

SQLskills
Sydney SQL Server User Group
10th December 2014

**Performance Troubleshooting Using
Wait Statistics (1/2-hr whirlwind tour!)**

Paul S. Randal
Paul@SQLskills.com





- **Team of world-renowned SQL Server experts:**
 - Erin Stellato (@ErinStellato)
 - Glenn Berry (@GlennAlanBerry)
 - Jonathan Kehayias (@SQLPoolBoy)
 - Kimberly L. Tripp (@KimberlyLTripp)
 - Paul S. Randal (@PaulRandal)
- **Instructor-led training: Immersion Events (US & Australia)**
- **Online training:** pluralsight <http://pluralsight.com/>
- **Consulting:** health checks, hardware, performance, upgrades
- **Remote DBA:** system monitoring and troubleshooting
- **Conferences:** PASS Summit, SQLintersection
- **Become a SQLskills Insider**
 - <http://www.sqlskills.com/Insider>



Australia 2015 Immersion Event

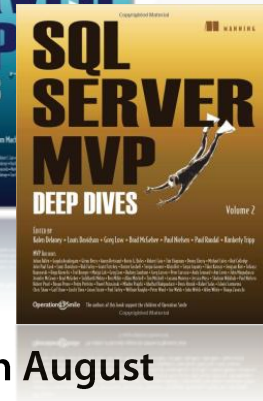
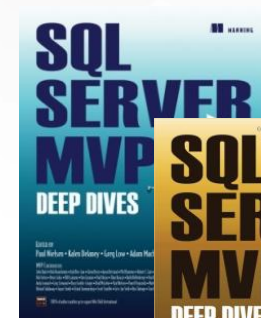
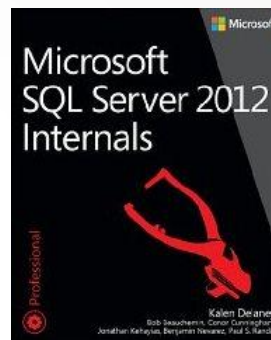
- **SQLskills IEPTO2: Immersion Event on Performance Tuning, Part 2**

SQL Server I/O	I/O Concepts for DBAs	Storage Area Networks for DBAs
SQLOS Scheduling and CPU Performance Tuning	SQLOS Memory Management and Memory Performance Tuning	Data Collection and Baselineing
Wait and Latch Statistics	Query Plan Analysis	Extended Events
Performance Issue Patterns	Statement Execution, Stored Procedures, and the Plan Cache	
Deadlock Analysis	Advanced Extended Events	

- **Sydney, NSW: week of 23 February 2015**
- **Register using the [Sydney](#) discount code and attend for the special price of US\$3,195**
 - Full price: US\$3,995, early-bird price of US\$3,495 valid until 31 Dec 2014
- **For more information: <http://www.sqlskills.com/sql-server-training/sydney-iepto2-20150223/>**

Author/Instructor: Paul S. Randal

- Consultant/Trainer/Speaker/Author
- CEO, [SQLskills.com](http://www.SQLskills.com)
 - Email: Paul@SQLskills.com
 - Blog: <http://www.SQLskills.com/blogs/Paul>
 - Twitter: @PaulRandal
- Contributing Editor of TechNet Magazine, author of the SQL Q&A column, articles on DBA topics, multiple whitepapers for Microsoft
- 5 years at DEC responsible for the VMS file-system and chkdsk equivalent
- Almost 9 years as developer/manager in the SQL Storage Engine team through August 2007, ultimately responsible for Core Storage Engine
 - Wrote DBCC component, other Storage Engine code, DBCC CHECKDB/repair for SQL 2005
- Regular presenter at worldwide conferences on HA/DR, administration, performance, and internals (#1 rated session and workshop at PASS Summit in 2013)
- Course author/instructor for Microsoft Certified Master qualifications
- Owner and Manager of the SQLintersection conference
- (I also like diving, electronics, robotics, Arduino, books, sheep...)





