SQLskills
PASS Data Architecture Virtual UG
January 28th 2015

Performance Troubleshooting Using Wait Statistics

Paul S. Randal

Paul@SQLskills.com

SQLskills
immerse yourself in sql server

- **Team of world-renowned SQL Server experts:**

  - Paul S. Randal (@PaulRandal)
  - Kimberly L. Tripp (@KimberlyLTripp)
  - Glenn Berry (@GlennAlanBerry)
  - Erin Stellato (@ErinStellato)
  - Jonathan Kehayias (@SQLPoolBoy)
  - Tim Radney (@TRadney)

- **Instructor-led training: Immersion Events (US, UK, and 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**
  - https://www.sqlskills.com/Insider

2015 Immersion Events

- **Classes in Chicago, Bellevue (WA), London, Dublin, Sydney (Australia)**
  - IE0: Immersion Event for Junior/Accidental DBA
  - IEPTO1/2: Immersion Events on Performance Tuning – Parts 1 and 2
  - IEHADR: Immersion Event on High Availability and Disaster Recovery
  - IEBI: Immersion Event on Business Intelligence
- **In-depth, instructor-led, technical training for SQL Server**
- **Here's our topic list for IEPTO2: Immersion Event on Performance Tuning, Part 2**

SQL Server I/O	I/O Concepts for DBAs	SANs for DBAs
SQLOS Scheduling and CPU Performance Tuning	SQLOS Memory Management and Memory Performance Tuning	Data Collection and Baselining
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	

- **For more information: https://www.sqlskills.com/training/**

Author/Instructor:
Paul S. Randal

- **Consultant/Trainer/Speaker/Author**
- **CEO and Owner, SQLskills.com**
  - Email: Paul@SQLskills.com
  - Blog: https://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…)**
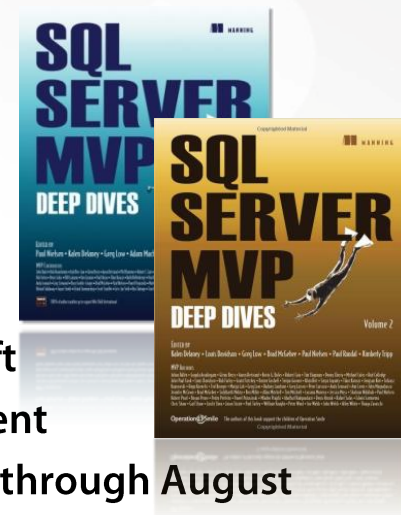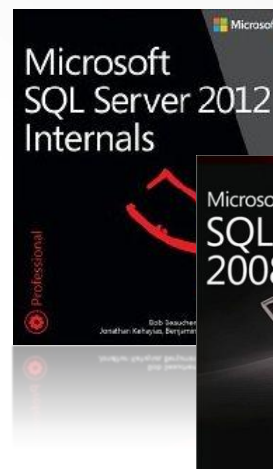
pluralsight ▶

- **Email paul@SQLskills.com with the subject line: User Group Pluralsight code to get a FREE (no catches, no credit card) 30-day trial of our 120+ hours of SQLskills content on Pluralsight**

- **For example:**
  - http://www.pluralsight.com/courses/sqlserver-logging
    - 7 hours on logging, recovery, and the transaction log (Paul)
  - http://www.pluralsight.com/training/Courses/TableOfContents/sqlserver-waits
    - 4.5 hours on waits, latches, spinlocks (Paul)
  - http://www.pluralsight.com/courses/sqlserver-optimizing-stored-procedure-performance
    - 7 hours on stored procedure performance tuning (Kimberly)

Overview

- **Very common to see 'knee-jerk' performance tuning where someone jumps to a conclusion based on superficial analysis of performance data**

- **Interpreting wait statistics is not hard, but needs practice**

- **We're going to cover**
  - Introduction
  - Thread lifecycle
  - Waits and wait times
  - DMVs
  - Some common wait types

Don't Assume Symptom = Root Cause

- **Performance troubleshooting is not an exact science**
  - The same symptoms can result from many root causes
- **For example, how many different things could cause I/O latencies?**
  - Overloaded/incorrectly-configured I/O subsystem
  - Synchronous I/O-subsystem mirroring
  - Buffer pool memory pressure
    - From plan cache bloat
    - From external Windows pressure
    - From an ad hoc query
    - From an inefficient query plan
  - Network latency
  - And more…

Interpreting the Data

- **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)

Thread Scheduling

- **SQL Server performs its own thread scheduling**
  - Called non-preemptive scheduling
  - More efficient (for SQL Server) than relying on Windows scheduling
  - Performed by the SQLOS layer of the Storage Engine
- **Each processor core (whether logical or physical) has a scheduler**
  - A scheduler is responsible for managing the execution of work by threads
  - Schedulers exist for user threads and for internal operations
  - Use the sys.dm_os_schedulers DMV to view schedulers
- **When SQL Server has to call out to the OS, it must switch the calling thread to preemptive mode so the OS can interrupt it if necessary**

