Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

SQL Server **Extended Events** is a powerful tool for monitoring and troubleshooting database performance issues, such as blocking, deadlocks, and long-running queries.

Here's how you can set up **Extended Events** to capture each of these scenarios.

****Please check below given EXTENDED Events Scripts in NON-Production DB Servers initially****

1. Capturing Blocking in SQL Server

Blocking occurs when one session holds a lock on a resource, and other sessions are waiting for that resource to be released. Extended Events can help you identify the blocking session and the blocked processes.

Here is a script to capture blocking using Extended Events:

```
CREATE EVENT SESSION [BlockingEvents]

ON SERVER

ADD EVENT sqlserver.blocked_process_report

(

    ACTION (sqlserver.client_app_name, sqlserver.client_hostname, sqlserver.database_id, sqlserver.sql_text, sqlserver.session_id)

    WHERE ([sqlserver].[database_id] = DB_ID('YourDatabaseName'))  -- Optionally filter for a specific database

)

ADD TARGET package0.event_file

(

    SET filename = 'C:\SQLServer\BlockingEvents.xel', max_file_size = 50, max_rollover_files = 5

)

WITH (MAX_MEMORY = 4096 KB, EVENT_RETENTION_MODE = ALLOW_SINGLE_EVENT_LOSS, MAX_DISPATCH_LATENCY = 30 SECONDS, TRACK_CAUSALITY = ON, STARTUP_STATE = ON);



-- Start the session

ALTER EVENT SESSION [BlockingEvents] ON SERVER STATE = START;
```

Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

Explanation:

- **sqlserver.blocked_process_report**: This event is triggered whenever a blocking situation occurs and lasts longer than the defined **blocked process threshold**.
- **Actions**: Captures additional information like the client application, hostname, database ID, SQL text, and session ID.
- **Event Target**: Saves the output to a file (`.xel`) for later review.
- **Optional Filter**: Use `WHERE ([sqlserver].[database_id] = DB_ID('YourDatabaseName'))` to filter for a specific database if needed.

Configure the `blocked process threshold`: You should set the threshold to define how long a blocking process must exist before it generates a blocked process report. The threshold is specified in seconds.

```
EXEC sp_configure 'show advanced options', 1;

RECONFIGURE;

EXEC sp_configure 'blocked process threshold', 10;  -- Set to 10 seconds

RECONFIGURE;
```

Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

2. Capturing Deadlocks in SQL Server

Deadlocks occur when two or more sessions are waiting on each other to release locks, and neither can proceed. SQL Server automatically resolves deadlocks by killing one of the processes (the deadlock victim). Extended Events can capture deadlock graphs for troubleshooting.

Here is the script to capture deadlocks using Extended Events:

```
CREATE EVENT SESSION [DeadlockEvents]

ON SERVER

ADD EVENT sqlserver.deadlock_graph

(

    ACTION (sqlserver.client_app_name, sqlserver.client_hostname, sqlserver.database_id, sqlserver.sql_text, sqlserver.session_id)

    WHERE ([sqlserver].[database_id] = DB_ID('YourDatabaseName'))  -- Optionally filter for a specific database

)

ADD TARGET package0.event_file

(

    SET filename = 'C:\SQLServer\DeadlockEvents.xel', max_file_size = 50, max_rollover_files = 5

)

WITH (MAX_MEMORY = 4096 KB, EVENT_RETENTION_MODE = ALLOW_SINGLE_EVENT_LOSS, MAX_DISPATCH_LATENCY = 30 SECONDS, TRACK_CAUSALITY = ON, STARTUP_STATE = ON);



-- Start the session

ALTER EVENT SESSION [DeadlockEvents] ON SERVER STATE = START;
```


Explanation:

- **sqlserver.deadlock_graph**: This event captures detailed information about deadlock situations, including a graph of the processes involved in the deadlock.
- **Actions**: Captures extra information like the SQL text, session ID, and application details to help identify the source of the deadlock.
- **Event Target**: Outputs the deadlock graph and associated details to an event file (`.xel`).

You can later analyze these `.xel` files using SQL Server Management Studio (SSMS) or custom queries.

Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

3. Capturing Long-Running Queries

Identifying long-running queries is crucial for performance optimization. SQL Server Extended Events can be used to capture query execution details that exceed a specified duration.

