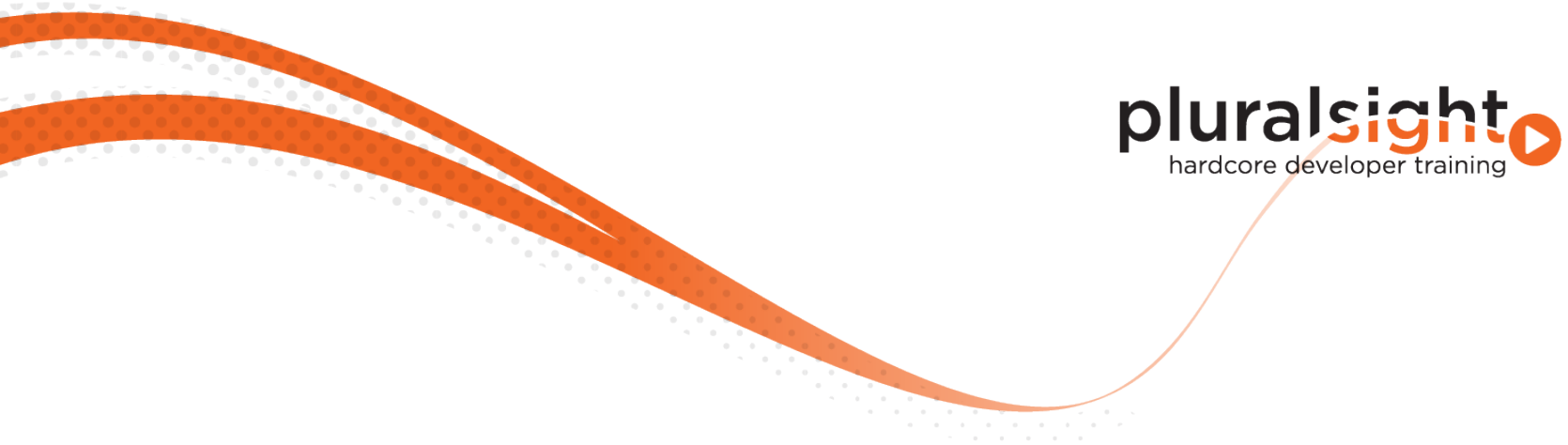
SQL Server: Performance Troubleshooting Using Wait Statistics

Module 4: Latches and Spinlocks

Paul S. Randal
Paul@SQLskills.com



pluralsight
hardcore developer training

Introduction

- **Wait statistics provide a huge amount of information about what is happening in SQL Server**
- **Sometimes more advanced analysis is required to get to the root of a performance problem**

- **In this module we'll cover:**
  - Latches and latch statistics
  - Spinlocks and spinlock statistics
  - The various DMVs to examine them
  - Examples of latch and spinlock contention

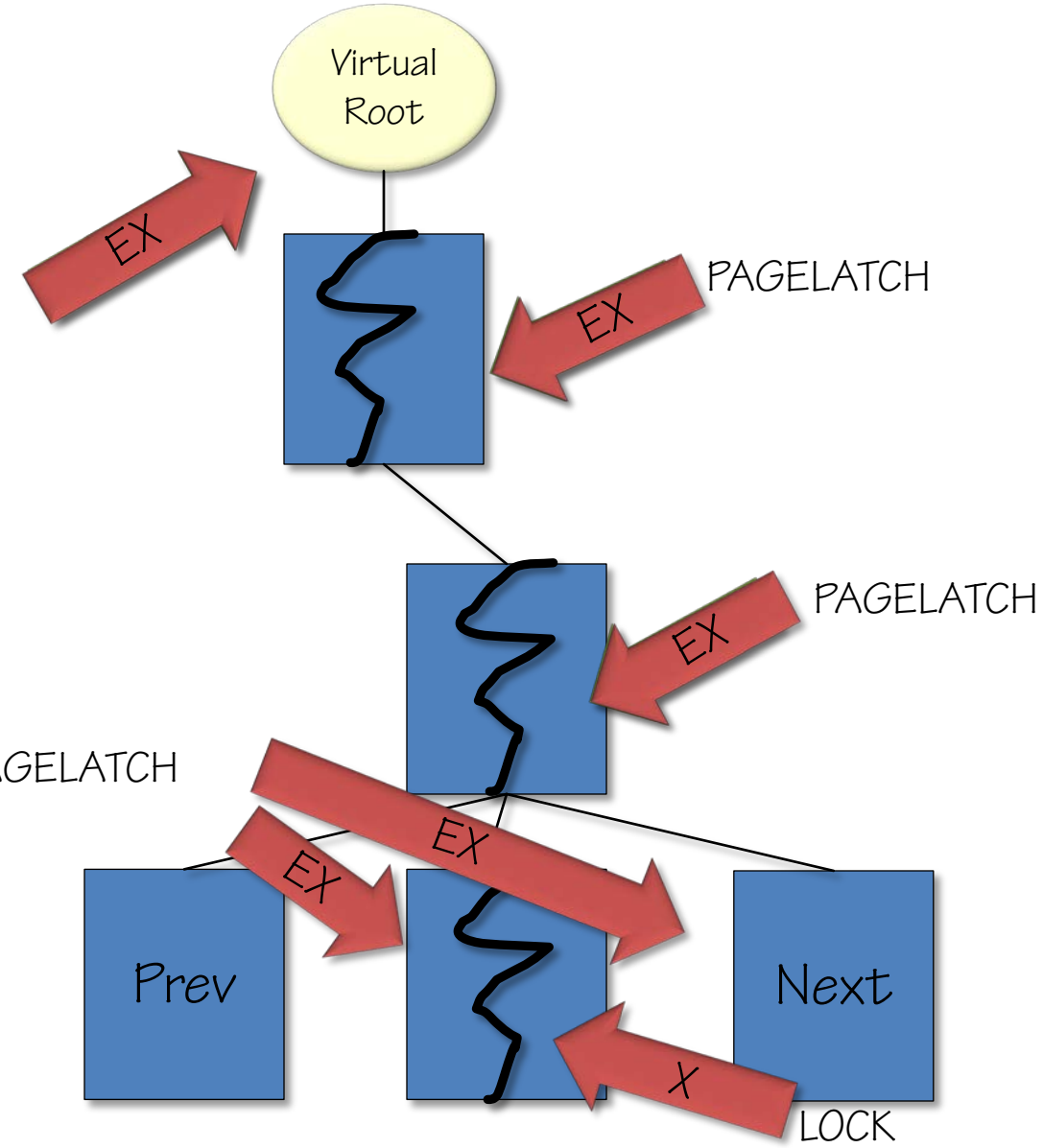
What are Latches?