- Email paul@SQLskills.com with the subject line: **Sydney Pluralsight code** to get a FREE 30-day trial of our SQLskills content on Pluralsight
- For example:
 - <http://www.pluralsight.com/training/Courses/TableOfContents/sqlserver-waits>
 - 4.5 hours on waits, latches, spinlocks

Overview

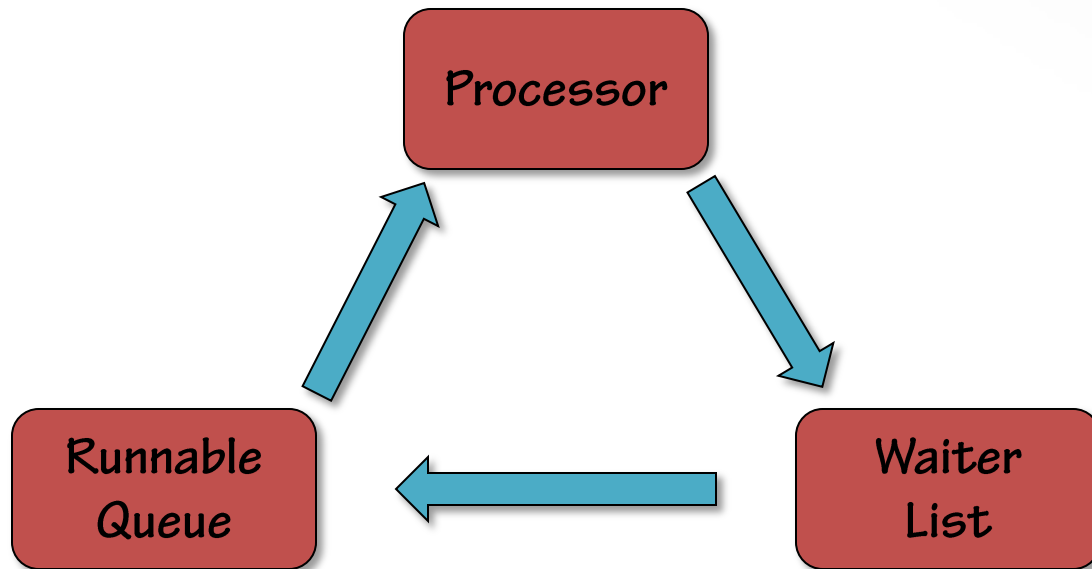
- **Don't do 'knee-jerk' performance troubleshooting**
 - Work through the data to see what may be the root cause
 - You'll end up spending less time overall
- **Proficiency in using wait statistics data comes from:**
 - Retrieving the data correctly
 - Understanding what common wait types mean
 - Recognizing patterns
 - Avoiding inappropriate Internet advice
 - Practice!
- **Even better is to have a series of snapshots of wait statistics over time**
 - Allows identification of changes and the time of the change
 - Allows trending

What are Waits?

- **The term 'wait' means that a thread running on a processor cannot proceed because a resource it requires is unavailable**
 - It has to wait until the resource is available
- **The resource being waited for is tracked by SQL Server**
 - Each resource maps to a wait type
- **Example resources that may be unavailable:**
 - A lock (LCK_M_XX wait type)
 - A data file page in the buffer pool (PAGEIOLATCH_XX wait type)
 - Results from part of a parallel query (CXPACKET wait type)
 - A latch (LATCH_XX wait type)

Components of a Scheduler

- All schedulers are composed of three 'parts'



- Threads transition around these parts until their work is complete
- One scheduler per processor core SQL Server can use

Thread States

- A thread can be in one of three states when being actively used as part of processing a query
- **RUNNING**
 - The thread is currently executing on the processor
- **SUSPENDED**
 - The thread is currently on the Waiter List waiting for a resource
- **RUNNABLE**
 - The thread is currently on the Runnable Queue waiting to execute on the processor
- Threads transition between these states until their work is complete

Wait Times Definition

- **Total time spent waiting:**
 - Known as 'wait time'
 - Time spent transitioning from RUNNING, through SUSPENDED, to RUNNABLE, and back to RUNNING
- **Time spent waiting for the resource to be available:**
 - Known as 'resource wait time'
 - Time spent on the Waiter List with state SUSPENDED
- **Time spent waiting to get the processor after resource is available:**
 - Known as 'signal wait time'
 - Time spent on the Runnable Queue with state RUNNABLE
- **Wait time = resource wait time + signal wait time**

Waits DMVs

■ **sys.dm_os_waiting_tasks**

- This DMV shows all threads that are currently suspended
- Think of it as the 'what is happening right now?' view of a server
- Usually very first thing to run when approaching a 'slow' server
 - The data is more useful when joined with other DMV results

