This site uses cookies for analytics, personalized content and ads. By continuing to browse this site, you agree to this use. Learn more

Troubleshooting slow running query using Extended Events Wait info event

Rate this article ★★★★☆

Sakthivel Chidambaram February 20, 2011

1

Share 2 0 0

Extended Events introduced in SQL Server 2008 is a detailed event infrastructure for SQL Server. If your environment is running with more SQL Server's of version > 2008, it is the right time to learn about Extended events since they help you a lot in troubleshooting SQL Server performance issues.

In general there are two things possible in SQL Server: Running *or* Waiting. Most often the percentage of wait will be more than percentage of running in a given Environment. So you should start with minimizing the waits percentage and once that comes below the running percentage, start focussing on minimizing the running percentage by tuning the query (query hints, plan guide etc..). So, I'm more interested in the troubleshooting methodology of using Waits info in case of SQL Server peformance issues because you fix the main culprit first. This whitepaper talks about the methodolody of troubleshooting performance issues based on Waits and Queues. (Note: Sometimes you might have to troubleshoot both waits and running percentage parallely depening on the situation)

While we troubleshoot a performance issue based on waits, there is a limitation that all existing DMV's, DMF's provide server lever waits. So if you need to troubleshoot waits encountered by one particular session or a query... What is the option??. The option is using the Extended Events available in SQL Server 2008 and higher version.

Here is the XEvents query to track the waits information for a query being executed from particular session. I'm also attaching the t-sql script, if the browser is not rendering this page correctly.

This code is provided AS-IS with no implied warranty

```
-- *** Change the Session ID values as per your need ***



-- Check for existing session

IF EXISTS(SELECT * FROM sys.server_event_sessions WHERE name='Waits_of_Particular_Session')

  DROP EVENT session Waits_of_Particular_Session ON SERVER;

GO
```

```
-- Create a session and add events and actions

CREATE EVENT SESSION Waits_of_Particular_Session

ON SERVER

ADD EVENT sqlserver.sql_statement_starting

(

        ACTION (sqlserver.session_id,

                sqlserver.sql_text,

                sqlserver.plan_handle)

        WHERE

                sqlserver.session_id > 52 AND sqlserver.session_id < 54

),

ADD EVENT sqlos.wait_info

(

        ACTION (sqlserver.session_id,

                sqlserver.sql_text,

                sqlserver.plan_handle)

        WHERE

                sqlserver.session_id > 52 AND sqlserver.session_id < 54

),

ADD EVENT sqlos.wait_info_external

(

        ACTION (sqlserver.session_id,

                sqlserver.sql_text,

                sqlserver.plan_handle)

        WHERE

                sqlserver.session_id > 52 AND sqlserver.session_id < 54

),

ADD EVENT sqlserver.sql_statement_completed

(
```

```
        ACTION (sqlserver.session_id,

                sqlserver.sql_text,

                sqlserver.plan_handle)

        WHERE

                sqlserver.session_id > 52 AND sqlserver.session_id < 54

)

ADD TARGET package0.asynchronous_file_target

    (SET filename=N'C:\Temp\Waits_of_Particular_Session.xel')

WITH (MAX_DISPATCH_LATENCY = 5 SECONDS, EVENT_RETENTION_MODE=ALLOW_SINGLE_EVENT_LOSS, TRACK_CAUS
ALITY=ON)

GO



-- Start the Session

ALTER EVENT SESSION Waits_of_Particular_Session ON SERVER

STATE = START

GO



-- Run the query from the SPID



-- Code to stop the session once the query execution completes

ALTER EVENT SESSION Waits_of_Particular_Session ON SERVER

STATE = STOP

GO



-- Parse the XML to show wait details

SELECT

    event_xml.value('(./@name)', 'varchar(1000)') as Event_Name,

    event_xml.value('(./data[@name="wait_type"]/text)[1]', 'nvarchar(max)') as Wait_Type,

    event_xml.value('(./data[@name="duration"]/value)[1]', 'int') as Duration,

    event_xml.value('(./data[@name="opcode"]/text)[1]', 'varchar(100)') as Operation,
```

```
    event_xml.value('(./action[@name="session_id"]/value)[1]', 'int') as SPID,

    event_xml.value('(./action[@name="sql_text"]/value)[1]', 'nvarchar(max)') as TSQLQuery,

    event_xml.value('(./action[@name="plan_handle"]/value)[1]', 'nvarchar(max)') as PlanHandle

FROM

        (SELECT CAST(event_data AS XML) xml_event_data FROM sys.fn_xe_file_target_read_file
('C:\Temp\Waits_of_Particular_Session*.xel', 'C:\Temp\Waits_of_Particular_Session*.xem', NULL, N
ULL)) AS event_table

        CROSS APPLY xml_event_data.nodes('//event') n (event_xml)

WHERE

        event_xml.value('(./@name)', 'varchar(1000)') IN ('wait_info','wait_info_external')



-- Code to drop the session

DROP EVENT SESSION Waits_of_Particular_Session ON SERVER;
```

Start the XEVENTS session -> Run your queries from the SPID being monitored by this XEVENTS session -> Once the query execution completes, stop the XEVENT session -> Run the portion of query above which will parse the XML and show you the waits detail for the particular SPID monitored.

