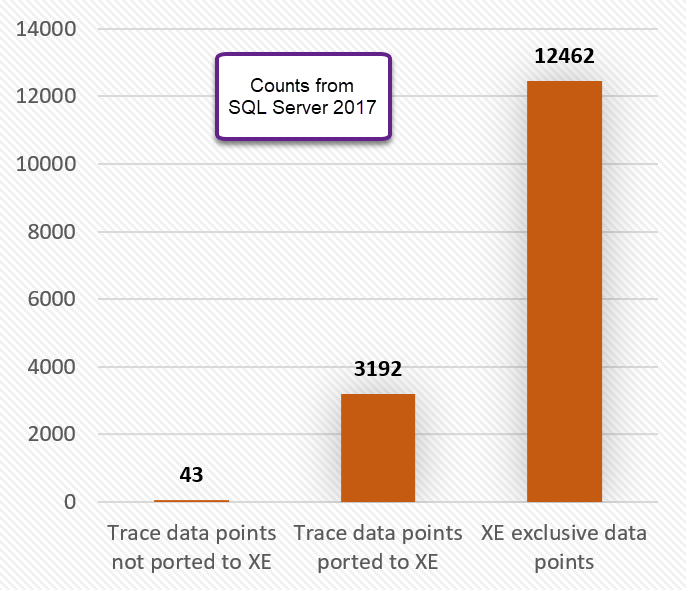
5 Extended Events Sessions Your SQL Server Instance Cannot Live Without

24th October 2017

[Derik Hammer](https://www.sqlhammer.com/author/derik/)

FROM: <https://www.sqlhammer.com/5-extended-events-sessions-sql-server-instance-cannot-live-without/>

[SQL Server Extend Events](https://technet.microsoft.com/en-us/library/bb630354(v=sql.105).aspx) is a general event-handling system. It correlates data from SQL Server and, to a limited degree, the operating system. Extended Events are an optimized replacement for [SQL Trace](https://docs.microsoft.com/en-us/sql/relational-databases/sql-trace/sql-trace). In most cases, Extended Events are much lighter weight and they are always the preferred method for trace events. In addition, there are thousands more Extended Events than SQL Trace events. The list of Extended Events is growing with every major version of SQL Server.

[[](https://www.sqlhammer.com/5-extended-events-sessions-sql-server-instance-cannot-live-without/xe-event-count-comparison/)<img class="size-full wp-image-3858 aligncenter" src="https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/XE-event-count-comparison.png?resize=645%2C555&#038;ssl=1" alt="" width="645" height="555" srcset="https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/XE-event-count-comparison.png?w=686&amp;ssl=1 686w, https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/XE-event-count-comparison.png?resize=300%2C258&amp;ssl=1 300w" sizes="(max-width: 645px) 100vw, 645px" data-recalc-dims="1" />](https://www.sqlhammer.com/5-extended-events-sessions-sql-server-instance-cannot-live-without/xe-event-count-comparison/)

Proactive placement

Extended Events are the all-around smart choice. They take a little bit of time to get used to, however. With thousands of new events and data points, it can be difficult to create an event session in a pinch. That is why it is important to have event sessions pre-scripted or pre-implemented on your SQL Server instances. A little bit of up-front work can save you a lot of time when you need information on the spot. Having them pre-scripted also prevents you from jumping back to [Profiler](https://docs.microsoft.com/en-us/sql/tools/sql-server-profiler/sql-server-profiler), which has a much heavier footprint on your server.

When I create Extended Event sessions, I tend to use the SQL Server Management Studio wizard to find the events and actions (additional fields) that I want. Then, I will script it out and save it for later.

Below are five Extended Events sessions that I have found particularly useful and recommend you add to your toolbox.

Most of these extended event sessions are configured with the ring buffer target because the intention of these sessions is for ad hoc performance analysis. Each could easily be converted to an event file target for historical tracking, alerting, or trend analysis. Read [here](https://msdn.microsoft.com/en-us/library/bb630339.aspx) for more information on targets.

1. Long running queries

Session

This event session tracks long running ad hoc queries and stored procedures. It returns useful fields for identifying what executed the query, such as, the client\_app\_name and client\_hostname. This session uses the [ring buffer](https://technet.microsoft.com/en-us/library/ff878182(v=sql.120).aspx) which is an in-memory target designed for short-term use.

CREATE EVENT SESSION [LongRunningQueries] ON SERVER

ADD EVENT sqlserver.sp\_statement\_completed

(

ACTION

(

package0.collect\_system\_time

,sqlserver.client\_app\_name

,sqlserver.client\_hostname

,sqlserver.database\_name

,sqlserver.plan\_handle

,sqlserver.query\_hash

,sqlserver.session\_id

)

WHERE duration > 5000000 -- longer than 5 seconds

),

ADD EVENT sqlserver.sql\_statement\_completed

(

ACTION

(

package0.collect\_system\_time

,sqlserver.client\_app\_name

,sqlserver.client\_hostname

,sqlserver.database\_name

,sqlserver.plan\_handle

,sqlserver.query\_hash

,sqlserver.session\_id

)

WHERE duration > 5000000 -- longer than 5 seconds

)

ADD TARGET package0.ring\_buffer(SET max\_events\_limit=(0 /\*unlimited\*/),max\_memory=(1048576 /\*1 GB\*/))

WITH (STARTUP\_STATE=OFF,MAX\_DISPATCH\_LATENCY = 5SECONDS)

CREATE EVENT SESSION [LongRunningQueries] ON SERVER

ADD EVENT sqlserver.sp\_statement\_completed

(

ACTION

(

package0.collect\_system\_time

,sqlserver.client\_app\_name

,sqlserver.client\_hostname

,sqlserver.database\_name

,sqlserver.plan\_handle

,sqlserver.query\_hash

,sqlserver.session\_id

)

WHERE duration > 5000000 -- longer than 5 seconds

),

ADD EVENT sqlserver.sql\_statement\_completed

(

ACTION

(

package0.collect\_system\_time

,sqlserver.client\_app\_name

,sqlserver.client\_hostname

,sqlserver.database\_name

,sqlserver.plan\_handle

,sqlserver.query\_hash

,sqlserver.session\_id

)

WHERE duration > 5000000 -- longer than 5 seconds

)

ADD TARGET package0.ring\_buffer(SET max\_events\_limit=(0 /\*unlimited\*/),max\_memory=(1048576 /\*1 GB\*/))

WITH (STARTUP\_STATE=OFF,MAX\_DISPATCH\_LATENCY = 5SECONDS)

CREATE EVENT SESSION [LongRunningQueries] ON SERVER

ADD EVENT sqlserver.sp\_statement\_completed

(

ACTION

(

package0.collect\_system\_time

,sqlserver.client\_app\_name

,sqlserver.client\_hostname

,sqlserver.database\_name

,sqlserver.plan\_handle

,sqlserver.query\_hash

,sqlserver.session\_id

)

WHERE duration > 5000000 -- longer than 5 seconds

),

ADD EVENT sqlserver.sql\_statement\_completed

(

ACTION

(

package0.collect\_system\_time

,sqlserver.client\_app\_name

,sqlserver.client\_hostname

,sqlserver.database\_name

,sqlserver.plan\_handle

,sqlserver.query\_hash

,sqlserver.session\_id

)

WHERE duration > 5000000 -- longer than 5 seconds

)

ADD TARGET package0.ring\_buffer(SET max\_events\_limit=(0 /\*unlimited\*/),max\_memory=(1048576 /\*1 GB\*/))

WITH (STARTUP\_STATE=OFF,MAX\_DISPATCH\_LATENCY = 5SECONDS)

Query

Querying the extended events data requires a little bit of work with XML. Jes Borland describes it well, [here](https://www.brentozar.com/archive/2015/01/query-extended-events-target-xml/). Below is the query written for this session specifically. Commented out, near the bottom, I included a couple of useful filters and sort orders.

--/\* Comment this part out after you run it once, unless you want to refresh the temp table.