■ **sys.dm_os_wait_stats**

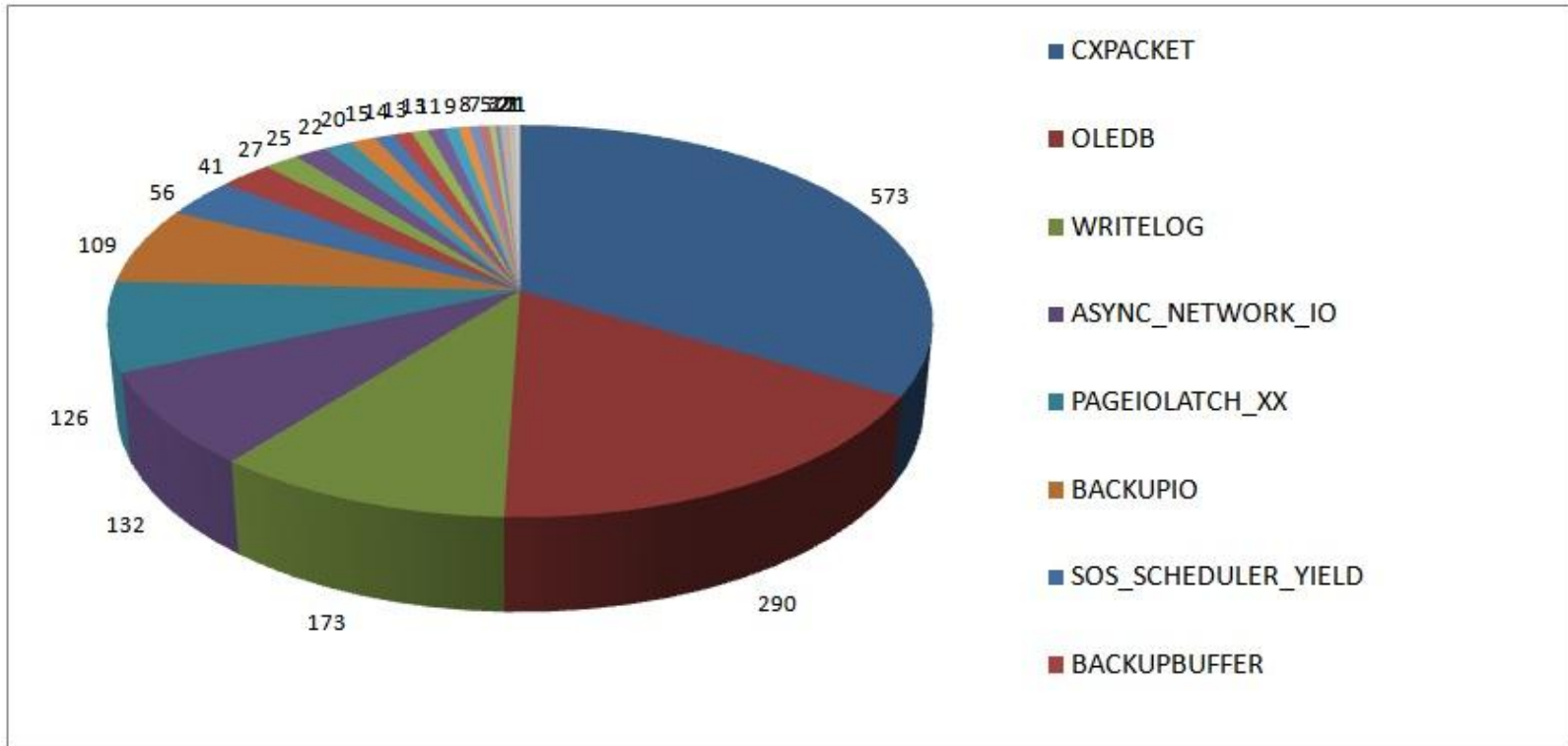
- This DMV shows aggregated wait statistics for all wait types
 - Aggregated since the server started or the wait statistics were cleared
- Think of this as the 'what has happened in the past?' view of a server
- Some math is required to make the results useful
 - Calculating the resource wait time
 - Calculating the average times rather than the total times

What's Relevant?

