SQL Server: Monitoring Page Splits using Extended Events

Hi Friends,

We all know that page splits are expensive operation for the database engine, also causing excessive fragmentation. I was trying a few things with extended events and observed that we can track/monitor page splits using Extended Events.

select \* from sys.dm\_xe\_objects

where name = 'page\_split'

Extended Events is a very flexible framework to troubleshoot database engine performance issues. It allows deep dive troubleshooting which is sometimes not possible using traditional troubleshooting techniques.

In this article, I want to demonstrate how you can monitor page splits using Extended Events. Not just monitor that its happening, but also extract the database in which its happening, the query that’s causing it and the file id which is the end victim of page splits.

Let us being. I have commented my code well.

First, just to show you some metadata, you can query a few DMVs to see that the default XE package has page split event.

SET NOCOUNT ON;

USE tempdb;

GO

-- sqlserver package has a page\_split event

select \* from sys.dm\_xe\_objects

where name = 'page\_split'

-- and the event has some default columns that can be trapped

select \* from sys.dm\_xe\_object\_columns

where object\_name = 'page\_split';

Next, we want to create an event session based on the above event. Apart from the default payload, we want to extract the SQL query that is causing page splits, amongst, database id, etc. This can be done using Actions. We want the event data to be recorded and we choose File Target.

-- but we also want the sql query (text) that causes page split

-- additional event data can be gathered using actions

-- so lets create an event first

-- make sure the directory u specify exists in c drive or specify your own path

CREATE EVENT SESSION Page\_Splits

ON SERVER

ADD EVENT

 sqlserver.page\_split

 (

 ACTION

 (

 sqlserver.database\_id,

 sqlserver.client\_app\_name,

 sqlserver.sql\_text

 )

 )

ADD TARGET package0.asynchronous\_file\_target

( SET filename = 'C:\Amit\track\_page\_splits.xel',

metadatafile = 'C:\Amit\track\_page\_splits.mta',

max\_file\_size = 10,

max\_rollover\_files = 10);

GO

Finally, start the event.

-- start the event

ALTER EVENT SESSION Page\_Splits

ON SERVER

STATE = START

GO

Now, let us simulate page splits. The below code is very self-explanatory to generate some page splits.

SET NOCOUNT ON;

USE tempdb;

GO

-- Create table T1

CREATE TABLE dbo.T1

(

 cl\_c1 UNIQUEIDENTIFIER NOT NULL DEFAULT(NEWID()),

 cl\_testData CHAR(2000) NOT NULL DEFAULT('sqlservergeeks.com')

);

GO

CREATE UNIQUE CLUSTERED INDEX idx\_cl\_c1 ON dbo.T1(cl\_c1);

GO

-- Insert rows (run for a few seconds then stop)

SET NOCOUNT ON;

USE tempdb;

TRUNCATE TABLE dbo.T1;

DECLARE @count AS INT

SET @count = 0;

WHILE @count <=100

BEGIN

 INSERT INTO dbo.T1 DEFAULT VALUES;

 SET @count = @count + 1;

END

-- check the data

select \* from T1

Now, let us get back to the Extended Events code. Meanwhile, you can see that 2 files have been created in the folder you specified in Target section.

The file size might still show 0 KB as the event session is running and we have not yet extract anything from the target. Now let us extract the event data. Replace the filenames appropriately. Extended Events framework provides a function to read the event data file which I am using here to get the data into a temporary table.

-- extract the data

-- replace the filename

DECLARE @xel\_filename varchar(256) = 'C:\Amit\track\_page\_splits\_0\_129646064157130000.xel'

DECLARE @mta\_filename varchar(256) = 'C:\Amit\track\_page\_splits\_0\_129646064157130000.mta'

SELECT CONVERT(xml, event\_data) as Event\_Data

INTO #File\_Data

FROM sys.fn\_xe\_file\_target\_read\_file(@xel\_filename, @mta\_filename, NULL, NULL)

Next, you can preview the data.

-- preview the data

select \* from #File\_Data

You can click on the XML link in the results pane and you will observe XML payload something like this…

--------------------------------------------------------------

<event name="page\_split" package="sqlserver" id="71" version="1" timestamp="2011-11-01T07:33:58.395Z">

 <data name="file\_id">

 <value>1</value>

 <text />

 </data>

 <data name="page\_id">

 <value>621</value>

 <text />

 </data>

 <action name="database\_id" package="sqlserver">

 <value>2</value>

 <text />

 </action>

 <action name="client\_app\_name" package="sqlserver">

 <value>Microsoft SQL Server Management Studio - Query</value>

 <text />

 </action>

 <action name="sql\_text" package="sqlserver">

 <value>DECLARE @count AS INT

SET @count = 0;

WHILE @count &lt;=50

BEGIN

 INSERT INTO dbo.T1 DEFAULT VALUES;

 SET @count = @count + 1;

END</value>

 <text />

 </action>

</event>

--------------------------------------------------------------

We can use XQuery to read the XML event information.

-- extract what you need using XQuery

-- you can modify this to suit your requirement

SELECT database\_name, sql\_text

FROM

(

SELECT

DB\_NAME(event\_data.value('(event/action[1]/value)[1]', 'VARCHAR(100)'))

AS database\_name,

Event\_Data.value('(event/action[3]/value)[1]', 'VARCHAR(4000)') AS sql\_text

FROM #File\_Data

) as table\_

When you run the above query, you can see the database name, the SQL query, etc that caused the splits. You can customize the above query to extract file id, page id, etc, if need be.

That’s it friends. Hope you enjoyed this article.

If you liked this article, do like us on FaceBook at <http://www.FaceBook.com/SQLServerGeeks>

Also tweet and pingback if you wish so.

Wait… you can stop the event now, if you wish so :)

-- stop the event

ALTER EVENT SESSION Page\_Splits

ON SERVER

STATE = STOP

GO

Regards

@A\_Bansal

@SQLServerGeeks

<http://www.amitbansal.net>

Visit my FaceBook page at <http://www.facebook.com/pages/Amit-Bansal/165298110161387>