IF OBJECT\_ID('tempdb..#capture\_waits\_data') IS NOT NULL

DROP TABLE #capture\_waits\_data

SELECT CAST(target\_data as xml) AS targetdata

INTO #capture\_waits\_data

FROM sys.dm\_xe\_session\_targets xet

JOIN sys.dm\_xe\_sessions xes

ON xes.address = xet.event\_session\_address

WHERE xes.name = 'LongRunningQueries'

AND xet.target\_name = 'ring\_buffer';

--\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SELECT xed.event\_data.value('(@timestamp)[1]', 'datetime2') AS datetime\_utc,

CONVERT(datetime2,SWITCHOFFSET(CONVERT(datetimeoffset,xed.event\_data.value('(@timestamp)[1]', 'datetime2')),DATENAME(TzOffset, SYSDATETIMEOFFSET()))) AS datetime\_local,

xed.event\_data.value('(@name)[1]', 'varchar(50)') AS event\_type,

xed.event\_data.value('(data[@name="statement"]/value)[1]', 'varchar(max)') AS statement,

xed.event\_data.value('(data[@name="duration"]/value)[1]', 'bigint')/1000 AS duration\_ms,

xed.event\_data.value('(data[@name="cpu\_time"]/value)[1]', 'bigint')/1000 AS cpu\_time\_ms,

xed.event\_data.value('(data[@name="physical\_reads"]/value)[1]', 'bigint') AS physical\_reads,

xed.event\_data.value('(data[@name="logical\_reads"]/value)[1]', 'bigint') AS logical\_reads,

xed.event\_data.value('(data[@name="writes"]/value)[1]', 'bigint') AS writes,

xed.event\_data.value('(data[@name="row\_count"]/value)[1]', 'bigint') AS row\_count,

xed.event\_data.value('(action[@name="database\_name"]/value)[1]', 'varchar(255)') AS database\_name,

xed.event\_data.value('(action[@name="client\_hostname"]/value)[1]', 'varchar(255)') AS client\_hostname,

xed.event\_data.value('(action[@name="client\_app\_name"]/value)[1]', 'varchar(255)') AS client\_app\_name

FROM #capture\_waits\_data

CROSS APPLY targetdata.nodes('//RingBufferTarget/event') AS xed (event\_data)

WHERE 1=1

/\* refine your search further than the XE session's filter

AND xed.event\_data.value('(data[@name="statement"]/value)[1]', 'varchar(max)') = 'EXEC spDemoSproc'

--\*/

/\* find queries within a time range

AND xed.event\_data.value('(@timestamp)[1]', 'datetime2') > CAST('20170925 09:57 AM' AS datetime2) AT TIME ZONE 'Eastern Standard Time'

--\*/

/\* Find highest resource consumption

ORDER BY

xed.event\_data.value('(data[@name="duration"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="cpu\_time"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="physical\_reads"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="logical\_reads"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="writes"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="row\_count"]/value)[1]', 'bigint') DESC

--\*/

--/\* Comment this part out after you run it once, unless you want to refresh the temp table.

IF OBJECT\_ID('tempdb..#capture\_waits\_data') IS NOT NULL

DROP TABLE #capture\_waits\_data

SELECT CAST(target\_data as xml) AS targetdata

INTO #capture\_waits\_data

FROM sys.dm\_xe\_session\_targets xet

JOIN sys.dm\_xe\_sessions xes

ON xes.address = xet.event\_session\_address

WHERE xes.name = 'LongRunningQueries'

AND xet.target\_name = 'ring\_buffer';

--\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SELECT xed.event\_data.value('(@timestamp)[1]', 'datetime2') AS datetime\_utc,

CONVERT(datetime2,SWITCHOFFSET(CONVERT(datetimeoffset,xed.event\_data.value('(@timestamp)[1]', 'datetime2')),DATENAME(TzOffset, SYSDATETIMEOFFSET()))) AS datetime\_local,

xed.event\_data.value('(@name)[1]', 'varchar(50)') AS event\_type,

xed.event\_data.value('(data[@name="statement"]/value)[1]', 'varchar(max)') AS statement,

xed.event\_data.value('(data[@name="duration"]/value)[1]', 'bigint')/1000 AS duration\_ms,

xed.event\_data.value('(data[@name="cpu\_time"]/value)[1]', 'bigint')/1000 AS cpu\_time\_ms,

xed.event\_data.value('(data[@name="physical\_reads"]/value)[1]', 'bigint') AS physical\_reads,

xed.event\_data.value('(data[@name="logical\_reads"]/value)[1]', 'bigint') AS logical\_reads,

xed.event\_data.value('(data[@name="writes"]/value)[1]', 'bigint') AS writes,

xed.event\_data.value('(data[@name="row\_count"]/value)[1]', 'bigint') AS row\_count,

xed.event\_data.value('(action[@name="database\_name"]/value)[1]', 'varchar(255)') AS database\_name,

xed.event\_data.value('(action[@name="client\_hostname"]/value)[1]', 'varchar(255)') AS client\_hostname,

xed.event\_data.value('(action[@name="client\_app\_name"]/value)[1]', 'varchar(255)') AS client\_app\_name

FROM #capture\_waits\_data

CROSS APPLY targetdata.nodes('//RingBufferTarget/event') AS xed (event\_data)

WHERE 1=1

/\* refine your search further than the XE session's filter

AND xed.event\_data.value('(data[@name="statement"]/value)[1]', 'varchar(max)') = 'EXEC spDemoSproc'

--\*/

/\* find queries within a time range

AND xed.event\_data.value('(@timestamp)[1]', 'datetime2') > CAST('20170925 09:57 AM' AS datetime2) AT TIME ZONE 'Eastern Standard Time'

--\*/

/\* Find highest resource consumption

ORDER BY

xed.event\_data.value('(data[@name="duration"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="cpu\_time"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="physical\_reads"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="logical\_reads"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="writes"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="row\_count"]/value)[1]', 'bigint') DESC

--\*/

--/\* Comment this part out after you run it once, unless you want to refresh the temp table.

IF OBJECT\_ID('tempdb..#capture\_waits\_data') IS NOT NULL

DROP TABLE #capture\_waits\_data

SELECT CAST(target\_data as xml) AS targetdata

INTO #capture\_waits\_data

FROM sys.dm\_xe\_session\_targets xet

JOIN sys.dm\_xe\_sessions xes

ON xes.address = xet.event\_session\_address

WHERE xes.name = 'LongRunningQueries'

AND xet.target\_name = 'ring\_buffer';

--\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SELECT xed.event\_data.value('(@timestamp)[1]', 'datetime2') AS datetime\_utc,

CONVERT(datetime2,SWITCHOFFSET(CONVERT(datetimeoffset,xed.event\_data.value('(@timestamp)[1]', 'datetime2')),DATENAME(TzOffset, SYSDATETIMEOFFSET()))) AS datetime\_local,

xed.event\_data.value('(@name)[1]', 'varchar(50)') AS event\_type,

xed.event\_data.value('(data[@name="statement"]/value)[1]', 'varchar(max)') AS statement,

xed.event\_data.value('(data[@name="duration"]/value)[1]', 'bigint')/1000 AS duration\_ms,

xed.event\_data.value('(data[@name="cpu\_time"]/value)[1]', 'bigint')/1000 AS cpu\_time\_ms,

xed.event\_data.value('(data[@name="physical\_reads"]/value)[1]', 'bigint') AS physical\_reads,

xed.event\_data.value('(data[@name="logical\_reads"]/value)[1]', 'bigint') AS logical\_reads,

xed.event\_data.value('(data[@name="writes"]/value)[1]', 'bigint') AS writes,

xed.event\_data.value('(data[@name="row\_count"]/value)[1]', 'bigint') AS row\_count,

xed.event\_data.value('(action[@name="database\_name"]/value)[1]', 'varchar(255)') AS database\_name,

xed.event\_data.value('(action[@name="client\_hostname"]/value)[1]', 'varchar(255)') AS client\_hostname,

xed.event\_data.value('(action[@name="client\_app\_name"]/value)[1]', 'varchar(255)') AS client\_app\_name

FROM #capture\_waits\_data

CROSS APPLY targetdata.nodes('//RingBufferTarget/event') AS xed (event\_data)

WHERE 1=1

/\* refine your search further than the XE session's filter

AND xed.event\_data.value('(data[@name="statement"]/value)[1]', 'varchar(max)') = 'EXEC spDemoSproc'

--\*/

/\* find queries within a time range

AND xed.event\_data.value('(@timestamp)[1]', 'datetime2') > CAST('20170925 09:57 AM' AS datetime2) AT TIME ZONE 'Eastern Standard Time'