- **An extremely important point to bear in mind is that waits ALWAYS occur inside SQL Server**
 - Look for actionable items and filter out things like background tasks
 - Filter out benign waits such as WAITFOR, LAZYWRITER_SLEEP
 - Look at the demo code to see what I mean
- **Need to identify the top, relevant waits and then drill in**
- **Example:**
 - 1000 waits for LCK_M_S
 - Is it a problem?
 - No, if that was over 8 hours, there were 10 million locks acquired, and total wait time for the LCK_M_S locks was only 50s altogether
 - Yes, if each wait was for 50s

Top Wait Types

- Survey results from 1700+ SQL Server instances across Internet



- Source: <http://www.sqlskills.com/blogs/paul/common-wait-stats-24-hours/> (<http://bit.ly/1n1m1lF> - capital i before the F)

PAGEIOLATCH_XX Wait and Solutions

- **Waiting for a data file page to be read from disk into memory**
 - Common modes to see are SH and EX
- **Do not assume the I/O subsystem or I/O path is the problem**
- **Further analysis:**
 - Determine which tables/indexes are being read
 - Analyze I/O subsystem latencies with sys.dm_io_virtual_file_stats and Avg Disk secs/Read performance counters
 - Move the affected data files to faster I/O subsystem?
 - Correlate with CXPACKET waits, suggesting parallel scans
 - Create appropriate nonclustered indexes to reduce scans?
 - Update statistics to allow efficient query plans?
 - Examine query plans for parallel scans and implicit conversions
 - Investigate buffer pool memory pressure and Page Life Expectancy
 - If data volume has increased, consider increasing memory

