

## Module 5

# Waits and Queues



## Overview

- Introduction
- Thread lifecycle
- DMVs
- Common wait types
  
- Doctor, my knee hurts...

## Wait Statistics

- A 'wait' is when a running thread decides it needs a resource to be able to continue
- Example resources:
  - A lock on a page (e.g. LCK\_M\_S wait)
  - A data page read into memory from disk (e.g. PAGEIOLATCH wait)
  - Results from part of a parallel query (e.g. CXPACKET wait)
  - Writing into the version store (e.g. a LATCH\_EX wait that is for the APPEND\_ONLY\_STORAGE\_FIRST\_ALLOC latch class)
- While waiting, the thread cannot continue
- Waits are investigated using DMVs (and other tools)

## Queues

- A 'queue' is what prevents a resource from being available – i.e. it contributes to the wait time
- Example 'queues':
  - Another thread holding a desired lock
  - Another thread not yet completed it's portion of a parallel query
  - CPU contention
  - I/O subsystem contention
- Queues are investigated using performance counters and DMVs
  - sys.dm\_os\_performance\_counters

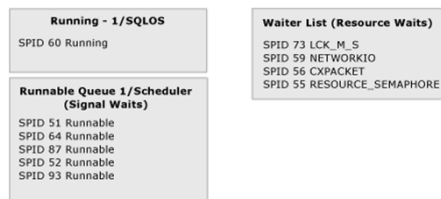
## Waits and Queues Together

- Very often someone investigating a performance problem doesn't know where to start looking
  - 'We have a sudden degradation in performance'
  - 'Feels like the server is slower than it was'
- The 'wait and queues' methodology uses the information that SQL Server already has to narrow down the bottleneck and remove irrelevant avenues of investigation
- Can often identify non-intuitive problems
- Monitoring over time provides a very powerful way to identify causes of performance changes
  - Easy to see what has changed (as far as wait statistics are concerned)

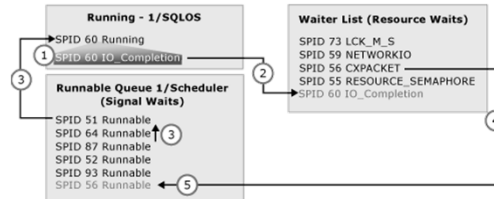
## Thread Execution (1)

- SQL Server executes a query using one or more threads called workers
- Each thread runs on a scheduler, one scheduler per processor core
- A thread can be:
  - RUNNING: active on the CPU
  - SUSPENDED: waiting for a resource
  - RUNNABLE: waiting to get back on the CPU

➤ Example:

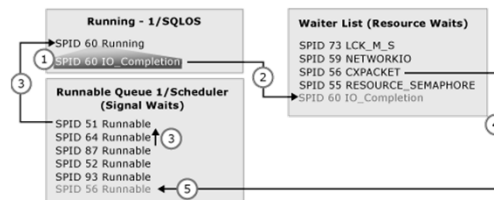


## Thread Execution (2)



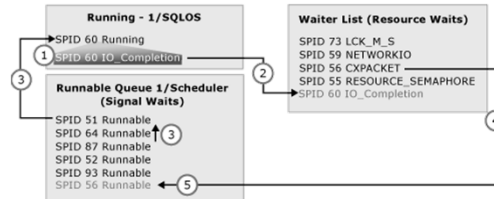
- Explaining 1 and 2...
- SPID 60 requires a non-data page from disk, so stops running and is placed on the Waiter List
- The Waiter List is not an ordered queue, it's just a list of the threads waiting for a resource
- In this example, SPID 60 changes state from RUNNING to SUSPENDED

## Thread Execution (3)



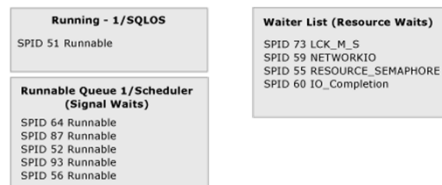
- Explaining 3...
- When SPID 60 moves to the Waiter List, the scheduler takes the SPID at the top of the RUNNABLE queue and makes it run
- The Runnable Queue IS an ordered list, but it may not be FIFO (for instance if Resource Governor is running)
- In this example SPID 51 changes state from RUNNABLE to RUNNING