--\*/

/\* Find highest resource consumption

ORDER BY

xed.event\_data.value('(data[@name="duration"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="cpu\_time"]/value)[1]', 'bigint') DESC

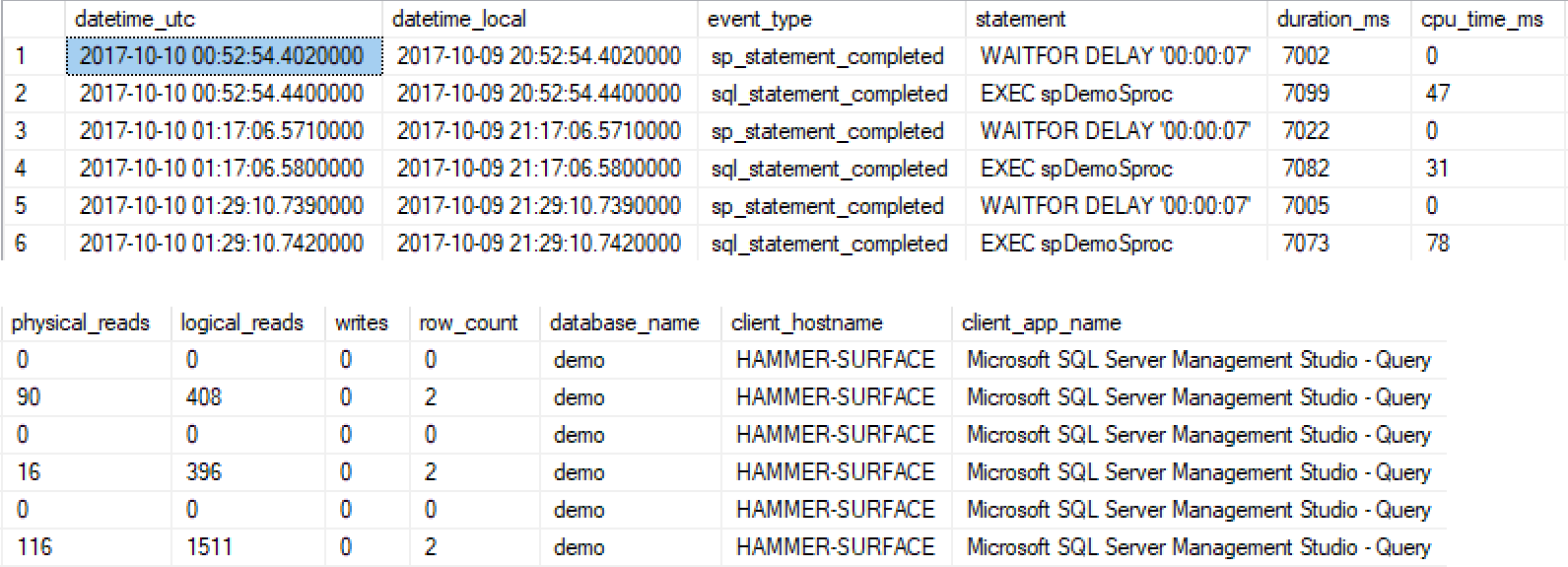
--xed.event\_data.value('(data[@name="physical\_reads"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="logical\_reads"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="writes"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="row\_count"]/value)[1]', 'bigint') DESC

--\*/

[[](https://www.sqlhammer.com/5-extended-events-sessions-sql-server-instance-cannot-live-without/long-running-query/)<img class="alignnone size-full wp-image-3859" src="https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/long-running-query.png?resize=645%2C233&#038;ssl=1" alt="" width="645" height="233" srcset="https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/long-running-query.png?w=1591&amp;ssl=1 1591w, https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/long-running-query.png?resize=300%2C108&amp;ssl=1 300w, https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/long-running-query.png?resize=768%2C278&amp;ssl=1 768w, https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/long-running-query.png?resize=1024%2C370&amp;ssl=1 1024w, https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/long-running-query.png?w=1290&amp;ssl=1 1290w" sizes="(max-width: 645px) 100vw, 645px" data-recalc-dims="1" />](https://www.sqlhammer.com/5-extended-events-sessions-sql-server-instance-cannot-live-without/long-running-query/)

2. Query timeouts

Session

This next query is designed to track connections which are interrupted or aborted, often caused by execution timeouts. Unlike the last session, this one uses the [pair matching](https://msdn.microsoft.com/en-us/library/ff878062(v=sql.120).aspx) target. The pair matching target is also an in-memory target but correlates matched pairs and reports on unmatched events. This works by detecting a sqlserver.sql\_statement\_started event without a matching sqlserver.sql\_statement\_completed event.

CREATE EVENT SESSION [QueryTimeouts] ON SERVER

ADD EVENT sqlserver.sql\_statement\_completed

(

ACTION

(

sqlserver.session\_id

,sqlserver.query\_hash

,sqlserver.tsql\_stack

)

),

ADD EVENT sqlserver.sql\_statement\_starting

(

ACTION

(

sqlserver.session\_id

,sqlserver.query\_hash

,sqlserver.tsql\_stack

)

)

ADD TARGET package0.pair\_matching

(

SET

begin\_event = 'sqlserver.sql\_statement\_starting',

begin\_matching\_actions = 'sqlserver.session\_id, sqlserver.tsql\_stack',

end\_event = 'sqlserver.sql\_statement\_completed',

end\_matching\_actions = 'sqlserver.session\_id, sqlserver.tsql\_stack',

respond\_to\_memory\_pressure = 0

)

WITH (MAX\_DISPATCH\_LATENCY=5 SECONDS, TRACK\_CAUSALITY=ON, STARTUP\_STATE=OFF)

CREATE EVENT SESSION [QueryTimeouts] ON SERVER

ADD EVENT sqlserver.sql\_statement\_completed

(

ACTION

(

sqlserver.session\_id

,sqlserver.query\_hash

,sqlserver.tsql\_stack

)

),

ADD EVENT sqlserver.sql\_statement\_starting

(

ACTION

(

sqlserver.session\_id

,sqlserver.query\_hash

,sqlserver.tsql\_stack

)

)

ADD TARGET package0.pair\_matching

(

SET

begin\_event = 'sqlserver.sql\_statement\_starting',

begin\_matching\_actions = 'sqlserver.session\_id, sqlserver.tsql\_stack',

end\_event = 'sqlserver.sql\_statement\_completed',

end\_matching\_actions = 'sqlserver.session\_id, sqlserver.tsql\_stack',

respond\_to\_memory\_pressure = 0

)

WITH (MAX\_DISPATCH\_LATENCY=5 SECONDS, TRACK\_CAUSALITY=ON, STARTUP\_STATE=OFF)

CREATE EVENT SESSION [QueryTimeouts] ON SERVER

ADD EVENT sqlserver.sql\_statement\_completed

(

ACTION

(

sqlserver.session\_id

,sqlserver.query\_hash

,sqlserver.tsql\_stack

)

),

ADD EVENT sqlserver.sql\_statement\_starting

(

ACTION

(

sqlserver.session\_id

,sqlserver.query\_hash

,sqlserver.tsql\_stack

)

)

ADD TARGET package0.pair\_matching

(

SET

begin\_event = 'sqlserver.sql\_statement\_starting',

begin\_matching\_actions = 'sqlserver.session\_id, sqlserver.tsql\_stack',

end\_event = 'sqlserver.sql\_statement\_completed',

end\_matching\_actions = 'sqlserver.session\_id, sqlserver.tsql\_stack',

respond\_to\_memory\_pressure = 0

)

WITH (MAX\_DISPATCH\_LATENCY=5 SECONDS, TRACK\_CAUSALITY=ON, STARTUP\_STATE=OFF)

Query

The below query aggregates the matched count which were successfully completed queries, the unmatched counts which represent the timeouts, and then it will display the detailed information for the queries that timed out.