Components of a Scheduler

- All schedulers are composed of three 'parts'
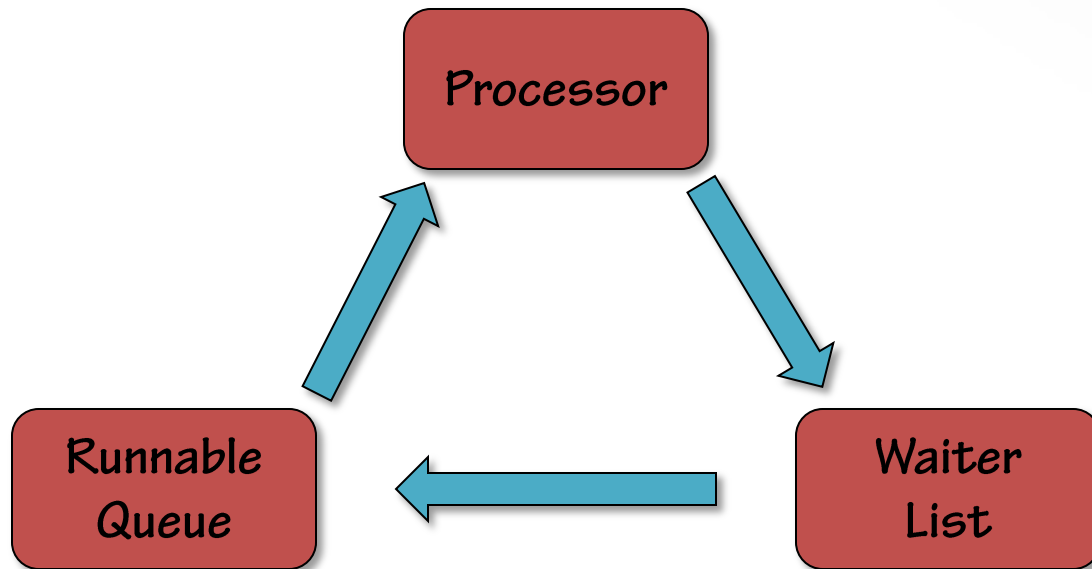
- Threads transition around these parts until their work is complete
- One scheduler per processor core SQL Server can use

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**

Transition: RUNNING to SUSPENDED

- **A thread continues executing on the processor until it must wait for a resource to become available**
    - The thread's state changes from RUNNING to SUSPENDED
    - The thread moves to the Waiter List
   - This process is called being 'suspended'

Processor

SPID 60: RUNNING TCH_SH

Runnable Queue

Waiter List
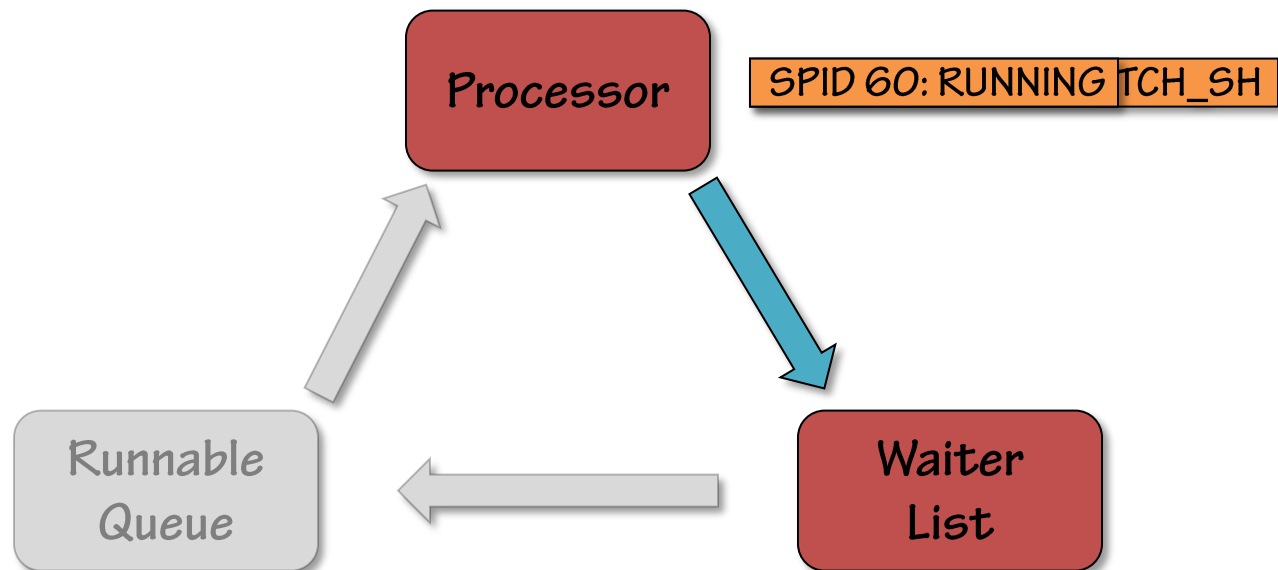
Transition: SUSPENDED to RUNNABLE

