SQL Server: Performance Troubleshooting Using Wait Statistics

Module 3: Waits

Paul S. Randal
Paul@SQLskills.com

pluralsight
hardcore developer training

Introduction

- **Before using wait statistics for troubleshooting, it's vital to understand what wait statistics actually mean**

- **In this module we'll cover:**
  - What waits and queues are
  - The various components of wait times
  - The various DMVs to examine wait statistics
  - Scripts to present wait statistics data appropriately
  - Using Extended Events to view wait statistics

What are Waits?

- **The term 'wait' means that a thread running on a processor cannot proceed because a resource it requires is unavailable**
- **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)
    - Discussed in Module 4
- **When a thread has to wait:**
  - It moves to the Waiter List (described in Module 2)
  - Its state changes from RUNNING to SUSPENDED
  - It remains on the Waiter List until the resource is available

What are Queues?

- **The term 'queues' is a generic term describing what is preventing a resource being available to the thread that is waiting for it**
  - This does not mean that all resources are held in actual queue structures
- **Example queues:**
  - For a LCK_M_XX wait, another thread holding an incompatible lock
  - For a PAGEIOLATCH_XX wait, the I/O subsystem needs to complete the I/O
  - For a CXPACKET wait, another thread needs to complete its portion of work
  - For a LATCH_XX wait, another thread holding an incompatible latch
- **Queues are investigated using DMVs, performance counters, and other tools**