- **A latch is a synchronization mechanism between threads**
  - Many people equate latches with locks, but they are quite different
- **A latch protects access to an in-memory data structure**
  - Whereas a lock protects transactional consistency
- **Latches are lightweight and are designed to be held for a short time**
  - Whereas a lock may be held until the end of a transaction
- **Latches cannot be controlled by SQL Server users**
  - Whereas locks can be controlled with hints and configuration options
- **Latches have a variety of modes, equating to the level of access to the in-memory data structure that is required**
  - E.g. an EX latch is required to change a data structure, and a SH latch is required to read most data structures
  - This is similar to the modes that a lock can have
- **SQL Server tracks latch wait times just like other waits**

Types of Latches

- **There are three types of latches:**
  - Latches waiting for data file pages to be read from disk into memory
    - Manifest as PAGEIOLATCH_XX waits
  - Latches for access to in-memory data file pages
    - Manifest as PAGELATCH_XX waits
  - Latches for access to all other data structures
    - Manifest as LATCH_XX waits
- **Examples of non-page latches:**
  - FGCB_ADD_REMOVE
  - ACCESS_METHODS_HOBT_VIRTUAL_ROOT
- **More information on all these latches will be in Module 5**

B-tree Page Split Example



From slide deck by Thomas Kejser (with permission)

Latch Contention

- **Just like with locks, latches can be a source of contention**
  - This means that what appears to be traditional blocking involving locks may actually be blocking involving latches
- **If one thread has a latch held exclusively then other threads must wait until that thread releases the exclusive latch**
- **This does not become a performance problem until there are many concurrent threads competing for access to the same latch**
  - As latches are only held for a short duration, a single thread waiting a very short time for another thread does not cause a problem
  - However, if hundreds of threads are waiting for a single thread, then that aggregates into a noticeable performance problem

- **Whitepaper on investigating latch contention: http://bit.ly/pS1kd1**

Tempdb Latch Contention Example

- **Classic contention issue on many SQL Servers**
- **Contention on in-memory PFS and SGAM pages in tempdb**
  - Wait type in sys.dm_os_waiting_tasks is PAGELATCH_EX
  - Resource description in sys.dm_os_waiting_tasks are '2:1:1' and '2:1:3'
- **Appears with many connections creating and dropping temp tables**
- **Alleviation methods include:**
  - Enabling trace flag 1118 to remove single-page allocations
  - Adding multiple tempdb data files so that round-robin allocation spreads the allocation load across multiple files
    - And hence the updating of allocation bitmaps is also spread, reducing contention
  - This blog post discusses this case in more depth: http://bit.ly/dsQB0g
- **This can also happen in user databases**
  - This blog post presents a scenario and wait stats analysis: http://bit.ly/isXIkW

sys.dm_os_latch_stats DMV

- **This DMV shows aggregated wait statistics for all non-page latch classes**
  - Aggregated since the server started or the latch statistics were cleared
- **This DMV provides:**
  - The name of each latch class
  - The number of times a wait has been for this latch class
  - The aggregate overall wait time for all waits for this latch class
  - The maximum wait time of any wait for this latch class
  - It does NOT list the latch modes being acquired
- **Some math is required to make the results useful**
  - Calculating the average times rather than the total times

What are Spinlocks?