- **A thread continues to wait until it is told that the resource is available**
  - The thread's state changes from SUSPENDED to RUNNABLE
  - The thread moves to the bottom of the Runnable Queue
  - This process is called being 'signaled'

Processor

Runnable Queue

Waiter List

SPID 60: PAGEIOLATCH_SH
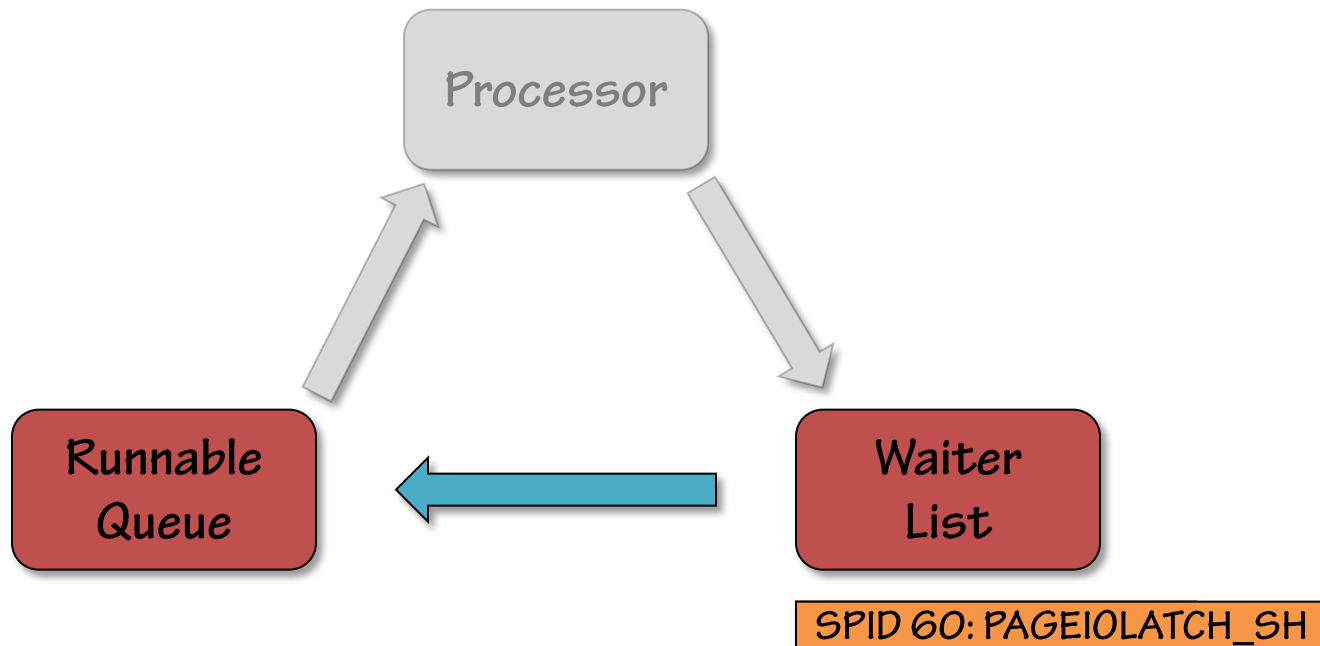
# Transition: RUNNABLE to RUNNING

