SQL Server: Performance Troubleshooting Using Wait Statistics

Module 6: Summary

Paul S. Randal

Paul@SQLskills.com

pluralsight

hardcore developer training

Introduction

- **To round out the course, this module will cover:**
  - Summarized methodologies
  - Real-world, end-to-end example from a consulting client

Waits, Latches, and Spinlocks

- **We use wait statistics analysis as one of the first steps in every performance investigation we conduct**
  - Rarely does wait statistics analysis *alone* allow the problem to be diagnosed
  - It shows some of the symptoms of the problem

- **The vast majority of performance troubleshooting cases can be addressed simply by using wait statistics**
  - Occasionally latch statistics must be used and rarely must spinlocks also be examined
  - As discussed in Module 4, latch and spinlock analysis gets progressively more advanced, requiring deep knowledge of SQL Server internals

Methodology: No Historical Data

- **Gather information about exactly when the performance problem arose and the user-visible characteristics of the problem**
- **Gather information about what changed before the problem arose**
- **Examine the output from sys.dm_os_waiting_tasks**
  - What is happening on the server right now?
- **Examine the output from sys.dm_os_wait_stats**
  - What has happened in the past?
- **Look at the top 3-4 relevant waits**
  - If LATCH_XX is present, examine the output from sys.dm_os_latch_stats
- **Avoid the temptation to knee-jerk and equate symptoms with the root-cause**
- **Gather further information from relevant sources to pin-point the problem**
  - DMVs, query plans, performance counters, code analysis

Methodology: Historical Data

- **Gather information about exactly when the performance problem arose and the user-visible characteristics of the problem**
- **Gather information about what changed before the problem arose**
- **Examine the historical data sets from before the change and correlate through the time the problem arose**
  - Look to see how the pattern of waits changes over time
- **Investigate current output from sys.dm_os_wait_stats and sys.dm_os_waiting_tasks to see whether the pattern is continuing or has returned to 'normal'**
  - If normal now, correlate with any other historical data captured from monitoring tools to try to determine why the problem occurred
  - If not normal, proceed with further data gathering, focusing on the waits that have risen to prevalence over time

Real-World Example: Symptoms

- **Auto dealership hosting service**
  - Lots of auto dealers from across the US hosted on one site
  - Each auto dealer uploads inventory each day
  - One large Listing table storing all inventory for all dealers
  - One large Visitor table tracking clicks on web pages
  - No DBA
- **System had performance problem:**
  - User queries on inventory and prices regularly timed out
  - Inventory updates regularly timed out
  - Climbing CPU usage
  - Response time getting longer and longer
  - Car dealers pressuring hosting service for fixes

- **First step: analyze wait statistics…**

Real-World Example: Analysis

- **No historical data so gathered wait statistics data using the queries shown in earlier modules**
- **Both DMVs showed the same three wait types:**
  - CXPACKET wait = parallelism
  - PAGEIOLATCH_SH wait = reading data file pages from disk
  - WRITELOG wait = waiting for log writes, with average wait more than 20ms
- **Possible issues from just wait statistics**
  - Many queries doing parallel table scans of data that is not memory resident
  - I/O subsystem for the log file over-loaded and/or high number of log flushes
- **Investigated further using DMVs to analyze:**
  - Query plans
  - Index and table structures, index usage, fragmentation, and statistics
  - I/O subsystem latencies

- **Next step: determine root-causes…**

Real-World Example: Root-Causes

- **Both large tables had random GUID cluster keys**
  - High fragmentation in the clustered indexes leading to poor readahead
  - Lots of page-split transaction log activity during web page click tracking
- **Both large tables had more than 50 single-column nonclustered indexes**
  - Indexes not being used for seeks, resulting in table scans
  - Large amounts of nonclustered index maintenance from inserts, updates, deletes contributing to transaction log activity
- **Insufficient buffer pool memory for application workload data**
- **Poorly laid out I/O subsystem contributing to high latencies**
- **Poorly written code from using an ORM system**

- **Final step: propose and implement solution**

Real-World Example: Solution

- **Solution included:**
  - Increasing server memory and provisioning more appropriate I/O subsystem
  - Changing main tables to have bigint IDENTITY cluster keys
  - Removing useless nonclustered indexes
  - Analyzing ORM-generated code and query plans to determine appropriate nonclustered indexes
    - ORM system could not be removed for political reasons
  - Implementing index maintenance and periodic health checks

- **End result: no performance problems and a happy client, with minimal investigation time**

Resources

- **Whitepapers:**
  - Performance Tuning Using Waits and Queues (http://bit.ly/aUh6S)
  - Diagnosing and Resolving Latch Contention (http://bit.ly/pS1kd1)
  - Diagnosing and Resolving Spinlock Contention (http://bit.ly/qZEJ4h)

- **Other links:**
  - Paul’s blog category on waits, latches, and spinlocks (http://bit.ly/Nsb4QL)
  - SQL Server 2012 Books Online on OS DMVs (http://bit.ly/NsbqH9)
  - PSS Wait Stats Repository - defunct (http://bit.ly/9P5qNW)

- **Be careful of what advice you read on the Internet!**

Summary

- **Wait statistics analysis is the most effective first step when performance troubleshooting**
  - SQL Server collects a wealth of useful information
  - Shows what is happening and what has happened
- **Keys to using the data correctly are understanding and practice**
  - What do the various waits mean?
  - Which waits are relevant?
  - What are the average wait times?
  - Are there recognizable patterns?
  - Don’t be scared to look deeper at latch statistics
- **Don’t be tempted to ‘knee-jerk’ and waste time!**

- **Happy troubleshooting!**