-- Create XML variable to hold Target Data

DECLARE @target\_data XML

SELECT @target\_data =

CAST(target\_data AS XML)

FROM sys.dm\_xe\_sessions AS s

JOIN sys.dm\_xe\_session\_targets AS t

ON t.event\_session\_address = s.address

WHERE s.name = 'QueryTimeouts'

AND t.target\_name = 'pair\_matching'

-- Query XML variable to get Target Execution information

SELECT

@target\_data.value('(PairingTarget/@orphanCount)[1]', 'int') AS orphanCount,

@target\_data.value('(PairingTarget/@matchedCount)[1]', 'int') AS matchedCount,

COALESCE(@target\_data.value('(PairingTarget/@memoryPressureDroppedCount)[1]', 'int'),0) AS memoryPressureDroppedCount

-- Query the XML variable to get the Target Data

SELECT

n.value('(event/action[@name="session\_id"]/value)[1]', 'int') as session\_id,

n.value('(event/@name)[1]', 'varchar(50)') AS event\_name,

n.value('(event/data[@name="statement"]/value)[1]', 'varchar(8000)') as statement,

NULLIF(n.value('(event/action[@name="query\_hash"]/value)[1]', 'numeric(20)'),0) as query\_hash\_numeric,

n.value('(event/@timestamp)[1]', 'datetime2') AS datetime\_utc,

DATEADD(hh,DATEDIFF(hh, GETUTCDATE(), CURRENT\_TIMESTAMP),n.value('(event/@timestamp)[1]', 'datetime2')) AS datetime\_local,

n.value('(event/action[@name="tsql\_stack"]/text)[1]', 'varchar(max)') as tsql\_stack

FROM

(

SELECT td.query('.') as n

FROM @target\_data.nodes('PairingTarget/event') AS q(td)

) as tab

--Excluding this currently running query.

WHERE n.value('(event/action[@name="session\_id"]/value)[1]', 'int') <> @@SPID

ORDER BY session\_id

-- Create XML variable to hold Target Data

DECLARE @target\_data XML

SELECT @target\_data =

CAST(target\_data AS XML)

FROM sys.dm\_xe\_sessions AS s

JOIN sys.dm\_xe\_session\_targets AS t

ON t.event\_session\_address = s.address

WHERE s.name = 'QueryTimeouts'

AND t.target\_name = 'pair\_matching'

-- Query XML variable to get Target Execution information

SELECT

@target\_data.value('(PairingTarget/@orphanCount)[1]', 'int') AS orphanCount,

@target\_data.value('(PairingTarget/@matchedCount)[1]', 'int') AS matchedCount,

COALESCE(@target\_data.value('(PairingTarget/@memoryPressureDroppedCount)[1]', 'int'),0) AS memoryPressureDroppedCount

-- Query the XML variable to get the Target Data

SELECT

n.value('(event/action[@name="session\_id"]/value)[1]', 'int') as session\_id,

n.value('(event/@name)[1]', 'varchar(50)') AS event\_name,

n.value('(event/data[@name="statement"]/value)[1]', 'varchar(8000)') as statement,

NULLIF(n.value('(event/action[@name="query\_hash"]/value)[1]', 'numeric(20)'),0) as query\_hash\_numeric,

n.value('(event/@timestamp)[1]', 'datetime2') AS datetime\_utc,

DATEADD(hh,DATEDIFF(hh, GETUTCDATE(), CURRENT\_TIMESTAMP),n.value('(event/@timestamp)[1]', 'datetime2')) AS datetime\_local,

n.value('(event/action[@name="tsql\_stack"]/text)[1]', 'varchar(max)') as tsql\_stack

FROM

(

SELECT td.query('.') as n

FROM @target\_data.nodes('PairingTarget/event') AS q(td)

) as tab

--Excluding this currently running query.

WHERE n.value('(event/action[@name="session\_id"]/value)[1]', 'int') <> @@SPID

ORDER BY session\_id

-- Create XML variable to hold Target Data

DECLARE @target\_data XML

SELECT @target\_data =

CAST(target\_data AS XML)

FROM sys.dm\_xe\_sessions AS s

JOIN sys.dm\_xe\_session\_targets AS t

ON t.event\_session\_address = s.address

WHERE s.name = 'QueryTimeouts'

AND t.target\_name = 'pair\_matching'

-- Query XML variable to get Target Execution information

SELECT

@target\_data.value('(PairingTarget/@orphanCount)[1]', 'int') AS orphanCount,

@target\_data.value('(PairingTarget/@matchedCount)[1]', 'int') AS matchedCount,

COALESCE(@target\_data.value('(PairingTarget/@memoryPressureDroppedCount)[1]', 'int'),0) AS memoryPressureDroppedCount

-- Query the XML variable to get the Target Data

SELECT

n.value('(event/action[@name="session\_id"]/value)[1]', 'int') as session\_id,

n.value('(event/@name)[1]', 'varchar(50)') AS event\_name,

n.value('(event/data[@name="statement"]/value)[1]', 'varchar(8000)') as statement,

NULLIF(n.value('(event/action[@name="query\_hash"]/value)[1]', 'numeric(20)'),0) as query\_hash\_numeric,

n.value('(event/@timestamp)[1]', 'datetime2') AS datetime\_utc,

DATEADD(hh,DATEDIFF(hh, GETUTCDATE(), CURRENT\_TIMESTAMP),n.value('(event/@timestamp)[1]', 'datetime2')) AS datetime\_local,

n.value('(event/action[@name="tsql\_stack"]/text)[1]', 'varchar(max)') as tsql\_stack

FROM

(

SELECT td.query('.') as n

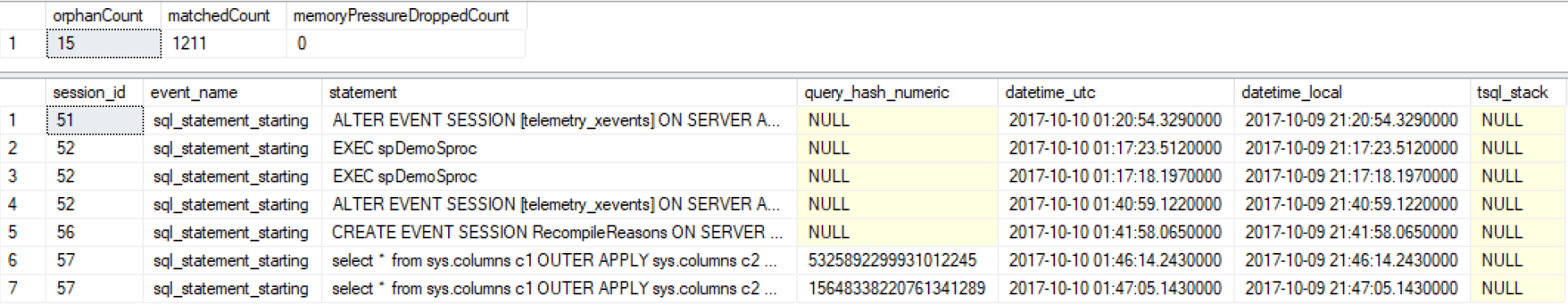
FROM @target\_data.nodes('PairingTarget/event') AS q(td)

) as tab

--Excluding this currently running query.

WHERE n.value('(event/action[@name="session\_id"]/value)[1]', 'int') <> @@SPID

ORDER BY session\_id

[[](https://www.sqlhammer.com/5-extended-events-sessions-sql-server-instance-cannot-live-without/timeouts-query/)<img class="alignnone size-full wp-image-3860" src="https://i2.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/timeouts-query.png?resize=645%2C125&#038;ssl=1" alt="" width="645" height="125" srcset="https://i2.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/timeouts-query.png?w=2124&amp;ssl=1 2124w, https://i2.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/timeouts-query.png?resize=300%2C58&amp;ssl=1 300w, https://i2.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/timeouts-query.png?resize=768%2C149&amp;ssl=1 768w, https://i2.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/timeouts-query.png?resize=1024%2C199&amp;ssl=1 1024w, https://i2.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/timeouts-query.png?w=1290&amp;ssl=1 1290w, https://i2.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/timeouts-query.png?w=1935&amp;ssl=1 1935w" sizes="(max-width: 645px) 100vw, 645px" data-recalc-dims="1" />](https://www.sqlhammer.com/5-extended-events-sessions-sql-server-instance-cannot-live-without/timeouts-query/)

3. One bad query

Session

This session is used to track a specific query or stored procedure. It is great for capturing actual runtime data about a query that you know is a problem. It implements a wild card search on the sql\_text by populating the @TextSearch variable with a snippet from your query.

