Query Options -> Execution - General. Set the "Execution time-out" value from its default 0 (no timeout) to e.g. 10 seconds. Do not forget to set it back to 0 after your tests.

In your application, change the command timeout parameter with a value less than you have in waitfor (in this case, 5 seconds):

```
using (SqlConnection awConnectionDb = new SqlConnection(connectionStringDb))
{
    awConnectionDb.Open();
    SqlCommand cmd1 = awConnectionDb.CreateCommand();
    cmd1.CommandTimeout = 5;
    cmd1.CommandText = string.Format("usp_Timeout");
    cmd1.ExecuteNonQuery();
}
```

How good have you found this content?