- **The thread waits on the Runnable Queue until it reaches the top and the processor becomes available**
  - The thread's state changes from RUNNABLE to RUNNING
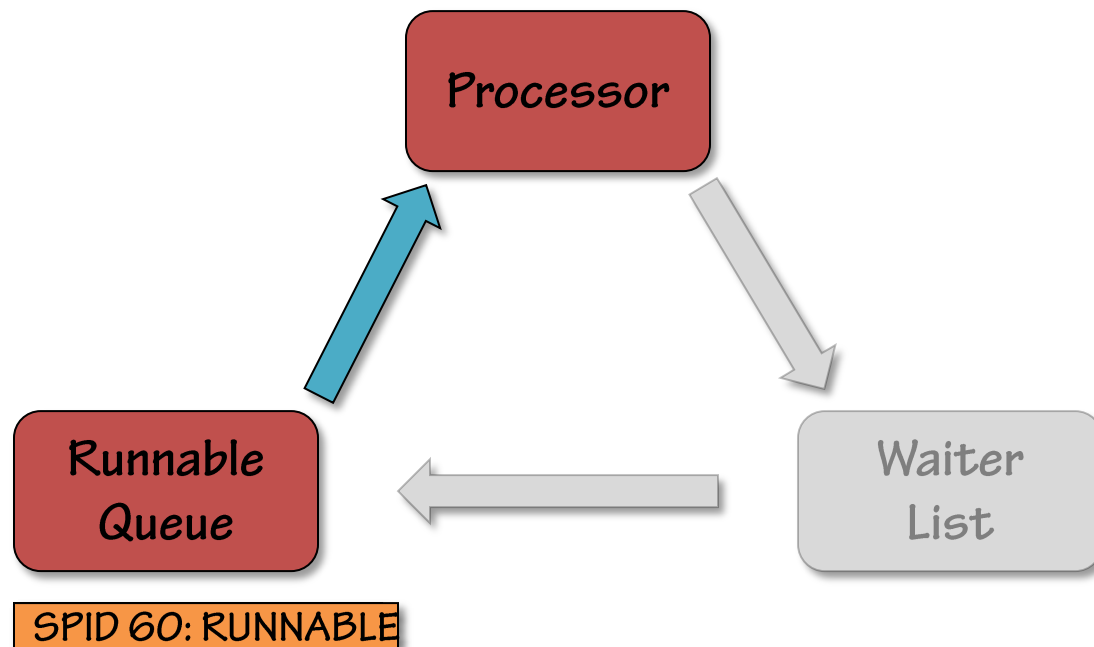
Wait Times Definition (1)

- **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**

Wait Times Definition (2)

Processor

SPID 60: RUNNING ED

Signal Wait Time

Resource Wait Time

Runnable
Queue

Waiter
List

SPID 60: RUNNABLE

SPID 60: SUSPENDED
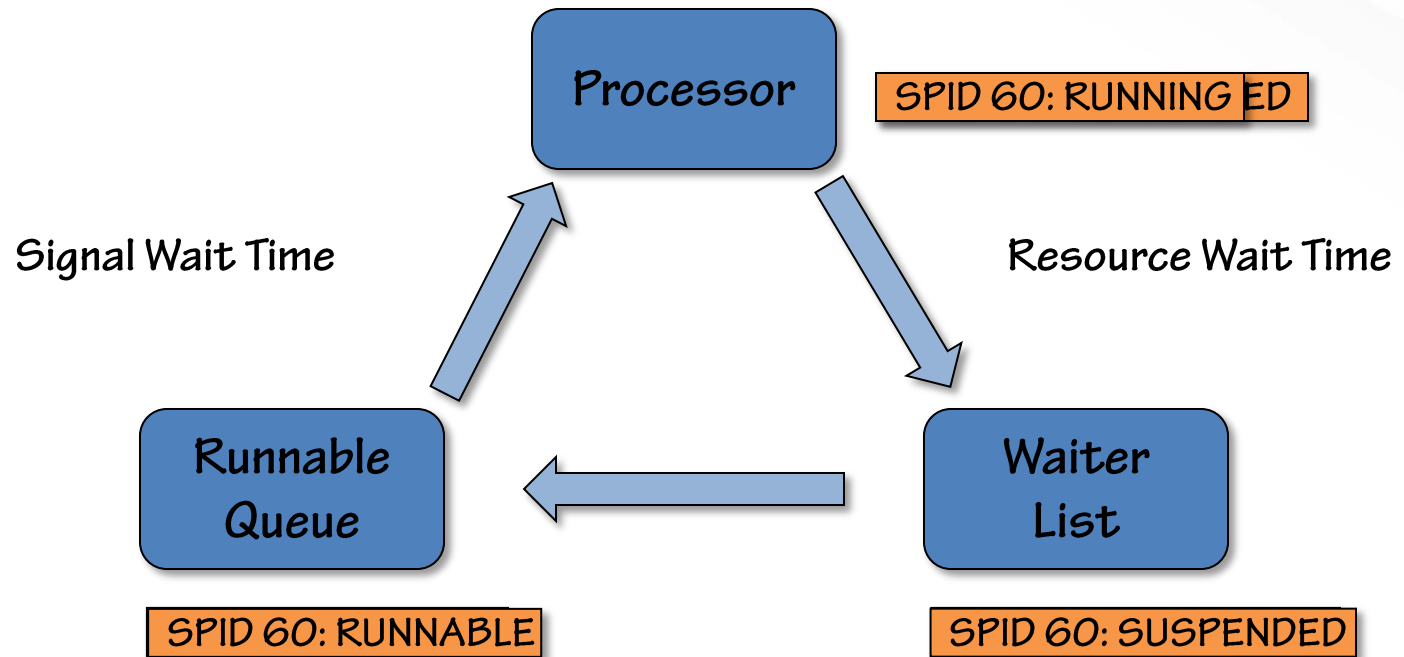
Wait Time = Resource Wait Time + Signal Wait Time

