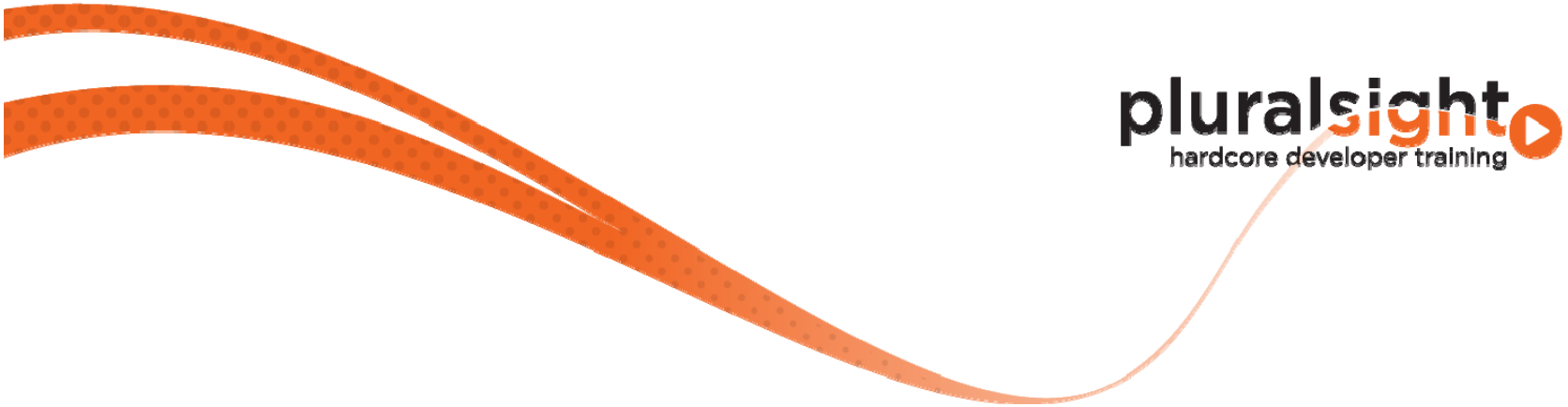
SQL Server: Advanced Extended Events

Module 7: Advanced Troubleshooting Scenarios

Jonathan M. Kehayias
Jonathan@SQLskills.com



pluralsight
hardcore developer training

Introduction

- **Extended Events can simplify the work of troubleshooting complex problems in SQL Server**
- **Properly leveraging all of the concepts from the SQL Server: Introduction to Extended Events course on Pluralsight along with the concepts in this course is important to successfully leverage Extended Events**

- **In this module we'll cover:**
  - Tempdb latch contention
  - Lock escalation
  - Problematic page splits
  - Troubleshooting orphaned transactions
  - Troubleshooting ASYNC_NETWORK_IO issues

Tempdb Latch Contention

- **Latch contention on allocation bitmap pages in tempdb can significantly affect performance of SQL Server**
  - Page Free Space (PFS) and Shared Global Allocation Map (SGAM) are the bitmaps where contention can occur
  - Contention on these pages occurs when tracking page allocation and deallocation with many small temp tables
  - Increasing the number of files can reduce contention on these pages as round-robin allocation divides the allocations over the available files
- **The latch_suspend_end event tracks when latch waits end inside of SQL Server by database_id, file_id, and page_id**
  - Using a predicate with the divides_evenly_by_int64 predicator can track contention that occurs on tempdb allocation pages specifically
- **Bucketing the events produced with the bucketizer target simplifies identification of allocation bitmap contention inside of tempdb**

Lock Escalation

- Lock escalation inside the engine can result in excessive blocking or deadlocks when the object or partition locks are held for long durations
- Identifying the statement(s) that caused lock escalation to occur is one of the first steps in resolving problems that result from the higher granularity locking that results from the escalation
- In SQL Server 2008, the sqlserver.lock_acquired event can be used to identify object level locking (resource_type = 5) along with the sqlserver.sp_statement_starting or sqlserver.sql_statement_starting events and TRACK_CAUSALITY to identify the cause of escalated locks
- In SQL Server 2012, the sqlserver.lock_escalation event provides all necessary information for troubleshooting lock escalation occurrences without having to collect additional data

Problematic Page Splits

- **Mid-page splits cause data to be moved from one page to a newly allocated page inside of a database**
  - Results in fragmentation
  - Generates significantly more transaction log records for the split operation
- **The sqlserver.page_split event does not distinguish between normal page allocation 'splits' that occur for increasing keys, vs. mid-page splits that result in data movement and higher fragmentation**
- **The sqlserver.transaction_log event can be used to identify the occurrences of the LOP_DELETE_SPLIT log operation, to track problematic page splits**
  - The database_id and alloc_unit_id can be used to find the objects splitting the most frequently
  - Can be used with fillfactor adjustments to verify improvements after rebuilding problem indexes

Troubleshooting Orphaned Transactions

- **Identifying the root cause of a session with an orphaned transaction can be incredibly difficult since the problem manifests long after the statements that generated the transaction were actually executed**
  - Extended Events cannot prevent the problem from occurring, but can provide the necessary information to troubleshoot the problem after the next occurrence
- **The database_transaction_begin and database_transaction_end events can be used along with the session_id action and the pair_matching target to identify sessions that have begin events with no corresponding end event**
  - Leveraging the tsql_frame action in the data collection allows identifying the exact line in code where the orphaned transaction began

Troubleshooting ASYNC_NETWORK_IO issues

- **ASYNC_NETWORK_IO waits generally occur when a client application takes too long to process returned records before notifying SQL Server that the records have been received**
  - In Extended Events the wait_type is NETWORK_IO in the wait_types map
- **Tracking the wait_info event along with the client_app_name action using the histogram target can identify the application that is causing the majority of waits**
- **Tracking the wait_info event along with the host_name action using the bucketizer target can identify the server that is causing the majority of waits in distributed applications**
  - Could be an incorrect network configuration, for example duplex speed for the network adapter, or a remote server with a slow network link

Summary

- **More advanced troubleshooting can be accomplished using Extended Events than other methods available in SQL Server**
- **In some cases the ability to leverage Extended Events for advanced troubleshooting requires additional information about how SQL Server functions**
  - More information about the internals of SQL Server can be found in the additional courses on Pluralsight by SQLskills.com
- **Many more scenarios for using Extended Events exist than were covered in this module**
  - These are merely suggested scenarios where advanced troubleshooting can be performed using Extended Events