Troubleshooting SQL Server with Extended Events

Marek Chmel
Lead Database Administrator, SQL Team at&t Czech Republic
MVP Data Platform

SQL Profiler and SQL Trace

- SQL Trace introduced with SQL Server 6.5
- SQL Profiler introduced with SQL Server 7.0
- Basic usage
  - Providing real-time insight into SQL Server Activity
  - Capturing queries and their usage
  - Auditing of user activity
  - Capturing a baseline
  - Performance troubleshooting tool

DEMO

SQL Trace and SQL Profiler

Extended Events

- Advanced event collection infrastructure introduced in SQL Server 2008
- Highly flexible implementation which allows complex configurations for event collection that simplify problem idenfification
- Examples
  - Capture stored procedures that exceed previous max duration, CPU, or I/O values
  - Identify statement timeouts/attention events
  - Capturing the first N executions of an event
  - Using the plan_handle and tsql_stack to capture execution plans and statement text
  - Capture session-level wait statistics
  - Examine details of the proportional-fill algorithm
  - Watch page splits occurring

Comparing Trace and XEvents

Trace	XEvents
Capture query info	Capture query info
Choose what to capture	Choose what to capture
Filter on different fields	Filter on different fields
Multiple options for analysis	Multiple options for analysis
	Multiple options for collection
	Flexible configuration
	Tracing newer features

Replacing SQL Trace

- The implementation of SQL Trace limited its flexibility and had negative impacts on performance during event collection
  - All events share a fixed set of data columns requiring some columns to be overloaded, providing different meanings for different events
  - Events generate all of the data columns, even when the trace doesn't require all of the data columns to be collected
  - Events fire if they are turned on in the bitmap in the trace controller filtering is applied, but filtering is only applied after the event has fired completely
- Trace I/O providers only allow for post-collection analysis of trace data

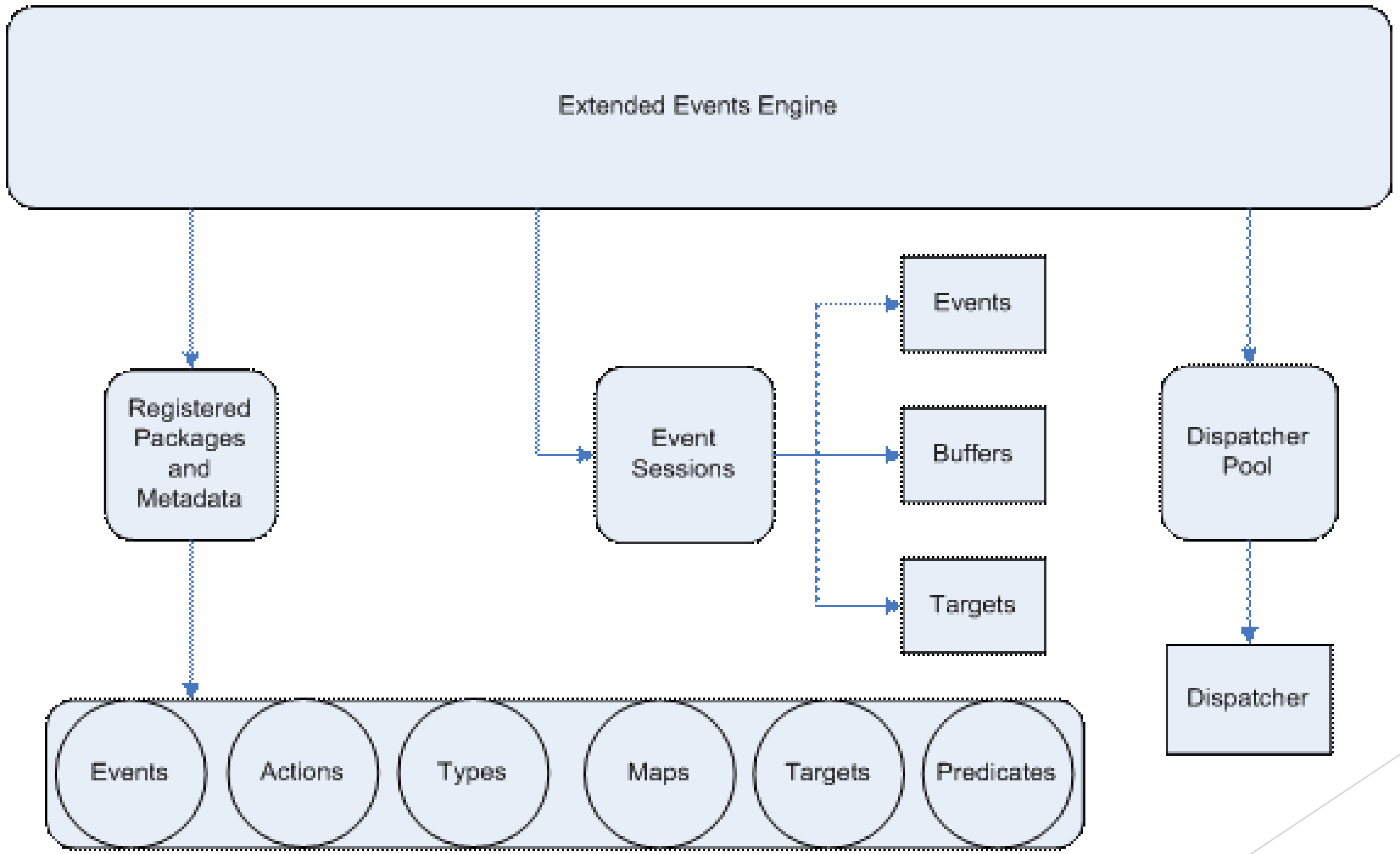
Changes in XEvents by SQL Server Version