Here is a sample output when I ran it in my machine:

Results | Messages

	Event_Name	Wait_Type	Duration	Operation	SPID	TSQLQuery	PlanHandle
1	wait_info	SOS_SCHEDULER_YIELD	0	Begin	53	select * from tbl_REQUESTS where session_id=54	<plan handle="0x060007004567001F40813785000000000...
2	wait_info	SOS_SCHEDULER_YIELD	0	End	53	select * from tbl_REQUESTS where session_id=54	<plan handle="0x060007004567001F40813785000000000...
3	wait_info	SOS_SCHEDULER_YIELD	0	Begin	53	select * from tbl_REQUESTS where session_id=54	<plan handle="0x060007004567001F40813785000000000...
4	wait_info	SOS_SCHEDULER_YIELD	0	End	53	select * from tbl_REQUESTS where session_id=54	<plan handle="0x060007004567001F40813785000000000...
5	wait_info	SOS_SCHEDULER_YIELD	0	Begin	53	select * from tbl_REQUESTS where session_id=54	<plan handle="0x060007004567001F40C10E8500000000...
6	wait_info	SOS_SCHEDULER_YIELD	0	End	53	select * from tbl_REQUESTS where session_id=54	<plan handle="0x060007004567001F40C10E8500000000...
7	wait_info	SOS_SCHEDULER_YIELD	0	Begin	53	select * from tbl_REQUESTS where session_id=54	<plan handle="0x060007004567001F40C10E8500000000...
8	wait_info	SOS_SCHEDULER_YIELD	0	End	53	select * from tbl_REQUESTS where session_id=54	<plan handle="0x060007004567001F40C10E8500000000...
9	wait_info	PAGEIOLATCH_SH	0	Begin	53	select * from readtrace.tblBatches where SPID=60	<plan handle="0x06000700A142413240E1648400000000...
10	wait_info	PAGEIOLATCH_SH	13	End	53	select * from readtrace.tblBatches where SPID=60	<plan handle="0x06000700A142413240E1648400000000...
11	wait_info	PAGEIOLATCH_SH	0	Begin	53	select * from readtrace.tblBatches where SPID=60	<plan handle="0x06000700A142413240E1648400000000...
12	wait_info	PAGEIOLATCH_SH	6	End	53	select * from readtrace.tblBatches where SPID=60	<plan handle="0x06000700A142413240E1648400000000...
13	wait_info	PAGEIOLATCH_SH	0	Begin	53	select * from readtrace.tblBatches where SPID=60	<plan handle="0x06000700A142413240E1648400000000...
14	wait_info	PAGEIOLATCH_SH	6	End	53	select * from readtrace.tblBatches where SPID=60	<plan handle="0x06000700A142413240E1648400000000...
15	wait_info	PAGEIOLATCH_SH	0	Begin	53	select * from readtrace.tblBatches where SPID=60	<plan handle="0x06000700A142413240E1648400000000...
16	wait_info	PAGEIOLATCH_SH	1	End	53	select * from readtrace.tblBatches where SPID=60	<plan handle="0x06000700A142413240E1648400000000...
17	wait_info	PAGEIOLATCH_SH	0	Begin	53	select * from readtrace.tblBatches where SPID=60	<plan handle="0x06000700A142413240E1648400000000...
18	wait_info	PAGEIOLATCH_SH	6	End	53	select * from readtrace.tblBatches where SPID=60	<plan handle="0x06000700A142413240E1648400000000...

Refer http://blogs.msdn.com/b/sqlsakthi/archive/2011/02/08/different-status-of-a-spid-in-sql-server-and-what-do-they-mean.aspx for status of a SPID when waiting for a Resource.

Wait_of_Particular_Session.sql

Tags t-sql waits xevents

Comments (1)

Name *

Email *

Website

Post Comment

Sakthi's SQL Server Channel
February 22, 2011 at 4:14 am
You can check the status of a SPID using any of these queries: SELECT status ,* FROM sys . sysprocesses

Reply

Follow Us

Popular Tags

t-sql memory performance replication info waits setup connectivity failure adomd log shipping backup security engine tlog recovery SQL 2012 DBA tools single-user welcome

Archives

January 2018 (1)
June 2012 (1)
May 2012 (5)
April 2012 (2)
All of 2018 (1)
All of 2012 (9)
All of 2011 (25)
All of 2010 (7)

Privacy & Cookies Terms of Use Trademarks

Microsoft

© 2018 Microsoft