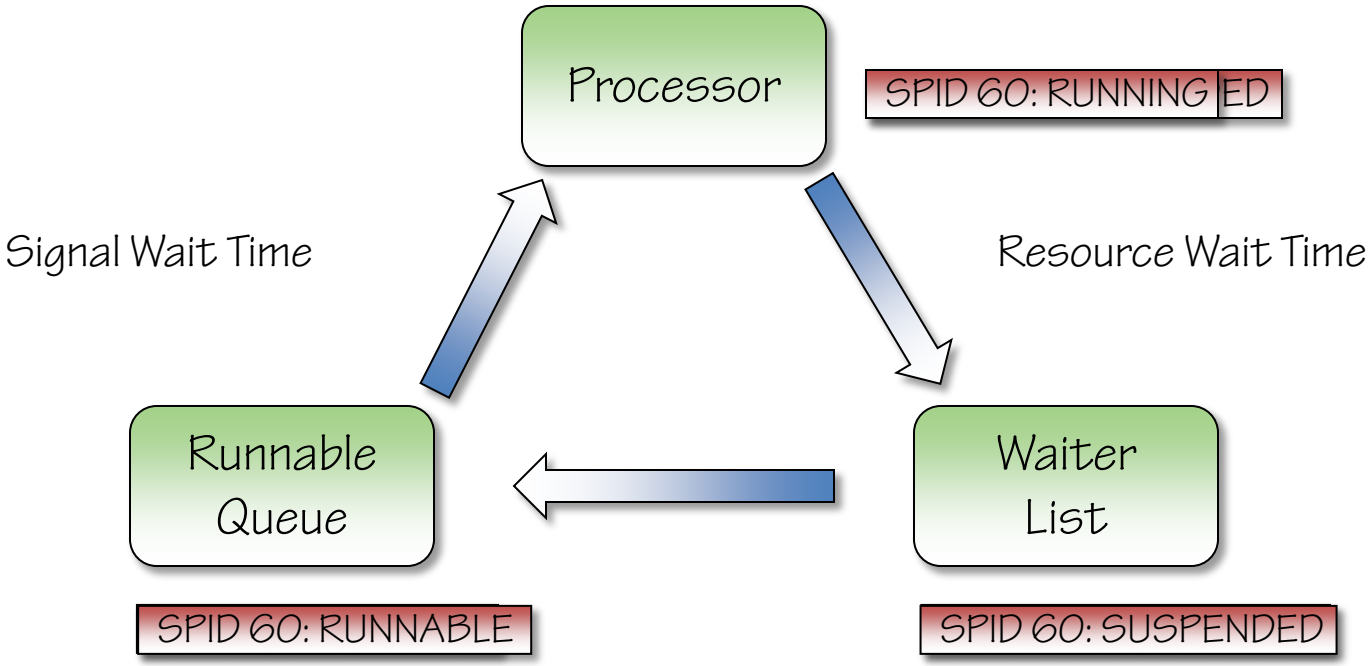
Waits and Queues Methodology

- **'Waits and queues' was a name coined by Tom Davidson in 2005**
  - Described in the whitepaper *Performance Tuning Using Waits and Queues*
    - The 'bible' of using wait statistics
    - It can be downloaded from Microsoft at http://bit.ly/aUh6S
- **Very powerful method to get initial direction on a problem**
  - Avoid flailing and investigating the wrong problem
  - Can also show problems that are not obvious
- **Most commercial performance monitoring tools capture and show wait statistics**
  - Many free tools also do this, such as the popular Who Is Active tool
- **Various releases of SQL Server have provided wait statistics views**
  - SQL Server 2005 had Performance Dashboard reports
  - SQL Server 2008 onwards has the Management Data Warehouse

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**

Wait Times Definition (2)



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 the 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

Filtering Benign Waits

- **An extremely important point to bear in mind is that waits ALWAYS occur inside SQL Server**
  - I.e. just because waits exist does not mean there is a performance problem
- **Rather than looking at all waits, most useful is to focus on highly prevalent wait types**
  - More processing of the sys.dm_os_wait_stats results is required
  - Common method is to show the top 95% of all waits by wait time
- **Some wait types are almost always benign and can be safely ignored**
  - Some have pathological, very rare cases where they can be problematic
- **For example, the WAITFOR wait type**
  - Only occurs when a WAITFOR DELAY statement is executed
  - When filtering the top 95% of waits by total wait time, not filtering out this wait can badly skew the results

Storing Wait Statistics

- **Capturing wait statistics information over time allows:**
  - Trending
  - Point-in-time analysis to see when a problem started to occur

- **Simple method:**
  - Use code from the sys.dm_os_wait_stats demo and add a GETDATE () call
  - Store the results in a table
  - Create SQL Agent job to capture the wait statistics every hour or so
  - Create another SQL Agent job to purge wait statistics older than a month

Clearing Wait Statistics

- Clearing the aggregated wait statistics can be done at any time using the code below:

```
DBCC SQLPERF ('sys.dm_os_wait_stats', CLEAR);
GO
```

- Clearing the wait statistics allows the effect of a workload change to be measured against previous wait statistics
- Be careful if you are taking periodic snapshots of wait statistics as this will invalidate your series of snapshots

Using Extended Events

- **Extended Events were added in SQL Server 2008 to all Editions**
  - Very lightweight mechanism for tracing activity in SQL Server
  - However, mis-configuration can make Extended Events very expensive
- **When a wait starts and ends, the sqlos.wait_info event fires**
  - Captures similar information to sys.dm_os_wait_stats
- **When a preemptive wait starts and ends, the sqlos.wait_info_external event fires**
  - Used when a thread is waiting for a call out to the OS and has to switch from non-preemptive to preemptive scheduling
- **Using the Extended Events system allows:**
  - Capturing of all wait types for a single operation
  - Monitoring for specific wait types occurring
  - Advanced analysis of SQL Server internals

Summary

- **Analysis of wait statistics can provide a quick way to find the root cause of a performance problem**
  - Also can provide deep insight into SQL Server's internals
- **Waits always occur, so it is imperative to focus on the most prevalent waits that are not benign**
- **The two DMVs to use for wait statistics analysis are:**
  - sys.dm_os_waiting_tasks – what is happening right now?
  - sys.dm_os_wait_stats – what has happened in the past?

- **The next module will discuss the more advanced latch and spinlock statistics and how they can be useful**