## Thread Execution (4)



- Explaining 4 and 5...
- Once a SPID is signaled that its resource wait is over, it moves to (usually the bottom of) the Runnable Queue
- In this example SPID 56 moves and changes state from SUSPENDED to RUNNABLE

## Thread Execution (5)



- How things look after SPIDs 60 and 51 change state
- Clockwise rotation from RUNNING to SUSPENDED to RUNNABLE to RUNNING continues until user request is complete

## Wait Times Defined

- 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 resource:
  - Known as 'resource wait time'
  - Time spent SUSPENDED on the waiting tasks queue
- Time spent waiting for CPU:
  - Known as 'signal wait time'
  - Time spent on the RUNNABLE queue after having been signaled that the resource is available
- Wait time = resource wait time + signal wait time
- Can think of query run time as sum of all wait times

## Relevant DMVs

- **sys.dm\_os\_wait\_stats**
  - Server-level aggregate of all wait types with cumulative and max wait times plus wait counts
  - `wait_time_ms` = total wait time for each type
  - `signal_wait_time_ms` = time spent waiting to move from RUNNABLE to RUNNING
  - $(\text{wait\_time\_ms} - \text{signal\_wait\_time\_ms})$  = resource wait time, time spent being SUSPENDED
- **sys.dm\_os\_waiting\_tasks**
  - List of tasks that are waiting for a resource
  - Can show non-intuitive patterns – 'blocking' is really WRITELOG waits
- Reset wait stats using:
  - `DBCC SQLPERF('sys.dm_os_wait_stats', CLEAR);`
  - Also reset whenever system restarts

## Latch Statistics

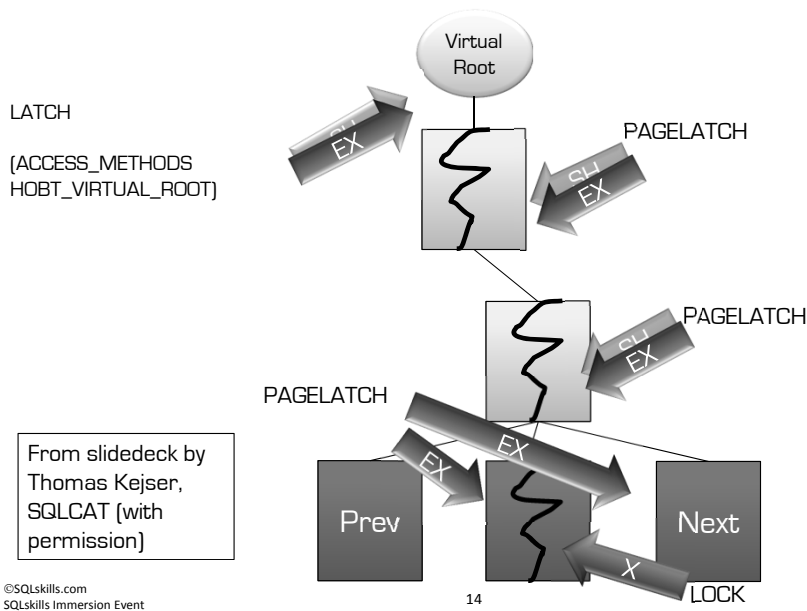
- `sys.dm_os_latch_stats`
- Very similar to `sys.dm_os_wait_stats` but gives wait statistics for latches broken down by all latch types
- Latches are like very lightweight locks, but are only held for the duration of the operation
  - E.g. FGCB\_ADD\_REMOVE
  - E.g. BUFFER
  - E.g. DBCC\_MULTIOBJECT\_SCANNER
    - Recent CHECKDB perf boost by fixing contention on this latch
- Looking into latch statistics is advanced and not done very often as it requires pretty deep internals knowledge to understand
- Reset wait stats using:
  - `DBCC SQLPERF('sys.dm_os_latch_stats', CLEAR);`
  - Also reset whenever system restarts