sys.dm_os_waiting_tasks DMV

- **This DMV shows all threads that are currently suspended**
- **Think of it as the 'what is happening right now?' view of a server**
- **Most useful information this DMV provides:**
  - Session ID and execution context ID of each thread
  - Wait type for each suspended thread
  - Description of the resource for some wait types
    - E.g. for locking wait types, the lock level and resource is described
  - Wait time for each suspended thread
  - If the thread is blocked by another thread, the ID of the blocking thread
    - Useful to find what's at the head of a blocking chain
    - Can show non-intuitive patterns
- **Usually very first thing to run when approaching a 'slow' server**
  - The data is more useful when joined with other DMV results
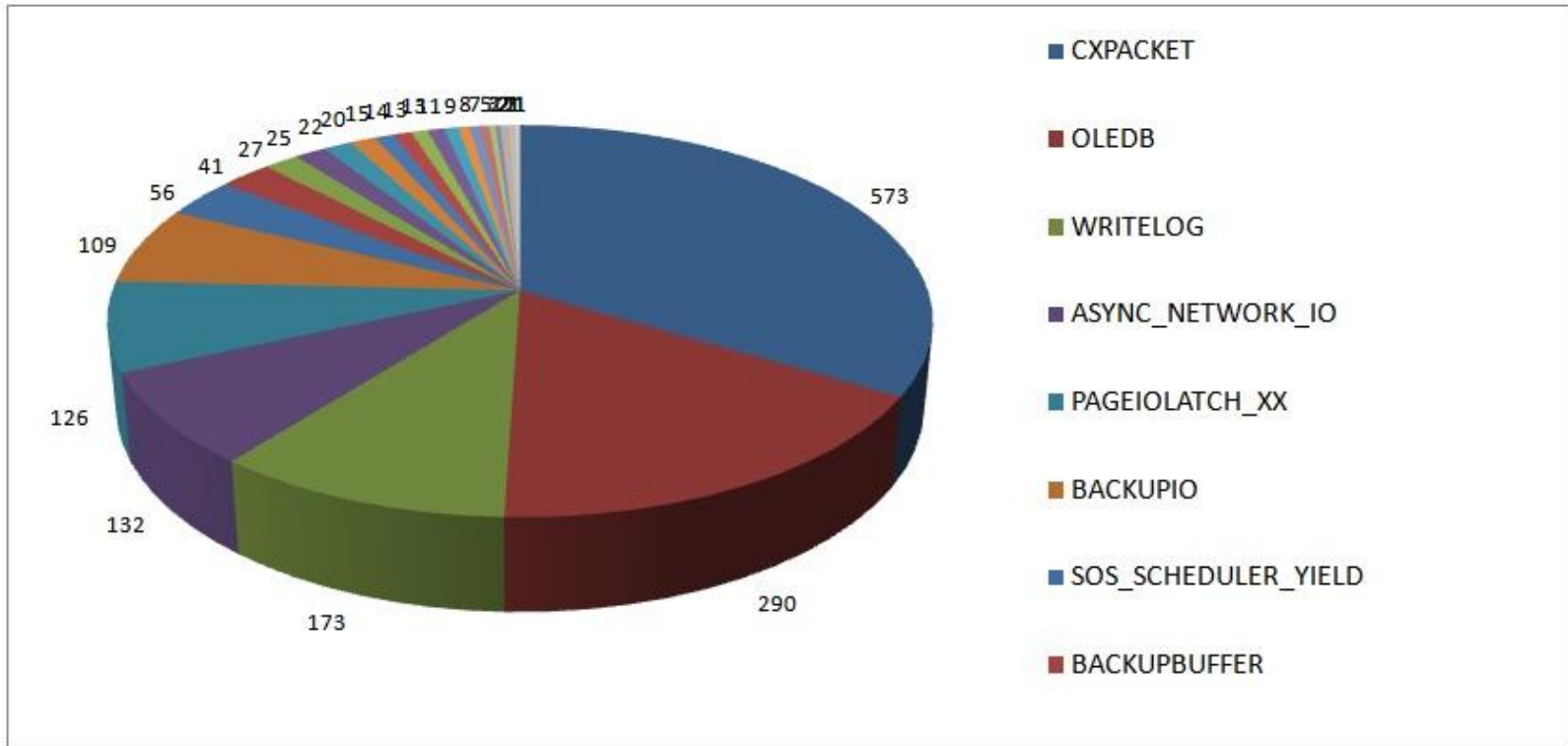
sys.dm_os_wait_stats DMV

- **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**
- **This DMV provides:**
  - The name of each wait type
  - The number of times a wait has been for this wait type
  - The aggregate overall wait time for all waits for this wait type
  - The maximum wait time of any wait for this wait type
  - The aggregate signal wait time for all waits for this wait type
- **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/1n1m1IF - 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

LCK_M_XX Wait and Solutions

- **A thread is waiting for a lock that cannot be granted because another thread is holding an incompatible lock**