DECLARE @TextSearch NVARCHAR(3998) = N'query text to search'

EXEC(N'

CREATE EVENT SESSION [Query\_' + @TextSearch + '] ON SERVER

ADD EVENT sqlserver.sp\_statement\_completed

(

ACTION

(

package0.collect\_system\_time,

sqlserver.client\_app\_name,

sqlserver.client\_hostname,

sqlserver.database\_name

)

WHERE ([sqlserver].[like\_i\_sql\_unicode\_string]([sqlserver].[sql\_text],N''%' + @TextSearch + '%''))

),

ADD EVENT sqlserver.sql\_statement\_completed

(

ACTION

(

package0.collect\_system\_time,

sqlserver.client\_app\_name,

sqlserver.client\_hostname,

sqlserver.database\_name

)

WHERE ([sqlserver].[like\_i\_sql\_unicode\_string]([sqlserver].[sql\_text],N''%' + @TextSearch + '%''))

)

ADD TARGET package0.ring\_buffer(SET max\_events\_limit=(0 /\*unlimited\*/),max\_memory=(1048576 /\*1 GB\*/))

WITH (STARTUP\_STATE=OFF,MAX\_DISPATCH\_LATENCY = 5SECONDS)

')

DECLARE @TextSearch NVARCHAR(3998) = N'query text to search'

EXEC(N'

CREATE EVENT SESSION [Query\_' + @TextSearch + '] ON SERVER

ADD EVENT sqlserver.sp\_statement\_completed

(

ACTION

(

package0.collect\_system\_time,

sqlserver.client\_app\_name,

sqlserver.client\_hostname,

sqlserver.database\_name

)

WHERE ([sqlserver].[like\_i\_sql\_unicode\_string]([sqlserver].[sql\_text],N''%' + @TextSearch + '%''))

),

ADD EVENT sqlserver.sql\_statement\_completed

(

ACTION

(

package0.collect\_system\_time,

sqlserver.client\_app\_name,

sqlserver.client\_hostname,

sqlserver.database\_name

)

WHERE ([sqlserver].[like\_i\_sql\_unicode\_string]([sqlserver].[sql\_text],N''%' + @TextSearch + '%''))

)

ADD TARGET package0.ring\_buffer(SET max\_events\_limit=(0 /\*unlimited\*/),max\_memory=(1048576 /\*1 GB\*/))

WITH (STARTUP\_STATE=OFF,MAX\_DISPATCH\_LATENCY = 5SECONDS)

')

DECLARE @TextSearch NVARCHAR(3998) = N'query text to search'

EXEC(N'

CREATE EVENT SESSION [Query\_' + @TextSearch + '] ON SERVER

ADD EVENT sqlserver.sp\_statement\_completed

(

ACTION

(

package0.collect\_system\_time,

sqlserver.client\_app\_name,

sqlserver.client\_hostname,

sqlserver.database\_name

)

WHERE ([sqlserver].[like\_i\_sql\_unicode\_string]([sqlserver].[sql\_text],N''%' + @TextSearch + '%''))

),

ADD EVENT sqlserver.sql\_statement\_completed

(

ACTION

(

package0.collect\_system\_time,

sqlserver.client\_app\_name,

sqlserver.client\_hostname,

sqlserver.database\_name

)

WHERE ([sqlserver].[like\_i\_sql\_unicode\_string]([sqlserver].[sql\_text],N''%' + @TextSearch + '%''))

)

ADD TARGET package0.ring\_buffer(SET max\_events\_limit=(0 /\*unlimited\*/),max\_memory=(1048576 /\*1 GB\*/))

WITH (STARTUP\_STATE=OFF,MAX\_DISPATCH\_LATENCY = 5SECONDS)

')

Query

--/\* Comment this part out after you run it once, unless you want to refresh the temp table.

DECLARE @SessionName NVARCHAR(4000) = 'Query\_demo'

IF OBJECT\_ID('tempdb..#capture\_waits\_data') IS NOT NULL

DROP TABLE #capture\_waits\_data

SELECT CAST(target\_data as xml) AS targetdata

INTO #capture\_waits\_data

FROM sys.dm\_xe\_session\_targets xet

JOIN sys.dm\_xe\_sessions xes

ON xes.address = xet.event\_session\_address

WHERE xes.name = @SessionName

AND xet.target\_name = 'ring\_buffer';

--\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SELECT xed.event\_data.value('(@timestamp)[1]', 'datetime2') AS datetime\_utc,

DATEADD(hh,DATEDIFF(hh, GETUTCDATE(), CURRENT\_TIMESTAMP),xed.event\_data.value('(@timestamp)[1]', 'datetime2')) AS datetime\_local,

xed.event\_data.value('(@name)[1]', 'varchar(50)') AS event\_type,

xed.event\_data.value('(data[@name="statement"]/value)[1]', 'varchar(max)') AS statement,

xed.event\_data.value('(data[@name="duration"]/value)[1]', 'bigint')/1000 AS duration\_ms,

xed.event\_data.value('(data[@name="cpu\_time"]/value)[1]', 'bigint')/1000 AS cpu\_time\_ms,

xed.event\_data.value('(data[@name="physical\_reads"]/value)[1]', 'bigint') AS physical\_reads,

xed.event\_data.value('(data[@name="logical\_reads"]/value)[1]', 'bigint') AS logical\_reads,

xed.event\_data.value('(data[@name="writes"]/value)[1]', 'bigint') AS writes,

xed.event\_data.value('(data[@name="row\_count"]/value)[1]', 'bigint') AS row\_count,

xed.event\_data.value('(action[@name="database\_name"]/value)[1]', 'varchar(255)') AS database\_name,

xed.event\_data.value('(action[@name="client\_hostname"]/value)[1]', 'varchar(255)') AS client\_hostname,

xed.event\_data.value('(action[@name="client\_app\_name"]/value)[1]', 'varchar(255)') AS client\_app\_name

FROM #capture\_waits\_data

CROSS APPLY targetdata.nodes('//RingBufferTarget/event') AS xed (event\_data)

WHERE 1=1

/\* refine your search further than the XE session's filter

AND xed.event\_data.value('(data[@name="statement"]/value)[1]', 'varchar(max)') = 'EXEC spDemoSproc'

--\*/

/\* find queries within a time range

AND xed.event\_data.value('(@timestamp)[1]', 'datetime2') > CAST('20170925 09:57 AM' AS datetime2) AT TIME ZONE 'Eastern Standard Time'

--\*/

/\* Find highest resource consumption

ORDER BY

xed.event\_data.value('(data[@name="duration"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="cpu\_time"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="physical\_reads"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="logical\_reads"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="writes"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="row\_count"]/value)[1]', 'bigint') DESC

--\*/

--/\* Comment this part out after you run it once, unless you want to refresh the temp table.

DECLARE @SessionName NVARCHAR(4000) = 'Query\_demo'

IF OBJECT\_ID('tempdb..#capture\_waits\_data') IS NOT NULL

DROP TABLE #capture\_waits\_data

SELECT CAST(target\_data as xml) AS targetdata

INTO #capture\_waits\_data

FROM sys.dm\_xe\_session\_targets xet

JOIN sys.dm\_xe\_sessions xes

ON xes.address = xet.event\_session\_address

WHERE xes.name = @SessionName

AND xet.target\_name = 'ring\_buffer';

--\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SELECT xed.event\_data.value('(@timestamp)[1]', 'datetime2') AS datetime\_utc,

DATEADD(hh,DATEDIFF(hh, GETUTCDATE(), CURRENT\_TIMESTAMP),xed.event\_data.value('(@timestamp)[1]', 'datetime2')) AS datetime\_local,

xed.event\_data.value('(@name)[1]', 'varchar(50)') AS event\_type,

xed.event\_data.value('(data[@name="statement"]/value)[1]', 'varchar(max)') AS statement,

xed.event\_data.value('(data[@name="duration"]/value)[1]', 'bigint')/1000 AS duration\_ms,

xed.event\_data.value('(data[@name="cpu\_time"]/value)[1]', 'bigint')/1000 AS cpu\_time\_ms,

xed.event\_data.value('(data[@name="physical\_reads"]/value)[1]', 'bigint') AS physical\_reads,

xed.event\_data.value('(data[@name="logical\_reads"]/value)[1]', 'bigint') AS logical\_reads,

xed.event\_data.value('(data[@name="writes"]/value)[1]', 'bigint') AS writes,

xed.event\_data.value('(data[@name="row\_count"]/value)[1]', 'bigint') AS row\_count,

xed.event\_data.value('(action[@name="database\_name"]/value)[1]', 'varchar(255)') AS database\_name,

xed.event\_data.value('(action[@name="client\_hostname"]/value)[1]', 'varchar(255)') AS client\_hostname,

xed.event\_data.value('(action[@name="client\_app\_name"]/value)[1]', 'varchar(255)') AS client\_app\_name