Here is a script to capture long-running queries using Extended Events:

```
CREATE EVENT SESSION [LongRunningQueries]

ON SERVER

ADD EVENT sqlserver.rpc_completed

(

    ACTION (sqlserver.client_app_name, sqlserver.client_hostname, sqlserver.database_id, sqlserver.sql_text, sqlserver.session_id)

    WHERE (duration > 5000000 AND [sqlserver].[database_id] = DB_ID('YourDatabaseName'))  -- Filter queries longer than 5 seconds

)

ADD EVENT sqlserver.sql_batch_completed

(

    ACTION (sqlserver.client_app_name, sqlserver.client_hostname, sqlserver.database_id, sqlserver.sql_text, sqlserver.session_id)

    WHERE (duration > 5000000 AND [sqlserver].[database_id] = DB_ID('YourDatabaseName'))  -- Filter queries longer than 5 seconds

)

ADD TARGET package0.event_file

(

    SET filename = 'C:\SQLServer\LongRunningQueries.xel', max_file_size = 50, max_rollover_files = 5

)

WITH (MAX_MEMORY = 4096 KB, EVENT_RETENTION_MODE = ALLOW_SINGLE_EVENT_LOSS, MAX_DISPATCH_LATENCY = 30 SECONDS, TRACK_CAUSALITY = ON, STARTUP_STATE = ON);



-- Start the session

ALTER EVENT SESSION [LongRunningQueries] ON SERVER STATE = START;
```

Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

Explanation:

- **sqlserver.rpc_completed**: Captures the completion of stored procedures (remote procedure calls).
- **sqlserver.sql_batch_completed**: Captures the completion of ad-hoc SQL batches.
- **Duration Filter**: Only captures queries that run for longer than a specified time (in this case, 5 seconds, where **duration** is in microseconds).
- **Actions**: Collects useful diagnostic data like the SQL text, application name, hostname, session ID, and database ID.
- **Event Target**: Saves the output to a file (`.xel`), which can be opened in SSMS or queried later.

Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

4. Combining All Three Event Sessions

To capture all three (blockings, deadlocks, and long-running queries) in a single **Extended Event session**, you can combine them into one script as follows:

```
CREATE EVENT SESSION [DBMonitoring]

ON SERVER

-- Blocking Events

ADD EVENT sqlserver.blocked_process_report

(

    ACTION (sqlserver.client_app_name, sqlserver.client_hostname, sqlserver.database_id, sqlserver.sql_text, sqlserver.session_id)

    WHERE ([sqlserver].[database_id] = DB_ID('YourDatabaseName'))

)

-- Deadlock Events

ADD EVENT sqlserver.deadlock_graph

(

    ACTION (sqlserver.client_app_name, sqlserver.client_hostname, sqlserver.database_id, sqlserver.sql_text, sqlserver.session_id)

    WHERE ([sqlserver].[database_id] = DB_ID('YourDatabaseName'))

)

-- Long Running Queries - RPC and Batch Completed

ADD EVENT sqlserver.rpc_completed

(

    ACTION (sqlserver.client_app_name, sqlserver.client_hostname, sqlserver.database_id, sqlserver.sql_text, sqlserver.session_id)

    WHERE (duration > 5000000 AND [sqlserver].[database_id] = DB_ID('YourDatabaseName'))

),

ADD EVENT sqlserver.sql_batch_completed

(
```

Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

```
    ACTION (sqlserver.client_app_name, sqlserver.client_hostname, sqlserver.database_id, sqlserver.sql_text,
sqlserver.session_id)

    WHERE (duration > 5000000 AND [sqlserver].[database_id] = DB_ID('YourDatabaseName'))

)

-- Output all events to file

ADD TARGET package0.event_file

(

    SET filename = 'C:\SQLServer\DBMonitoring.xel', max_file_size = 50, max_rollover_files = 5

)

WITH (MAX_MEMORY = 4096 KB, EVENT_RETENTION_MODE = ALLOW_SINGLE_EVENT_LOSS,
MAX_DISPATCH_LATENCY = 30 SECONDS, TRACK_CAUSALITY = ON, STARTUP_STATE = ON);



-- Start the session

ALTER EVENT SESSION [DBMonitoring] ON SERVER STATE = START;
```

Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

5. Analyzing the Collected Data

Once the data has been collected, you can review it by querying the event file or using SQL Server Management Studio (SSMS):

```
-- Query the event file
SELECT
    event_data.value('(event/data[@name="sql_text"]/value)[1]', 'nvarchar(max)') AS sql_text,
    event_data.value('(event/data[@name="duration"]/value)[1]', 'bigint') AS duration,
    event_data.value('(event/data[@name="database_id"]/value)[1]', 'int') AS database_id,
    event_data.value('(event/data[@name="session_id"]/value)[1]', 'int') AS session_id
FROM sys.fn_xe_file_target_read_file('C:\SQLServer\DBMonitoring*.xel', NULL, NULL, NULL)
CROSS APPLY (SELECT CAST(event_data AS XML)) AS T(event_data);
```

This query reads the `.xel` files generated by the Extended Events session and extracts relevant information like the SQL text, duration, database ID, and session ID for analysis.

Summary:

1. The scripts provided enable you to capture database blockings, deadlocks, and long-running queries in SQL Server using Extended Events.
2. These tools provide detailed diagnostic information that can help you identify performance bottlenecks, troubleshoot locking issues, and optimize your SQL Server environment.