- **A spinlock is an even lighter-weight thread synchronization mechanism than a latch**
  - Used like a latch for data structure access control
- **Spinlocks are used when the data structure access will be for an extremely short time so the overhead of acquiring a latch is too much**
- **Examples of spinlocks:**
  - FGCB_PRP_FILL
  - OPT_IDX_STATS
  - BUF_FREE_LIST
- **Troubleshooting spinlocks usually requires very deep knowledge of SQL Server internals**
  - However, it is interesting and useful to know what spinlocks are

Spinlock Internals

- **There is no waiting mechanism for spinlocks like there is for latches**
- **A thread tests the spinlock to see if it can be acquired**
- **If not, the thread sits in a loop checking whether it has the spinlock**
  - When the thread cannot acquire the spinlock, this is called a 'collision'
  - When the thread loops, this is called spinning
    - The spinlock is not checked on each spin, as this would take a lot of CPU
    - Checking for spinlock ownership is lighter weight than the initial acquire attempt
  - Spins required after a collision do not count as more collisions
  - The number of collisions, and the number of spins are tracked
- **After a certain number of spins, the thread yields and sleeps for a small time**
  - This is called a 'backoff'
  - For most spinlocks the backoff is for a fixed time, but some use an exponential backoff
- **SQL Server tracks all of this**

Spinlock Contention

- **When a large number of threads are contending for access to a single spinlock, this can lead to performance problems**
- **All these symptoms must be present for high CPU usage to potentially be from spinlock contention:**
  - High and increasing spins and backoffs for a spinlock
  - High CPU usage
  - Many connections to the server
  - CPU usage, spins, and backoffs increasing much faster than the workload is increasing (possibly an exponential divergence)
- **However, it is far more likely to NOT be spinlock contention so investigate other waits and latches first**
  - Common for some spinlocks to have very high spins and collisions
- **As mentioned previously, troubleshooting spinlock contention is very advanced so will not be covered in depth in this course**
- **Whitepaper on investigating spinlock contention: http://bit.ly/qZEJ4h**

sys.dm_os_spinlock_stats DMV

- **This DMV shows aggregated spinlock statistics**
  - Aggregated since the server started or the spinlock statistics were cleared
- **This DMV provides:**
  - The name of each spinlock
  - The number of times a collision has occurred for this spinlock
  - The total number of spins for this spinlock
  - The number of spins per collision for this spinlock
  - The total sleep time for this spinlock
  - The number of backoffs for this spinlock
- **Some work is required to make the results useful**
  - Storing the results for two calls to the DMV with some time in between
  - Differencing the two sets of results

Clearing Latch and Spinlock Statistics

- **Just like wait statistics, latch statistics can be cleared in all releases from SQL Server 2005 onwards using the code below:**

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

- **The ability to clear spinlock statistics was only introduced in SQL Server 2012 and can be done using the code below:**

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

- **Clearing these statistics allows the effect of a workload change to be measured against previous statistics**
  - However, these statistics are much more sensitive to 'pollution' from other workloads and background tasks than regular wait statistics
- **Be careful if you are taking periodic snapshots of these statistics as this will invalidate your series of snapshots**

Using Extended Events

- **For very advanced troubleshooting there are events that allow tracking of latches and spinlocks**
- **For latches:**
  - sqlserver.latch_suspend_begin
  - sqlserver.latch_suspend_end
  - These are similar to the sqlos.wait_info and sqlos.wait_info_external events but have a lot more information about the latch itself
- **For spinlocks:**
  - sqlos.spinlock_backoff
  - The spinlocks whitepaper has an example of using this
- **These are used rarely and are included here for completeness**

Transaction Log Example

- **Taking the transaction log and the logging system as an example, there are waits, latches, and spinlocks associated with it**
- **Waits:**
  - WRITELOG, LOGBUFFER, LOGGENERATION, LOGMGR, LOGMGR_FLUSH, LOGMGR_QUEUE, LOGMGR_RESERVE_APPEND
- **Latches:**
  - LOG_MANAGER, LOGBLOCK_GENERATIONS
- **Spinlocks:**
  - BUF_WRITE_LOG, LOGCACHE_ACCESS, LOGFLUSHQ, LOGLC, LOGLFM
- **From waits to latches to spinlocks, understanding the uses and troubleshooting becomes progressively harder and less likely to be required**
  - This is common across the SQL Server engine

Summary

- **Latches are a lighter weight synchronization mechanism that protects an in-memory data structure for short accesses**
- **Spinlocks are even lighter weight and are used when a latch would be too expensive**
- **Contention and waiting can occur for both, with different symptoms**
  - For latches, the waiting threads are suspended
  - For spinlocks, the waiting threads use a lot of CPU
- **Delving into latch waits is a lot more common than looking at spinlock statistics, and both require more advanced knowledge of SQL Server**

- **The next module will discuss common wait types and latch classes that manifest when troubleshooting performance problems and how to resolve them**