FROM #capture\_waits\_data

CROSS APPLY targetdata.nodes('//RingBufferTarget/event') AS xed (event\_data)

WHERE 1=1

/\* refine your search further than the XE session's filter

AND xed.event\_data.value('(data[@name="statement"]/value)[1]', 'varchar(max)') = 'EXEC spDemoSproc'

--\*/

/\* find queries within a time range

AND xed.event\_data.value('(@timestamp)[1]', 'datetime2') > CAST('20170925 09:57 AM' AS datetime2) AT TIME ZONE 'Eastern Standard Time'

--\*/

/\* Find highest resource consumption

ORDER BY

xed.event\_data.value('(data[@name="duration"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="cpu\_time"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="physical\_reads"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="logical\_reads"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="writes"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="row\_count"]/value)[1]', 'bigint') DESC

--\*/

--/\* Comment this part out after you run it once, unless you want to refresh the temp table.

DECLARE @SessionName NVARCHAR(4000) = 'Query\_demo'

IF OBJECT\_ID('tempdb..#capture\_waits\_data') IS NOT NULL

DROP TABLE #capture\_waits\_data

SELECT CAST(target\_data as xml) AS targetdata

INTO #capture\_waits\_data

FROM sys.dm\_xe\_session\_targets xet

JOIN sys.dm\_xe\_sessions xes

ON xes.address = xet.event\_session\_address

WHERE xes.name = @SessionName

AND xet.target\_name = 'ring\_buffer';

--\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SELECT xed.event\_data.value('(@timestamp)[1]', 'datetime2') AS datetime\_utc,

DATEADD(hh,DATEDIFF(hh, GETUTCDATE(), CURRENT\_TIMESTAMP),xed.event\_data.value('(@timestamp)[1]', 'datetime2')) AS datetime\_local,

xed.event\_data.value('(@name)[1]', 'varchar(50)') AS event\_type,

xed.event\_data.value('(data[@name="statement"]/value)[1]', 'varchar(max)') AS statement,

xed.event\_data.value('(data[@name="duration"]/value)[1]', 'bigint')/1000 AS duration\_ms,

xed.event\_data.value('(data[@name="cpu\_time"]/value)[1]', 'bigint')/1000 AS cpu\_time\_ms,

xed.event\_data.value('(data[@name="physical\_reads"]/value)[1]', 'bigint') AS physical\_reads,

xed.event\_data.value('(data[@name="logical\_reads"]/value)[1]', 'bigint') AS logical\_reads,

xed.event\_data.value('(data[@name="writes"]/value)[1]', 'bigint') AS writes,

xed.event\_data.value('(data[@name="row\_count"]/value)[1]', 'bigint') AS row\_count,

xed.event\_data.value('(action[@name="database\_name"]/value)[1]', 'varchar(255)') AS database\_name,

xed.event\_data.value('(action[@name="client\_hostname"]/value)[1]', 'varchar(255)') AS client\_hostname,

xed.event\_data.value('(action[@name="client\_app\_name"]/value)[1]', 'varchar(255)') AS client\_app\_name

FROM #capture\_waits\_data

CROSS APPLY targetdata.nodes('//RingBufferTarget/event') AS xed (event\_data)

WHERE 1=1

/\* refine your search further than the XE session's filter

AND xed.event\_data.value('(data[@name="statement"]/value)[1]', 'varchar(max)') = 'EXEC spDemoSproc'

--\*/

/\* find queries within a time range

AND xed.event\_data.value('(@timestamp)[1]', 'datetime2') > CAST('20170925 09:57 AM' AS datetime2) AT TIME ZONE 'Eastern Standard Time'

--\*/

/\* Find highest resource consumption

ORDER BY

xed.event\_data.value('(data[@name="duration"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="cpu\_time"]/value)[1]', 'bigint') DESC

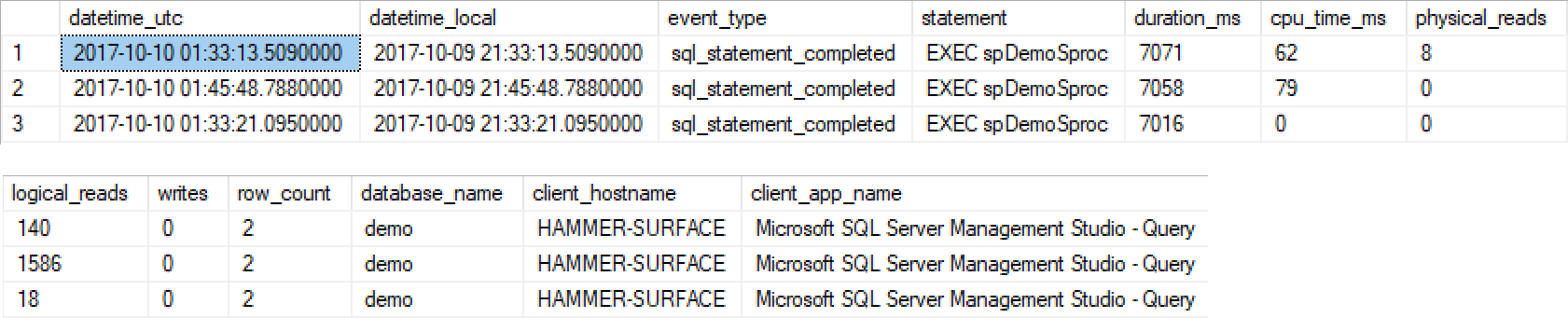
--xed.event\_data.value('(data[@name="physical\_reads"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="logical\_reads"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="writes"]/value)[1]', 'bigint') DESC

--xed.event\_data.value('(data[@name="row\_count"]/value)[1]', 'bigint') DESC

--\*/

[[](https://www.sqlhammer.com/5-extended-events-sessions-sql-server-instance-cannot-live-without/bad-query-query/)<img class="alignnone size-full wp-image-3861" src="https://i0.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/bad-query-query.png?resize=645%2C131&#038;ssl=1" alt="" width="645" height="131" srcset="https://i0.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/bad-query-query.png?w=1681&amp;ssl=1 1681w, https://i0.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/bad-query-query.png?resize=300%2C61&amp;ssl=1 300w, https://i0.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/bad-query-query.png?resize=768%2C156&amp;ssl=1 768w, https://i0.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/bad-query-query.png?resize=1024%2C208&amp;ssl=1 1024w, https://i0.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/bad-query-query.png?w=1290&amp;ssl=1 1290w" sizes="(max-width: 645px) 100vw, 645px" data-recalc-dims="1" />](https://www.sqlhammer.com/5-extended-events-sessions-sql-server-instance-cannot-live-without/bad-query-query/)

4. Why are there excessive recompilations?

Session

This session is useful for detecting excessive recompilations which cause queries to slow and CPU usage to spike. It uses the [histogram](https://msdn.microsoft.com/en-us/library/ff878023.aspx) target to capture the reasons for recompiles and how many of each reason occurred.