- **Do not assume that locking is the root cause**

- **Further analysis:**
  - Follow the blocking chain using sys.dm_os_waiting_tasks to see what the lead blocking thread is waiting for
  - Use blocked process report to capture info on queries waiting for locks
    - Michael Swart's blog post (http://bit.ly/ki3bYI)
  - Lock escalation from a large update or table scan?
    - Consider a different indexing strategy to use NC index seeks
    - Consider using snapshot isolation, a different isolation level, or locking hints
  - Something preventing a transaction from releasing its locks quickly?
    - E.g. synchronous DBM/AG, DTC, or log throughput problems

Demo

Page latches and locks

WRITELOG Wait

- **What does it mean:**
  - Waiting for a transaction log block buffer to flush to disk
- **Avoid knee-jerk response:**
  - Do not assume that the transaction log file I/O system has a problem (although this is often the case)
  - Do not create additional transaction log files
- **Further analysis:**
  - Correlate WRITELOG wait time with I/O subsystem latency using sys.dm_io_virtual_file_stats
    - Look for LOGBUFFER waits, showing internal contention for log buffers
  - Look at average disk write queue length for log drive
    - If constantly 31/32 (111/112 on SQL 2012+) then the internal limit has been reached for outstanding transaction log writes for a single database
  - Look at average size of transactions
  - Investigate whether frequent page splits are occurring
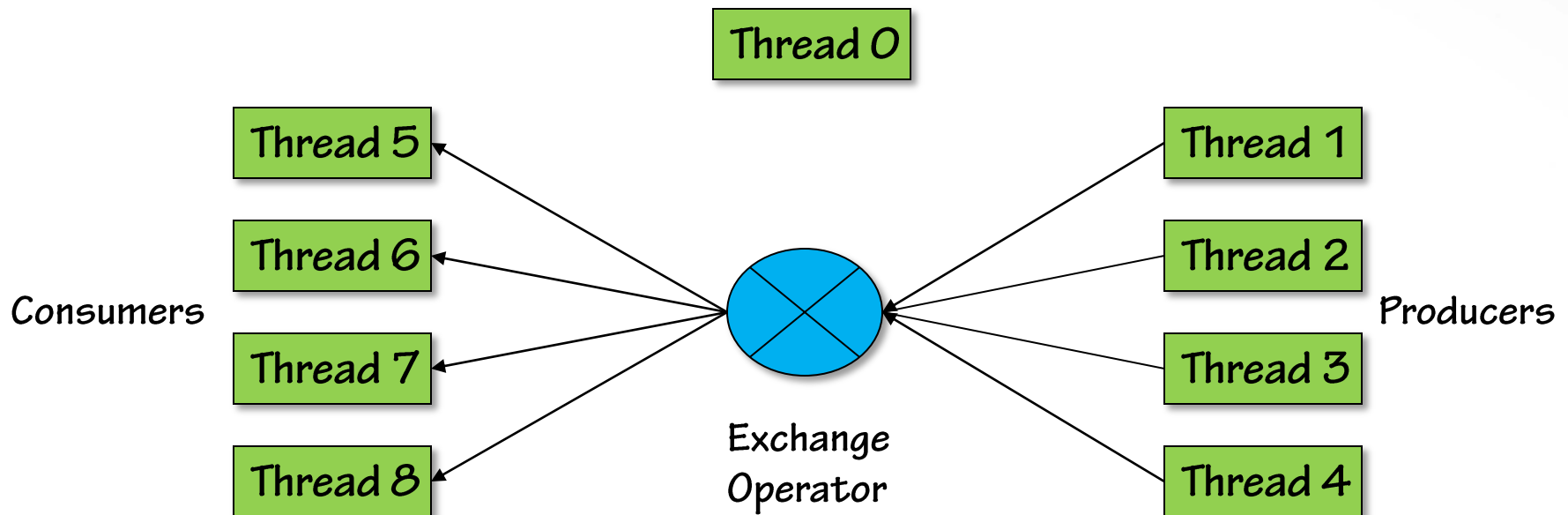
WRITELOG Wait Solutions

- **Move the log to a faster I/O subsystem**
- **Increase the size of transactions to prevent many minimum-size log block flushes to disk**
- **Remove unused nonclustered indexes to reduce logging overhead from maintaining unused indexes during DML operations**
- **Change index keys or introduce FILLFACTORs to reduce page splits**
- **Investigate whether synchronous database mirroring/AGs/SAN replication is introducing delays**
- **Potentially split the workload over multiple databases or servers**
- **SQL Server 2014**
  - Delayed durability
  - In-memory OLTP