SQL Server	Events in SQL Trace	Events in XEvents
2008 SP4	180	255
2008R2 SP3	180	264
2012 SP3	180	646
2014 SP1	180	749
2016	180	1303

XEvent architecture

- The Extended Events engine services diagnostic data collection from the modules loaded in the process
- Each module loads a package of metadata into the engine that provides information about the events provided by the module
- Event sessions provide a functional boundary for event collection
- Events only provide state information for the point in execution that the event was fired, additional information can be triggered through the use of actions

Architecture Layout



Objects: Packages

- Packages are loaded by individual modules at runtime
- Default package0 package is loaded by the Extended Events engine and contains generic objects that are not specific to any single module
  - E.g.: all targets, generic types, predicate comparators, and some actions
- Packages are containers that define the available objects and their definitions
- Packages are not a functional boundary of usage
  - Objects from one package can be used with objects from another package
- Examples of packages: sqlservr.exe, sqlos.dll

Objects: Events

- Events correspond to well-known points in the code
  - E.g. a Transact-SQL statement finished executing; a deadlock occurred
- Events deliver a basic payload of information
  - The payload is defined by a (versioned) schema of information immediately available to the event
  - Events may contain optional (customizable) data elements that are only collected when specified
  - Events will always return all non-customizable data elements
- Events are defined using the Event Tracing for Windows (ETW) model (channel, keyword) to allow integration with ETW

DEMO

XEvent packages and events

Objects: Predicates and Actions

- Predicates are Boolean expressions that define the conditions required for an event to actually fire
- Predicates support short-circuit evaluation
  - The first false evaluation prevents event from firing
- Predicates can use basic arithmetic operators, or textual comparators for more complex expressions
- Actions only execute after predicate evaluation determines the event will fire
- Actions execute synchronously on the thread that fired the event
- Actions collect additional state data to add to the event data
- Some actions have side effects like performing a memory dump

DEMO

Predicates and Actions

Objects: Targets

- Targets are the data consumers for Extended Events, and two targets provide functionality similar to what was previously available in SQL Trace:
  - The ring_buffer target provides an in-memory storage location for events being collected
  - The event_file target provides a file system storage location for events being collectedSynchronous and asynchronous targets exist
- Aggregating targets aggregate data based on criteria
  - Event Bucketizer (providing a histogram)
  - Event Counter
  - Event Pairing (which matches events)

DEMO

Targets

DEMO

Using XEvents UI

Default templates

- Count Query Locks
  - Counts occurrences of the sqlserver.lock_acquired event using the histogram target based on the query_hash action
  - This template can used to identify the most lock-intensive queries for investigation and tuning
- Query Batch Sampling
  - Collects SQL batch and RPC level statements as well as error information
  - This template can be used to understand the flow of queries that are executing on a server and track errors back to the queries that caused them
  - Events are only collected from 20% of the active sessions on the server at any given time
  - The sampling rate can be changed by modifying the filter for the event session

