SQL Server: Myths and Misconceptions

Module 5: Locking

Paul S. Randal
Paul@SQLskills.com

Introduction

- **Locking always occurs in SQL Server**
  - Lots of confusion about how locking works
  - Many misconceptions about what operations cause blocking
- **In this module:**
  - Six myths around locking and blocking

Locking Myth #1

UNTRUE!!

- Myth: Online index operations do not acquire locks

- There are two blocking locks held by online index operations
  - ‘Almost Online Index Operations’ isn’t a very good marketing term
- Share table lock held at the start so the new index can be created
- Schema-Modification lock held at the end so the old index can be dropped
- Both locks can cause blocking while they’re queued
- Table Intent-Share lock held for the duration of the operation
  - Usually is not a blocking lock

Locking Myth #2

UNTRUE!!

- **Myth: Backups cause blocking**

- **Backups do not generally take locks so do not cause blocking**
- **One exception: bulk-operation lock prevents minimally-logged operations from occurring concurrently with backups**
- **Backups are very I/O intensive so might cause I/O contention which *appears* to be blocking as the workload slows down drastically**
- **To avoid this affect, consider:**
  - Taking large data backups at off-peak times
  - Taking more frequent transaction log backups
  - Separating database and backup file storage

Locking Myth #3

UNTRUE!!

- **Myth: DBCC CHECKDB causes blocking**

- **DBCC CHECKDB has been online by default since SQL Server 2000**
  - In SQL Server 7.0 and before, it did take blocking locks
  - In SQL Server 2000 it performed online log analysis
  - In SQL Server 2005 onwards it uses a database snapshot
- **It is one of the most I/O intensive operations in SQL Server**
  - Its read-ahead mechanism is very aggressive
  - The I/O load can make it *seem* like the workload is blocked

Locking Myth #4

UNTRUE!!

- **Myth: Lock escalation goes row to page, then page to table**

- **Lock escalation never does a two-step escalation**
- **Lock escalation usually escalates to a table lock**
- **SQL Server 2008 introduced a partition-level lock**
  - Lock escalation can be set to escalate to a partition
  - This allows concurrent partition-level operations with lock escalation
  - Default is still table-level lock escalation as some customers experienced deadlocks when partition-level escalation was tried as the default during early SQL Server 2008 CTPs

Locking Myth #5

YES AND NO!!

- **Myth: Blocking and deadlocks can occur with unrelated table rows**

- **Before SQL Server 2008 R2 it was possible to have blocking and deadlocks involving unrelated rows in a table**
  - The lock hash algorithm before SQL Server 2008 R2 was prone to occasional invalid lock hash collisions
  - The rewritten algorithm is 100 million times less likely to have this occur

- **You can use the undocumented %%LOCKRES%% function to investigate suspected lock hash collisions**
  - See http://bit.ly/JNDQWC for an example

Locking Myth #6

UNTRUE!!

- **Myth: NOLOCK / READ UNCOMMITTED means no locks are acquired**

- **NOLOCK and READ UNCOMMITTED do have to acquire some locks**
  - Schema-stability locks to prevent the structure of the table/index changing
  - BULK_OPERATION locks on heaps to prevent reading of unformatted pages
- **NOLOCK and READ UNCOMMITTED are the same under the covers**
- **And they still have to take latches to access the physical page images in memory, so there's still some potential for blocking at the latch level**