Demo

Slow transaction log
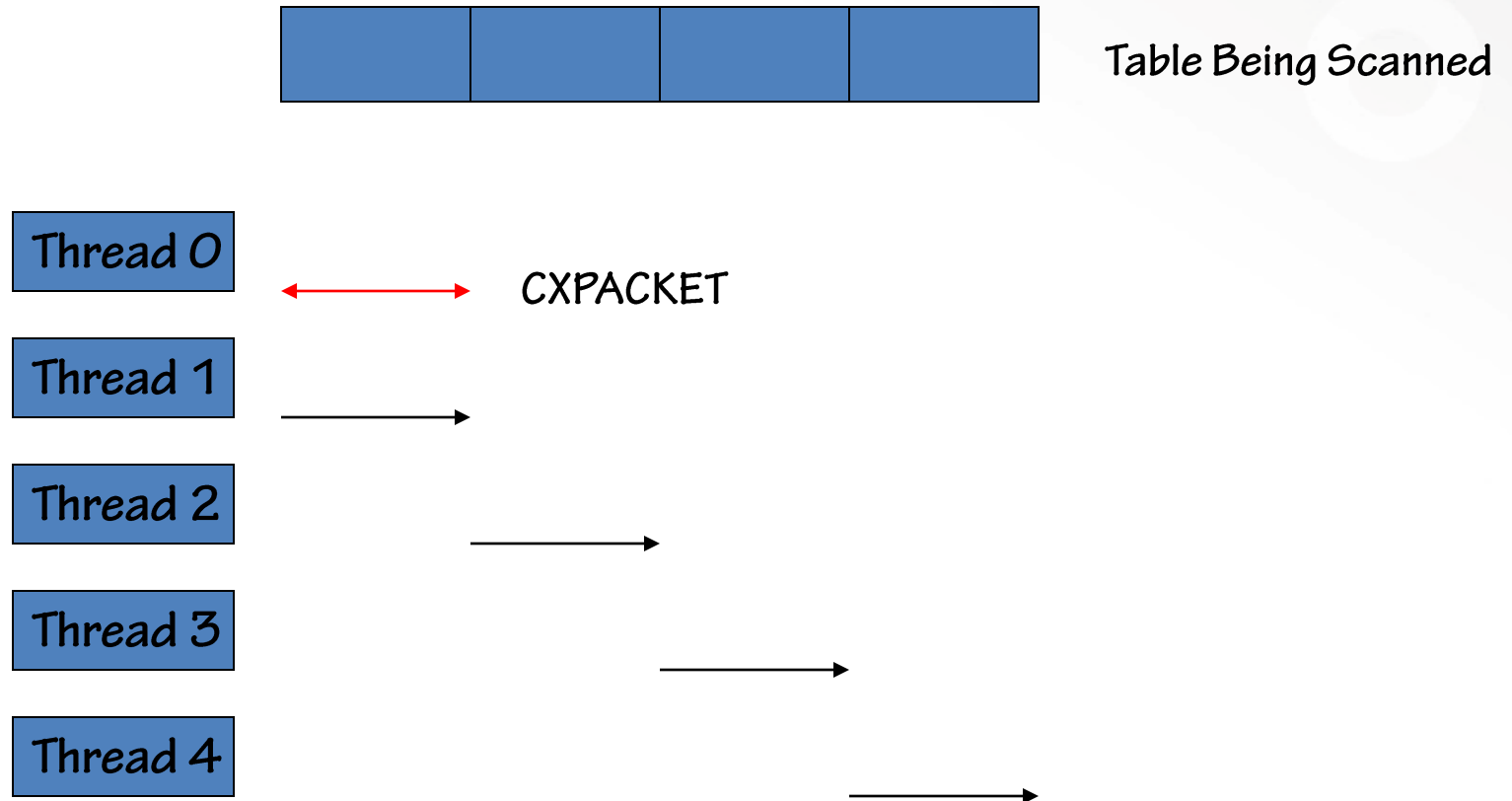
Parallel Threads Example

- **As part of a query plan, you may see the** **operator, for example**
- **This is a Repartition Streams operation**
  - Uses producer and consumer threads, plus a control thread
- **For a degree-of-parallelism = 4 operation, the threads would look like:**