CREATE EVENT SESSION RecompileReasons ON SERVER

ADD EVENT sqlserver.sql\_statement\_recompile

ADD TARGET package0.histogram

(

SET filtering\_event\_name=N'sqlserver.sql\_statement\_recompile',

source=N'recompile\_cause',

source\_type=(0)

)

WITH (STARTUP\_STATE=OFF, MAX\_DISPATCH\_LATENCY = 5SECONDS)

CREATE EVENT SESSION RecompileReasons ON SERVER

ADD EVENT sqlserver.sql\_statement\_recompile

ADD TARGET package0.histogram

(

SET filtering\_event\_name=N'sqlserver.sql\_statement\_recompile',

source=N'recompile\_cause',

source\_type=(0)

)

WITH (STARTUP\_STATE=OFF, MAX\_DISPATCH\_LATENCY = 5SECONDS)

CREATE EVENT SESSION RecompileReasons ON SERVER

ADD EVENT sqlserver.sql\_statement\_recompile

ADD TARGET package0.histogram

(

SET filtering\_event\_name=N'sqlserver.sql\_statement\_recompile',

source=N'recompile\_cause',

source\_type=(0)

)

WITH (STARTUP\_STATE=OFF, MAX\_DISPATCH\_LATENCY = 5SECONDS)

Query

SELECT sv.subclass\_name as recompile\_cause,

shredded.recompile\_count

FROM sys.dm\_xe\_session\_targets AS xet

JOIN sys.dm\_xe\_sessions AS xes ON (xe.address = xet.event\_session\_address)

CROSS APPLY ( SELECT CAST(xet.target\_data as xml) ) as target\_data\_xml ([xml])

CROSS APPLY target\_data\_xml.[xml].nodes('/HistogramTarget/Slot') AS nodes (slot\_data)

CROSS APPLY

(

SELECT nodes.slot\_data.value('(value)[1]', 'int') AS recompile\_cause,

nodes.slot\_data.value('(@count)[1]', 'int') AS recompile\_count

) as shredded

JOIN sys.trace\_subclass\_values AS sv ON shredded.recompile\_cause = sv.subclass\_value

WHERE xes.name = 'RecompileReasons'

AND sv.trace\_event\_id = 37 -- SP:Recompile

SELECT sv.subclass\_name as recompile\_cause,

shredded.recompile\_count

FROM sys.dm\_xe\_session\_targets AS xet

JOIN sys.dm\_xe\_sessions AS xes ON (xe.address = xet.event\_session\_address)

CROSS APPLY ( SELECT CAST(xet.target\_data as xml) ) as target\_data\_xml ([xml])

CROSS APPLY target\_data\_xml.[xml].nodes('/HistogramTarget/Slot') AS nodes (slot\_data)

CROSS APPLY

(

SELECT nodes.slot\_data.value('(value)[1]', 'int') AS recompile\_cause,

nodes.slot\_data.value('(@count)[1]', 'int') AS recompile\_count

) as shredded

JOIN sys.trace\_subclass\_values AS sv ON shredded.recompile\_cause = sv.subclass\_value

WHERE xes.name = 'RecompileReasons'

AND sv.trace\_event\_id = 37 -- SP:Recompile

SELECT sv.subclass\_name as recompile\_cause,

shredded.recompile\_count

FROM sys.dm\_xe\_session\_targets AS xet

JOIN sys.dm\_xe\_sessions AS xes ON (xe.address = xet.event\_session\_address)

CROSS APPLY ( SELECT CAST(xet.target\_data as xml) ) as target\_data\_xml ([xml])

CROSS APPLY target\_data\_xml.[xml].nodes('/HistogramTarget/Slot') AS nodes (slot\_data)

CROSS APPLY

(

SELECT nodes.slot\_data.value('(value)[1]', 'int') AS recompile\_cause,

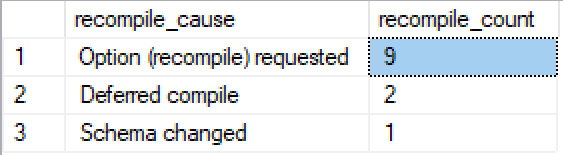
nodes.slot\_data.value('(@count)[1]', 'int') AS recompile\_count

) as shredded

JOIN sys.trace\_subclass\_values AS sv ON shredded.recompile\_cause = sv.subclass\_value

WHERE xes.name = 'RecompileReasons'

AND sv.trace\_event\_id = 37 -- SP:Recompile

[[](https://www.sqlhammer.com/5-extended-events-sessions-sql-server-instance-cannot-live-without/recompile-reasons-1/)<img class="alignnone size-full wp-image-3862" src="https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/recompile-reasons-1.png?resize=563%2C155&#038;ssl=1" alt="" width="563" height="155" srcset="https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/recompile-reasons-1.png?w=563&amp;ssl=1 563w, https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/recompile-reasons-1.png?resize=300%2C83&amp;ssl=1 300w" sizes="(max-width: 563px) 100vw, 563px" data-recalc-dims="1" />](https://www.sqlhammer.com/5-extended-events-sessions-sql-server-instance-cannot-live-without/recompile-reasons-1/)

5. Memory grants

Session

This session captures queries where the memory grant is larger than the filter predicate, 2 MB in this example. It will output the granted memory, the actual used memory, degree of parallelism, and a few other useful data points.

CREATE EVENT SESSION [MemoryUsage] ON SERVER

ADD EVENT sqlserver.query\_memory\_grant\_usage

(

ACTION

(

sqlserver.query\_hash

,sqlserver.query\_plan\_hash

,sqlserver.sql\_text

)

WHERE granted\_memory\_kb > 2048 --2 MB

)

ADD TARGET package0.ring\_buffer(SET max\_events\_limit=(0 /\*unlimited\*/),max\_memory=(1048576 /\*1 GB\*/))

WITH (STARTUP\_STATE=OFF,MAX\_DISPATCH\_LATENCY = 5SECONDS)

CREATE EVENT SESSION [MemoryUsage] ON SERVER

ADD EVENT sqlserver.query\_memory\_grant\_usage

(

ACTION

(

sqlserver.query\_hash

,sqlserver.query\_plan\_hash

,sqlserver.sql\_text

)

WHERE granted\_memory\_kb > 2048 --2 MB

)

ADD TARGET package0.ring\_buffer(SET max\_events\_limit=(0 /\*unlimited\*/),max\_memory=(1048576 /\*1 GB\*/))

WITH (STARTUP\_STATE=OFF,MAX\_DISPATCH\_LATENCY = 5SECONDS)

CREATE EVENT SESSION [MemoryUsage] ON SERVER

ADD EVENT sqlserver.query\_memory\_grant\_usage

(

ACTION

(

sqlserver.query\_hash

,sqlserver.query\_plan\_hash

,sqlserver.sql\_text

)

WHERE granted\_memory\_kb > 2048 --2 MB

)

ADD TARGET package0.ring\_buffer(SET max\_events\_limit=(0 /\*unlimited\*/),max\_memory=(1048576 /\*1 GB\*/))

WITH (STARTUP\_STATE=OFF,MAX\_DISPATCH\_LATENCY = 5SECONDS)

Query

--/\* Comment this part out after you run it once, unless you want to refresh the temp table.

IF OBJECT\_ID('tempdb..#capture\_waits\_data') IS NOT NULL

DROP TABLE #capture\_waits\_data

SELECT CAST(target\_data as xml) AS targetdata

INTO #capture\_waits\_data

FROM sys.dm\_xe\_session\_targets xet

JOIN sys.dm\_xe\_sessions xes

ON xes.address = xet.event\_session\_address

WHERE xes.name = 'MemoryUsage'

AND xet.target\_name = 'ring\_buffer';

--\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SELECT xed.event\_data.value('(@timestamp)[1]', 'datetime2') AS datetime\_utc,

DATEADD(hh,DATEDIFF(hh, GETUTCDATE(), CURRENT\_TIMESTAMP),xed.event\_data.value('(@timestamp)[1]', 'datetime2')) AS datetime\_local,

xed.event\_data.value('(@name)[1]', 'varchar(50)') AS event\_type,

xed.event\_data.value('(data[@name="ideal\_memory\_kb"]/value)[1]', 'bigint') AS ideal\_memory\_kb,

xed.event\_data.value('(data[@name="granted\_memory\_kb"]/value)[1]', 'bigint') AS granted\_memory\_kb,

xed.event\_data.value('(data[@name="used\_memory\_kb"]/value)[1]', 'bigint') AS used\_memory\_kb,

xed.event\_data.value('(data[@name="usage\_percent"]/value)[1]', 'int') AS usage\_percent,

xed.event\_data.value('(data[@name="dop"]/value)[1]', 'int') AS dop,

xed.event\_data.value('(data[@name="granted\_percent"]/value)[1]', 'int') AS granted\_percent,

xed.event\_data.value('(action[@name="sql\_text"]/value)[1]', 'varchar(max)') AS sql\_text,

xed.event\_data.value('(action[@name="query\_plan\_hash"]/value)[1]', 'numeric(20)') AS query\_plan\_hash,

xed.event\_data.value('(action[@name="query\_hash"]/value)[1]', 'numeric(20)') AS query\_hash

FROM #capture\_waits\_data

CROSS APPLY targetdata.nodes('//RingBufferTarget/event') AS xed (event\_data)

WHERE 1=1

--/\* Search for large memory grants.