Default templates

- Query Batch Tracking
  - Collects all batch and RPC level statements as well as error information
  - This template can used to understand the flow of queries that are executing on your system and track errors back to the queries that caused them
- Query Detail Sampling
  - Collects detailed statement and error information
  - This template can be used to track each statement that has executed on your system as a result of query batches or stored procedures and track errors back to the specific statement that caused them
  - Also collects the query hash and query plan hash for every statement

Default Templates

- Query Detail Tracking
  - Collects detailed statement and error information
  - This template can be used to track each statement that has executed on your system as a result of query batches or stored procedures and track errors back to the specific statement that caused them
  - Also collects the query hash and query plan hash for every statement
- Query Wait Statistic
  - Collects internal and external wait statistics for individual query statements, batches and RPCs
  - Collects the query hash and query plan hash for every statement it tracks.
  - Events are only collected from 20% of the active sessions on the server at any given time
  - The sampling rate can be changed by modifying the filter for the event session

Default Templates

- Activity Tracking
  - Similar to the Default Trace that exists in the SQL Trace system
  - Does not include security audit events that are in the Default Trace, which are exposed by the SQL Server Audit feature instead
- Connection Tracking
  - Tracks connection activity for a server using the login and logout events
  - Includes the connectivity_ring_buffer_recorded event to diagnose any connection problems on the server
- Database Log File IO Tracking
  - Monitors the I/O for database log files, file_id = 2, on the server
  - Tracks asynchronous I/O, database log flushes, file writes, spinlock backoffs of type LOGFLUSHQ and waits of type WRITELOG
  - Collects raw data in a ring buffer and aggregates spinlock backoff information based on the input buffer (sql_text) in a histogram

Management DDLs

- CREATE EVENT SESSION
  - Creates a new event session based on the events, actions, predicates, targets, and session options provided
  - All event sessions are created in a stopped state
- ALTER EVENT SESSION
  - Add or remove events and targets from an event session
  - Change session configuration options for a stopped event session
  - Alter the state of an event session to start or stop
- DROP EVENT SESSION
  - Removes an event session from the system entirely
  - Memory-resident targets are not available after an event session is dropped

Troubleshooting Scenarios
Blocking issues

- The blocked_process_report event fires based on the value configured for the ‘blocked process threshold’ sp_configure option in the SQL Server
- XML report that contains information about the blocking and blocked processes in a blocking scenario for further debugging to identify and prevent the problem
- Setting the ‘blocked process threshold’ too low can result in excessive event generation
  - For example, if the threshold is set at 10 seconds and a blocking scenario lasts for 38 seconds, three blocked_process_report events will be generated (one every 10 seconds)
  - In the same example, if there are multiple blocked sessions in a blocking chain, each blocked session will generate a blocked_process_report event every 10 seconds

Troubleshooting Scenarios Recomplication issues

- The sql_statement_starting and sp_statement_starting events contain a ‘state’ column that specifies whether the statement was recompiled during execution
  - The state column is a mapped to the statement_starting_state map and provides three values: Normal, Recompiled, and Execution Plan Flush
  - Recompilation causes the event to fire twice: once for state=Recompiled and once for state=Normal
- The sql_statement_recompile event fires for any statement-level recompilation in the system
  - Ad hoc batches, stored procedures, and triggers are included
  - The recompile_cause column is mapped to the statement_recompile_cause map and provides the reason the recompile occurred

Troubleshooting Scenarios
Session Wait Statistics

- Understanding the causes of waits inside SQL Server can help identify performance bottlenecks and potential future problems
- The wait_info and wait_info_external events fire whenever a task has to wait during its execution
- Predicates on the session_id global field can allow tracking waits for a specific session in the server, or can be used to sample all sessions on the server

Troubleshooting Scenarios
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

Session End

Ing. Marek Chmel, MSc

Lead Database Administrator, Cloud, Platform, Application & Data Layer Team

mc654x@att.com or marek.chmel@technet.ms