©SQLskills.com  
SQLskills Immersion Event

13

SQL

## B-Tree Root Split and Latches



©SQLskills.com  
SQLskills Immersion Event

14

SQL

## Extended Events

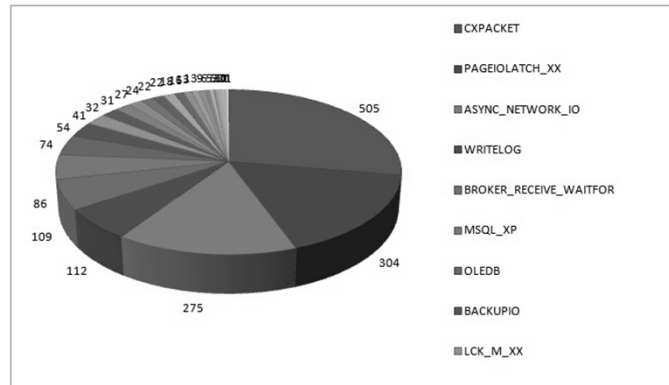
- Using the DMVs gives a view of what's happening across the whole server
- Using the `wait_info` event, it is possible to build an event session that captures all events for a single query to allow deeper analysis on a busy system
- We'll look at that in the next module...

## What's Relevant?

- Just because there are waits, does not mean they are the problem
  - Look for actionable items and filter out things like background tasks
  - 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
- If signal wait times are low, CPU pressure is not an issue
  - CPU pressure addressed in Module 9

## Top Wait Types

- Survey results from 1823 SQL Server instances across Internet



- Source: <http://www.sqlskills.com/BLOGS/PAUL/post/Wait-statistics-or-please-tell-me-where-it-hurts.aspx>

©SQLskills.com  
SQLskills Immersion Event

17

## Common Waits: PAGEIOLATCH\_XX

- Waiting for a page to be read in from disk
- Does NOT mean poor I/O subsystem performance
- Look at the type to get clues:
  - SH = page is going to be scanned
  - EX = page is going to be updated
- Consider:
  - Incorrect nonclustered indexes
  - Poor query plan
  - Not enough buffer pool memory for workload
  - I/O subsystem bottleneck
- Correlate with Avg. Disk secs/Read perfmon counter and sys.dm\_io\_virtual\_file\_stats

©SQLskills.com  
SQLskills Immersion Event

18

## Common Waits: PAGELATCH\_XX

- Contention for access to a page already in memory
- Does NOT mean poor I/O subsystem performance
- Look at the type to get clues:
  - SH = page is going to be scanned
  - EX = page is going to be updated
- Consider:
  - Tempdb requiring multiple files
  - Page splits from random inserts
  - Hot spot in a clustered index

## Common Waits: LCK\_M\_XX

- Waiting for a lock that is held in a conflicting mode by another thread
- Does NOT mean blocking is definitely the problem
- Any number of other things may be preventing the thread holding the lock from proceeding
  - E.g. network, disk I/O, log contention
- Look at the thread holding the lock
  - Use sys.dm\_os\_waiting\_tasks to find the head of the blocking chain
- If it \*is\* blocking that is the issue, consider indexes, snapshot isolation, code rewrite

## Demo

### Hotspots and page splits



## Common Waits: WRITELOG

- Waiting for log buffer flush to finish
- Usually always indicates I/O subsystem overloaded
  - Could be many log flushes from tiny transactions, along with log file I/O throttling discussed in Module 1
- However, as throughput gets higher and higher, effect of synchronization between writers and log flush manager can affect time
  - In that case, use sys.dm\_io\_virtual\_file\_stats
- Might see LOGBUFFER too
  - Waiting for space to write a log record because of maxed-out log writes

## Demo

### Slow transaction log



## Common Waits: CXPACKET

- Parallel operation where one thread takes longer than others to execute its portion of the task
- Does NOT mean set MAXDOP=1 for instance
- Could be:
  - Table scans being performed because of missing indexes
  - Incorrect query plan because of out-of-date statistics
- If these are actually a problem:
  - Consider MAXDOP for the query
  - Consider MAXDOP = schedulers/NUMA node
  - Consider MAXDOP for the instance, but beware of mixed-mode workloads
  - Consider using Resource Governor for MAXDOP
  - Consider setting cost threshold for parallelism higher