AND xed.event\_data.value('(data[@name="used\_memory\_kb"]/value)[1]', 'bigint') > 5120 -- 5MB

--\*/

--/\* Search for grants too large for the actual used

AND xed.event\_data.value('(data[@name="usage\_percent"]/value)[1]', 'bigint') < 50

--\*/

--/\* Comment this part out after you run it once, unless you want to refresh the temp table.

IF OBJECT\_ID('tempdb..#capture\_waits\_data') IS NOT NULL

DROP TABLE #capture\_waits\_data

SELECT CAST(target\_data as xml) AS targetdata

INTO #capture\_waits\_data

FROM sys.dm\_xe\_session\_targets xet

JOIN sys.dm\_xe\_sessions xes

ON xes.address = xet.event\_session\_address

WHERE xes.name = 'MemoryUsage'

AND xet.target\_name = 'ring\_buffer';

--\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SELECT xed.event\_data.value('(@timestamp)[1]', 'datetime2') AS datetime\_utc,

DATEADD(hh,DATEDIFF(hh, GETUTCDATE(), CURRENT\_TIMESTAMP),xed.event\_data.value('(@timestamp)[1]', 'datetime2')) AS datetime\_local,

xed.event\_data.value('(@name)[1]', 'varchar(50)') AS event\_type,

xed.event\_data.value('(data[@name="ideal\_memory\_kb"]/value)[1]', 'bigint') AS ideal\_memory\_kb,

xed.event\_data.value('(data[@name="granted\_memory\_kb"]/value)[1]', 'bigint') AS granted\_memory\_kb,

xed.event\_data.value('(data[@name="used\_memory\_kb"]/value)[1]', 'bigint') AS used\_memory\_kb,

xed.event\_data.value('(data[@name="usage\_percent"]/value)[1]', 'int') AS usage\_percent,

xed.event\_data.value('(data[@name="dop"]/value)[1]', 'int') AS dop,

xed.event\_data.value('(data[@name="granted\_percent"]/value)[1]', 'int') AS granted\_percent,

xed.event\_data.value('(action[@name="sql\_text"]/value)[1]', 'varchar(max)') AS sql\_text,

xed.event\_data.value('(action[@name="query\_plan\_hash"]/value)[1]', 'numeric(20)') AS query\_plan\_hash,

xed.event\_data.value('(action[@name="query\_hash"]/value)[1]', 'numeric(20)') AS query\_hash

FROM #capture\_waits\_data

CROSS APPLY targetdata.nodes('//RingBufferTarget/event') AS xed (event\_data)

WHERE 1=1

--/\* Search for large memory grants.

AND xed.event\_data.value('(data[@name="used\_memory\_kb"]/value)[1]', 'bigint') > 5120 -- 5MB

--\*/

--/\* Search for grants too large for the actual used

AND xed.event\_data.value('(data[@name="usage\_percent"]/value)[1]', 'bigint') < 50

--\*/

--/\* Comment this part out after you run it once, unless you want to refresh the temp table.

IF OBJECT\_ID('tempdb..#capture\_waits\_data') IS NOT NULL

DROP TABLE #capture\_waits\_data

SELECT CAST(target\_data as xml) AS targetdata

INTO #capture\_waits\_data

FROM sys.dm\_xe\_session\_targets xet

JOIN sys.dm\_xe\_sessions xes

ON xes.address = xet.event\_session\_address

WHERE xes.name = 'MemoryUsage'

AND xet.target\_name = 'ring\_buffer';

--\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

SELECT xed.event\_data.value('(@timestamp)[1]', 'datetime2') AS datetime\_utc,

DATEADD(hh,DATEDIFF(hh, GETUTCDATE(), CURRENT\_TIMESTAMP),xed.event\_data.value('(@timestamp)[1]', 'datetime2')) AS datetime\_local,

xed.event\_data.value('(@name)[1]', 'varchar(50)') AS event\_type,

xed.event\_data.value('(data[@name="ideal\_memory\_kb"]/value)[1]', 'bigint') AS ideal\_memory\_kb,

xed.event\_data.value('(data[@name="granted\_memory\_kb"]/value)[1]', 'bigint') AS granted\_memory\_kb,

xed.event\_data.value('(data[@name="used\_memory\_kb"]/value)[1]', 'bigint') AS used\_memory\_kb,

xed.event\_data.value('(data[@name="usage\_percent"]/value)[1]', 'int') AS usage\_percent,

xed.event\_data.value('(data[@name="dop"]/value)[1]', 'int') AS dop,

xed.event\_data.value('(data[@name="granted\_percent"]/value)[1]', 'int') AS granted\_percent,

xed.event\_data.value('(action[@name="sql\_text"]/value)[1]', 'varchar(max)') AS sql\_text,

xed.event\_data.value('(action[@name="query\_plan\_hash"]/value)[1]', 'numeric(20)') AS query\_plan\_hash,

xed.event\_data.value('(action[@name="query\_hash"]/value)[1]', 'numeric(20)') AS query\_hash

FROM #capture\_waits\_data

CROSS APPLY targetdata.nodes('//RingBufferTarget/event') AS xed (event\_data)

WHERE 1=1

--/\* Search for large memory grants.

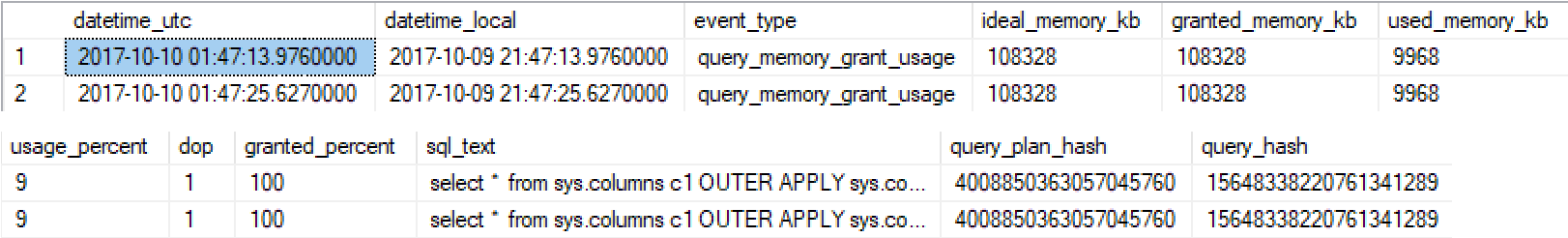
AND xed.event\_data.value('(data[@name="used\_memory\_kb"]/value)[1]', 'bigint') > 5120 -- 5MB

--\*/

--/\* Search for grants too large for the actual used

AND xed.event\_data.value('(data[@name="usage\_percent"]/value)[1]', 'bigint') < 50

--\*/

[[](https://www.sqlhammer.com/5-extended-events-sessions-sql-server-instance-cannot-live-without/memory-query/)<img class="alignnone size-full wp-image-3863" src="https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/memory-query.png?resize=645%2C98&#038;ssl=1" alt="" width="645" height="98" srcset="https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/memory-query.png?w=1622&amp;ssl=1 1622w, https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/memory-query.png?resize=300%2C45&amp;ssl=1 300w, https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/memory-query.png?resize=768%2C116&amp;ssl=1 768w, https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/memory-query.png?resize=1024%2C155&amp;ssl=1 1024w, https://i1.wp.com/www.sqlhammer.com/wp-content/uploads/2017/10/memory-query.png?w=1290&amp;ssl=1 1290w" sizes="(max-width: 645px) 100vw, 645px" data-recalc-dims="1" />](https://www.sqlhammer.com/5-extended-events-sessions-sql-server-instance-cannot-live-without/memory-query/)

Scripts

Click [here](https://www.sqlhammer.com/wp-content/uploads/2017/10/ExtendedEventQueriesAndSessions.zip) to download the full set of scripts.

NOTE --- SAVED AS EXTENDEDEVENTSQUERIESANDSESSIONS.ZIP