SQL Server: Myths and Misconceptions

Module 2: Performance

Paul S. Randal

Paul@SQLskills.com

pluralsight
hardcore developer training

Introduction

- **Everybody wants high performance for their workloads**
  - Lots of misconceptions about how to tune for performance
  - Lots of misconceptions about how some performance features work

- **In this module:**
  - Seventeen myths around performance and performance tuning

Performance Myth #1

YES AND NO!!

- **Myth: Instant file initialization can be controlled from SQL Server**

- **Instant file initialization is controlled at the Windows level**
- **SQL Server service account must be granted a Windows privilege**
  - Perform Volume Maintenance Tasks using Local Security Policy Editor
  - Restart SQL Server as it only checks at startup
- **Use trace flags 3004 (+3605) to watch zero initialization happening**
- **Use trace flag 1806 to temporarily disable instant file initialization**

- **Also, instant file initialization is possible in *all* editions of SQL Server**

Performance Myth #2

UNTRUE!!

- Myth: Using temporary tables for intermediate query results is always a good idea

- Using temporary tables may or may not help query performance
- Creating a temporary table to hold intermediate results forces SQL Server to interrupt the data pipeline through a query to persist the results to disk
- Sometimes just doing one query rather than pre-aggregating or pre-sorting can be way more efficient and lead to far lower run time and tempdb usage
- Always compare the methods before production
- If using temporary tables, make sure to only pull the data that's required, and create nonclustered indexes after table population

Performance Myth #3

UNTRUE!!

- **Myth: Tempdb should be X% the size of the largest database**

- **There is no formula for calculating tempdb size**
- **What's the largest use of tempdb?**
  - Memory spill (e.g. hash or sort)
  - Index rebuild with SORT_IN_TEMPDB
  - DBCC CHECKDB of largest database
  - Explicit use of tempdb through temporary tables
- **If it grows, and you're comfortable with the resulting size, explicitly set the size of the files**
  - Otherwise tempdb will resize when the instance restarts

Performance Myth #4

UNTRUE!!

- **Myth: Data compression is only for data warehouses**
- **It is true that data compression was originally written as a data warehousing feature**
  - Best for saving disk space and I/Os on large data reads
- **If you have enough CPU headroom, data compression can work fine for OLTP workloads**
  - Trade-off against cost savings from space savings in storage
  - More CPU used by PAGE compression than ROW compression

Performance Myth #5

UNTRUE!!

- **Myth: Data file shrinking does not affect performance**

- **Of course it does!**
- **While shrink is running:**
  - Lots of exclusive page locks
  - Everything it does is fully logged
  - Lots of page reads that aren’t part of the regular workload
  - Lots of dirty pages created leading to checkpoint I/O
- **After shrink has run:**
  - Index fragmentation
  - Resource usage from removing index fragmentation

Performance Myth #6

UNTRUE!!

- Myth: SQL Server does in-place updates of key columns

- SQL Server cannot do in-place updates of key columns
  - Unfortunately most references get this wrong
- SQL Server must do a delete-plus-insert operation of the whole record
  - Even for a fixed length key…
  - This is for 'Halloween protection'
- This can even lead to page splits occurring…

Performance Myth #7

MAYBE!!

- **Myth: Using snapshot isolation has no effect on performance**

- **If there are no updates performed, there is no effect as no versions are generated**
- **Otherwise:**
  - Generating a version adds a 14-byte tag to the end of a record, which can lead to page splits
  - Generating versions causes I/O in tempdb
  - Reading versions causes I/O in tempdb
- **But snapshot isolation can be fabulous for removing blocking and deadlocking problems**
  - Especially when the code cannot be changed
  - The page split issue can be worked around using FILLFACTORs

Performance Myth #8

UNTRUE!!

- **Myth: Checkpoints only write committed changes to disk**

- **Checkpoint writes all pages marked dirty**
  - This is regardless of whether the change was made by a committed or uncommitted transaction
- **Use sys.dm_os_buffer_descriptors to examine the relative proportion of dirty vs. clean pages in the buffer pool**
  - See http://bit.ly/pYeB4K for more information and scripts to use

Performance Myth #9

IT DEPENDS!!

- **Myth: Adding the 9th table column is a size-of-data operation**
  - Or 17th, 25th, 33rd, etc (basically when #columns modulo 8 = 1)

- **Each data record has a NULL bitmap with one bit per column in the record**
  - Regardless of whether the columns are nullable
- **If the new column has a NULL default, nothing is changed except Storage Engine metadata for the table**
  - It remembers that there's potentially an extra column
  - Each record has a count of columns in the record
- **If the new column has a non-NULL default, every record must be changed by the ALTER TABLE statement**
  - Fixed in SQL Server 2012!