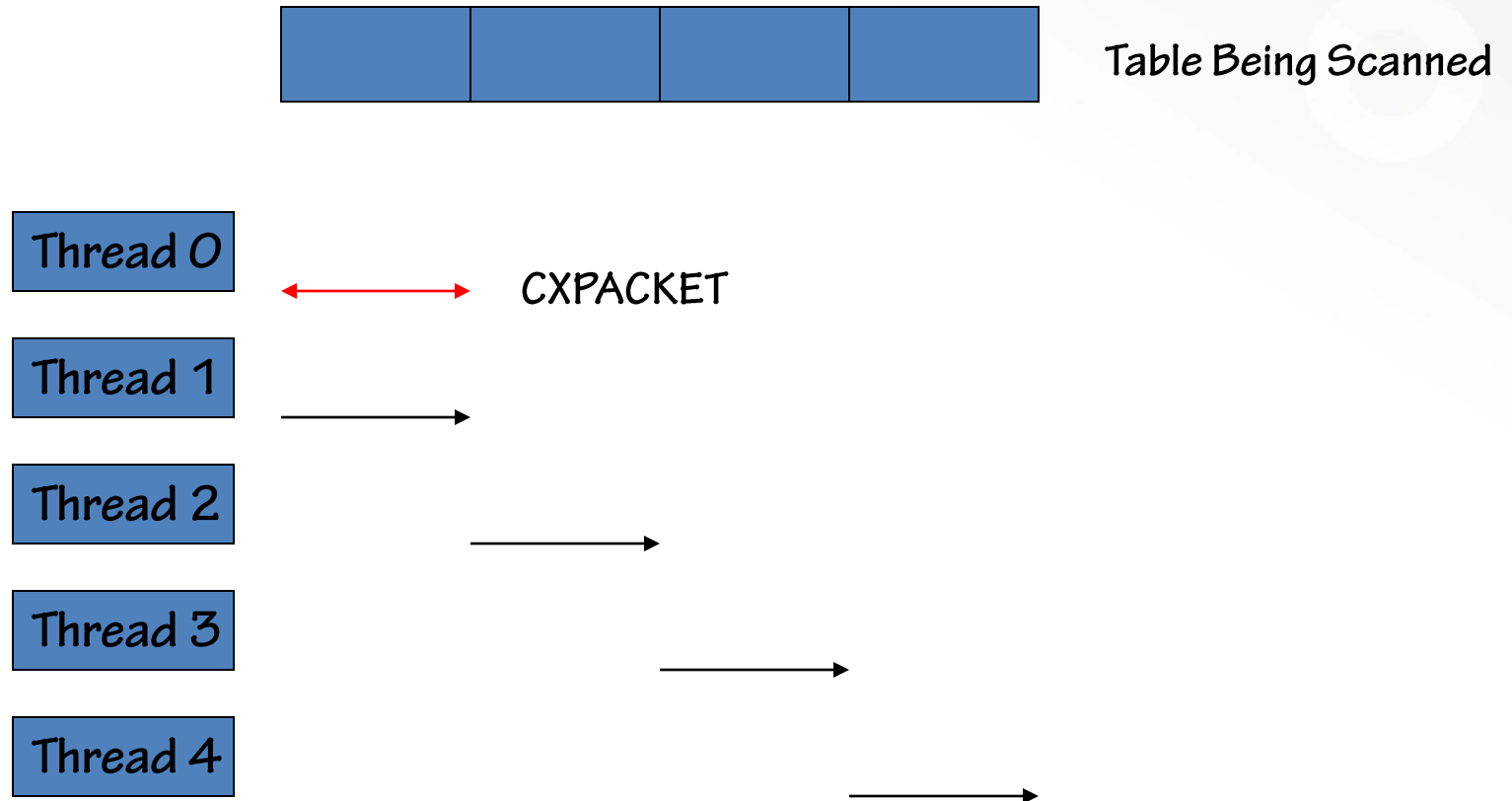
PAGELATCH_XX Wait and Solutions

- **Waiting for access to an in-memory data file page**
 - Common modes to see are SH and EX
- **Do not confuse these with PAGEIOLATCH_XX waits**
- **Does not mean add more memory or I/O capacity**
- **Further analysis:**
 - Determine the page(s) that the thread is waiting for access to
 - Classic tempdb contention?
 - Add more tempdb data files
 - Enable trace flag 1118
 - Reduce temp table usage
 - Analyze the table and index structures involved
 - Excessive page splits occurring in indexes
 - Insert-point hotspot in a clustered index with an ever-increasing key