Thread 0

Thread 5

Thread 6

Consumers

Thread 7

Thread 8

Thread 1

Thread 2

Producers

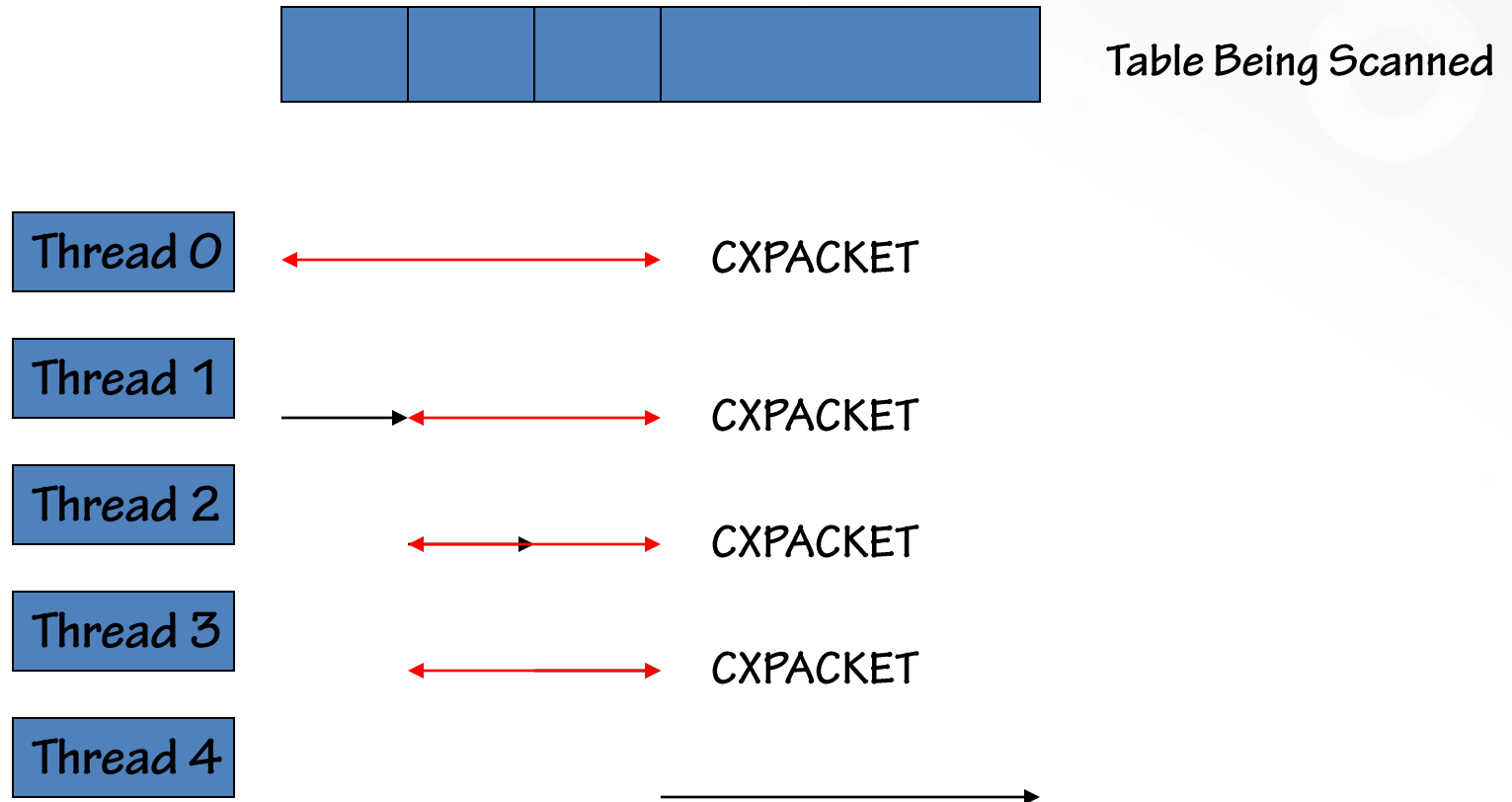Thread 3

Thread 4

Exchange
Operator

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)

Table Being Scanned

Thread 0

CXPACKET

Thread 1

Thread 2

Thread 3

Thread 4

CXPACKET Wait Example (2)

Table Being Scanned

Thread 0 — CXPACKET

Thread 1 — CXPACKET

Thread 2 — CXPACKET

Thread 3 — CXPACKET

Thread 4

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)

OLEDB Wait

- **What does it mean:**
  - The OLE DB mechanism is being used
- **Avoid knee-jerk response:**
  - Do not assume that linked servers are being used
- **Further analysis:**
  - What are the queries doing that are waiting for OLEDB?
  - If linked servers are being used, what is causing the delay on the linked server?
- **Possible root-causes:**
  - DBCC CHECKDB and related commands use OLE DB internally
  - Many DMVs use OLE DB internally so it could be a third-party monitoring tool that is repeatedly calling DMVs (especially if they're very short waits)
  - Poor performance of a linked server

PREEMPTIVE_OS_XX Waits

- **What does it mean:**
  - A thread has called out to the OS
  - Threads must switch to preemptive mode when doing so
  - Note that the thread status will be RUNNING instead of SUSPENDED

- **Further analysis:**
  - 194 PREEMPTIVE_OS_XX waits in SQL Server 2012 and 2014
  - These waits are very minimally and poorly documented
  - To determine what SQL Server is asking the OS to do, usually can remove the PREEMPTIVE_OS_ prefix and search in MSDN for the remainder of the wait type, as it will be the name of a Windows API

- **Possible root-causes and solutions:**
  - Depends on the wait type
  - For instance, increasing PREEMPTIVE_OS_CREATEFILE waits occur when using FILESTREAM on an incorrectly prepared NTFS volume

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!