Performance Myth #10

MAYBE!!

- Myth: Tempdb data files should be 1:1 with processor cores

- This is one of the biggest myths out there
- In 2000 the mantra was #files = #cores, and use trace flag 1118
- In 2005 onwards, the "official" mantra is the same as 2000
- So what to do?
  - 1:1 does not apply to entire range of small-medium-enterprise
  - Most of us say #files = ¼ to ½ #cores and work up
  - From Bob Ward's session at SQL PASS in 2011:
    - < 8 cores, #files = #cores
    - > 8 cores, # files = 8, and increase in blocks of 4
  - And use trace flag 1118
- Don't forget all data files must be same size
- Microsoft are now changing their guidance on this
- Does not apply to log files: only one is required

Performance Myth #11

IT DEPENDS!!

- **Myth: The best thing to put on an SSD is tempdb and/or the transaction log**

- **Don’t fall into the trap of listening to other people**
- **Investigate where your biggest I/O subsystem bottleneck is and use the SSD there**
  - That may be a volatile data file
- **Or design a new I/O subsystem layout to take advantage of the SSD**
- **By following the tempdb/log advice, you may not get any gain from using the SSD**
  - Lack of ROI will not make the hardware budget owners happy

Performance Myth #12

UNTRUE!!

- **Myth: Disk queue length should *always* be very low**

- **All threads in SQL Server can issue asynchronous I/Os**
- **Some parts of SQL Server will drive the disk queue length into the 100s**
  - DBCC CHECKDB
  - sys.dm_db_index_physical_stats
  - Bursts of activity in the log or checkpoints

- **Generally you want it to be low, but be aware that spikes are normal**

Performance Myth #13

UNTRUE!!

- **Myth: 300 is a good Page Life Expectancy threshold**

- **300 is a terrible value to use**
- **Page life expectancy measures how quickly the buffer pool is being completely flushed and refilled**
  - Measured in seconds
  - Instantaneous measure, not a rolling average
- **Do you think that flushing your 100GB buffer pool every 5 minutes is a sign of a healthy SQL Server?**
- **A sustained value of 300 is too LOW to trigger worry**
- **That guidance is from 5+ years ago!**
- **Use (buffer pool size in GB / 4) * 300 as a better Page Life Expectancy threshold**
  - For a 100GB buffer pool, this would give a threshold of 7500

Performance Myth #14

UNTRUE!!

- Myth: NUMA does not affect Page Life Expectancy

- Beware of the Buffer Manager Page Life Expectancy counter on servers with NUMA configured
- The Buffer Manager counter is an average of the Page Life Expectancy from each partition of the buffer pool
- Monitor each Buffer Node Page Life Expectancy counter
- See http://bit.ly/oTjRwl for an example

Performance Myth #15

UNTRUE!!

- **Myth: CXPACKET waits mean disable parallelism**

- **CXPACKET waits mean that parallel queries are running**
  - It can also mean a parallel query where one of the threads is taking longer than the others because of skewed statistics, for example
- **Do NOT just set MAXDOP = 1**
- **Look deeper…**
  - Could be missing indexes causing parallel table scans
  - Could be outdated statistics causing a bad plan
- **If you must reduce parallelism, consider:**
  - Increasing the cost threshold for parallelism
  - Setting MAXDOP just for that query
  - Using Resource Governor to limit MAXDOP for groups of queries

Performance Myth #16

UNTRUE!!

- **Myth: ASYNC_NETWORK_IO waits mean the network has an issue**

- **This is usually not a network problem**
- **ASYNC_NETWORK_IO means that SQL Server is waiting for an application to acknowledge receipt of data**
- **Most often the result of poor application programming**
  - RBAR (Row-By-Agonizing-Row) consumption of data
  - For instance, do a large select using SSMS on the same server as the instance of SQL Server and you'll see ASYNC_NETWORK_IO waits, with no network involved

Performance Myth #17

UNTRUE!!

- **Myth: Instance-wide MAXDOP cannot be overridden**

- **Anyone can override the instance-wide MAXDOP setting**
  - Using a MAXDOP query hint
- **To truly limit someone's MAXDOP, use Resource Governor**
  - Workload Group MAX_DOP (yes, there's an underscore) cannot be overridden