## Demo

### Parallelism



## Common Waits: PREEMPTIVE\_XX

- SQL Server does non-preemptive scheduling
  - 'Preempt' = 'interrupt' (i.e. Windows controls the scheduling rather than SQL Server)
- Must switch to preemptive for external calls, which means thread can be preempted
- 2005/2008 status is RUNNING instead of SUSPENDED
  - 2008 also shows which wait type
- Not documented very well
- Examples:
  - PREEMPTIVE\_OS\_GETFILESIZE
  - PREEMPTIVE\_OS\_FILEOPS
  - PREEMPTIVE\_OS\_CREATEFILE
  - All are expected wait type for FILESTREAM operations

## Common Waits: **ASYNC\_NETWORK\_IO**

- SQL Server is waiting for a client to acknowledge receipt of sent data
- Does NOT mean there's a network issue
- Far more likely that the client is processing its data inefficiently (e.g. asking for a many rows and then processing them RBAR – row-by-agonizing-row)
- Consider:
  - Client application architecture
  - Network issues

## Common Waits: **SOS\_SCHEDULER\_YIELD**

- A thread uses its entire 4ms quanta on the scheduler and then yields
- It goes to the bottom of the runnable queue as there's no need for it to be SUSPENDED (i.e. zero-length resource wait)
  - You won't see these in sys.dm\_os\_waiting\_tasks
- Your monitoring may miss these as they may also have zero or very small signal waits too
  - Common code to get top 95% of waits by wait time will skip these
- Consider:
  - Large scans occurring
  - Possibly excessive spinlock contention

## Demo

### Other wait types



## Real-Life Example (1)

- Car sales listing host service
  - Lots of dealers from across US hosted on one site
  - Set of listings for each dealer has same DealerID in very large Listing table
- System has big performance problems
  - Listing queries take a long time
  - Updates sometimes time out
- Analyze wait information...

## Real-Life Example (2)

- Majority of waits are:
  - CXPACKET wait = parallelism
  - PAGEIOLATCH\_SH wait = reading from disk
  - WRITELOG wait = \*possible\* I/O subsystem contention
    - Avg wait time > 20ms
- Possible problems:
  - Many queries doing parallel table scans?
  - I/O subsystem overloaded?
- Further investigation using Access Methods perf counters, missing index/index usage/plan cache DMVs showed many concurrent table scans of Listing table as queries are poorly written so plans not using NC indexes
- Also log file on same storage as data files, causing I/O subsystem contention (proven with sys.dm\_io\_virtual\_file\_stats)

©SQLskills.com  
SQLskills Immersion Event

31

SQL

## Summary of Use

- Take snapshots at regular intervals so you can tell what has changed over time
- Without series of snapshots, look at the top 3-4 waits
- Pay attention to average wait time
- Don't knee-jerk!

©SQLskills.com  
Designing for Performance and Availability

32

SQL

## Resources

- 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 [www.SQLskills.com/whitepapers.asp](http://www.SQLskills.com/whitepapers.asp)
- Blog: Wait statistics, or please tell me where it hurts
  - <http://www.sqlskills.com/BLOGS/PAUL/post/Wait-statistics-or-please-tell-me-where-it-hurts.aspx>
- Websites
  - Service Broker Wait Types – SQL SSB Team
    - [http://blogs.msdn.com/b/sql\\_service\\_broker/archive/2008/12/01/service-broker-wait-types.aspx](http://blogs.msdn.com/b/sql_service_broker/archive/2008/12/01/service-broker-wait-types.aspx)
  - PSS Wait Stats Repository
    - <http://blogs.msdn.com/b/psssql/archive/2009/11/03/the-sql-server-wait-type-repository.aspx>
  - sys.dm\_os\_wait\_stats
    - <http://technet.microsoft.com/en-us/library/ms179984.aspx>

## Review

- Introduction
- Thread lifecycle
- DMVs
- Common wait types