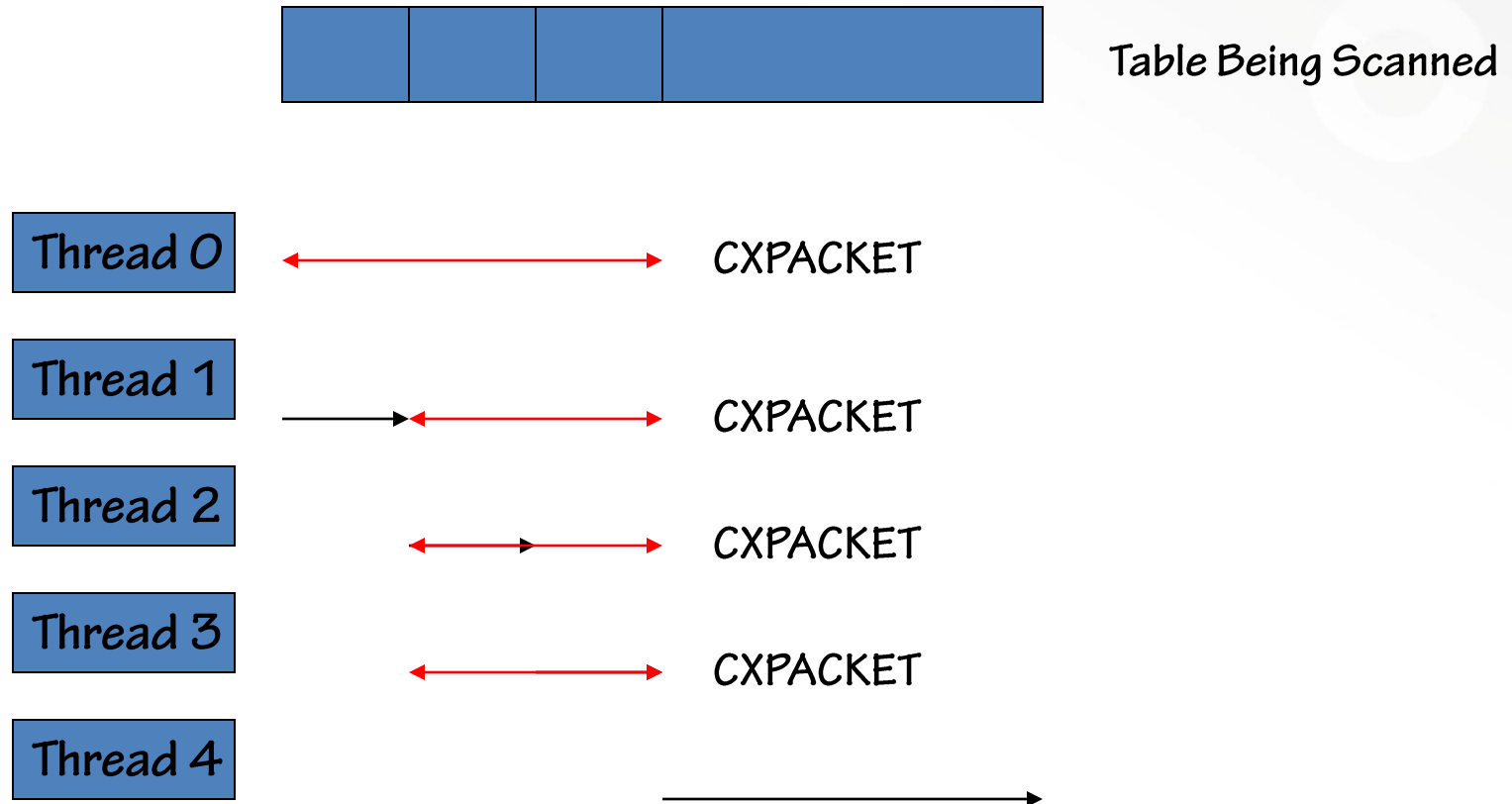
CXPACKET Wait Explanation

- **What does it mean:**
 - Parallel operations are taking place
 - Accumulating very fast implies skewed work distribution amongst threads or one of the workers is being blocked by something
- **Avoid knee-jerk response:**
 - Do not set server-wide MAXDOP to 1, disabling parallelism
- **Further analysis:**
 - Correlation with PAGEIOLATCH_SH waits? Implies large scans
 - Also may see with ACCESS_METHODS_DATASET_PARENT latch or ACCESS_METHODS_SCAN_RANGE_GENERATOR latch
 - Examine query plans of requests that are accruing CXPACKET waits to see if the query plans make sense for the query being performed
 - What is the wait type of the parallel thread that is taking too long? (i.e. the thread that does not have CXPACKET as its wait type)

CXPACKET Wait Example (1)



CXPACKET Wait Example (2)



CXPACKET Wait Solutions

- **Possible root-causes:**

- Just parallelism occurring (e.g. <http://bit.ly/1kPymkZ> from CSS)
- Table scans being performed because of missing nonclustered indexes or incorrect query plan
- Out-of-date statistics causing skewed work distribution

- **If there is actually a problem:**

- Make sure statistics are up-to-date and appropriate indexes exist
- Consider MAXDOP for the query
- Consider MAXDOP = physical cores per NUMA node
- Consider MAXDOP for the instance, but beware of mixed workloads
- Consider using Resource Governor for MAX_DOP
- Consider setting 'cost threshold for parallelism' higher than the query cost shown in the execution plan

Demo

Parallelism

ASYNC_NETWORK_IO Wait

- **What does it mean:**
 - SQL Server is waiting for a client to acknowledge receipt of sent data
- **Avoid knee-jerk response:**
 - Do not assume that the problem is network latency
 - It is a network delay as far as SQL Server is concerned though
- **Further analysis:**
 - Analyze client application code
 - Analyze network latencies
- **Possible root-causes and solutions:**
 - Nearly always a poorly-coded application that is processing results one record at a time (RBAR = Row-By-agonizing-Row)
 - Very easy to demonstrate using a large query and SQL Server Management Studio running on the same machine as SQL Server
 - Otherwise look for network hardware issues, incorrect duplex settings, or TCP chimney offload problems (see <http://bit.ly/aPzoAx>)

Summary: Methodology

- **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` (see my blog or the Pluralsight course)
- **Avoid temptation to knee-jerk and equate symptoms with root-cause**
- **Gather further information from relevant sources to pin-point problems**
 - DMVs, query plans, performance counters, code analysis

Resources

- **Brand-new whitepaper:**

- SQL Server Performance Tuning Using Wait Statistics: A Beginners Guide
 - <http://www.sqlskills.com/help/sql-server-performance-tuning-using-wait-statistics/>

- **Older whitepapers:**

- Performance Tuning Using Waits and Queues
- Diagnosing and Resolving Latch Contention on SQL Server
- Diagnosing and Resolving Spinlock Contention on SQL Server
 - Gnarly links – see top of our whitepapers page at <http://bit.ly/19j0cOd> (zero then oh)

- **Blog post categories**

- <http://www.sqlskills.com/blogs/paul/category/wait-stats/>
- <http://www.sqlskills.com/blogs/paul/category/latches/>
- <http://www.sqlskills.com/blogs/paul/category/spinlocks/>

Thank you!

