SQLBits

Session: Friday, 17:10

Performance Mythbusters (L2-300)

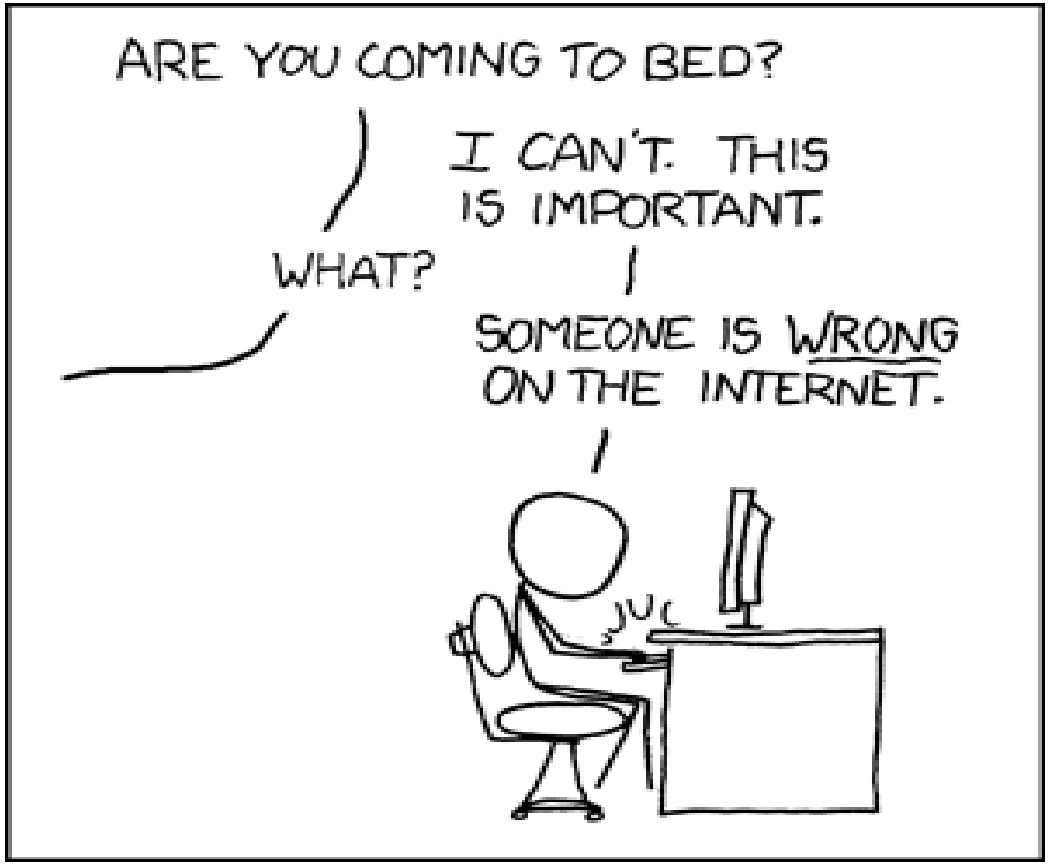
Paul S. Randal

paul@SQLskills.com

This is me: Paul S. Randal

- Consultant/Trainer/Speaker/Author
- CEO, SQLskills.com
  - Email: Paul@SQLskills.com
  - Blog: https://www.SQLskills.com/blogs/Paul
  - Twitter: @PaulRandal
  - 5 years at DEC responsible for the VMS file-system and chkdsk
  - Almost 9 years as developer/manager in the SQL Storage Engine team through August 2007, ultimately responsible for Core Storage Engine
- Instructor-led training, consulting (anything you need)
- Online training: http://pluralsight.com/
- Get our newsletter: https://www.sqlskills.com/Insider

And This is My Life…



Source: http://xkcd.com/386/

Why is This Important?

- Lots of myths and misconceptions have grown and persisted over the years about SQL Server performance
- Adherence to these misconceptions can lead to:
  - Bad practices
  - Wasted time and resources
  - Confusion
  - Arguments ☺
- Let's debunk some myths!

Myth #0

BUSTED!!

- Oracle is performs much better than SQL Server
- This one's obviously untrue! ☺
- On to the real ones…

Myth #1

BUSTED!!

- Versioning (snaphot isolation) always helps performance

- Using snapshot isolation or read-committed snapshot isolation is either for reducing blocking/deadlock, or for per statement/transaction unchanging view of a database
- Creates versions
  - Makes rows 14-bytes longer, so maybe causes page splits
  - Uses version store in tempdb, so causes tempdb workload
- Could be 10-15% throughput drop for some workloads
- What about Accelerated Database Recovery in 2019?

Myth #2

IT DEPENDS !!

- Buffer Manager: Page life expectancy is the best PLE counter to use

- If on a NUMA system, Buffer Manager PLE is the harmonic mean of PLEs from all Buffer Nodes
  - Buffer pool is split into partitions on NUMA system
- Monitor all Buffer Node: Page life expectancy counters
- Example harmonic mean calculation
  - 4 buffer nodes with PLE = 4000, 4000, 4000, 2200
  - Buffer Manager PLE = 4 / (1/(1000x4000) + 1/(1000x4000) + 1/(1000x4000) + 1/(1000x3300)) / 1000 = 3321

Myth #3

MAYBE !!

- You can't override max degree of parallelism

- Server/database-scoped MAXDOP can be overridden by anyone with any privilege using a MAXDOP query hint!
- Track who is overriding MAXDOP = 1 using Extended Events
  - https://www.sqlskills.com/blogs/paul/who-is-overriding-maxdop-1-on-the-instance/
- Enforce your MAXDOP by using Resource Governor with a MAX_DOP = 1 setting and forcing all connections to use it
- Enforce MAXDOP = 1 by setting 'cost threshold for parallelism' very high

Myth #4

BUSTED!!

- AG readable secondaries don't cause performance problems

- All queries on a readable secondary are converted to snapshot isolation
- This means 14-byte versioning tags must be present on the secondary
- This means the tags have to be added on the primary
  - As the primary and secondaries are exact physical copies of each other
  - But the tags don't have to be filled in on the primary, just the space needs to be accounted for
- All changes to the primary database will incur a versioning tag which will start to cause page splits and index fragmentation!

Myth #5

KIND OF !!

- TRUNCATE TABLE (or PARTITION) is so fast because it's not logged

- It's always logged, but sometimes the logging is delayed
- Same thing for DROP TABLE
- Mechanism is called 'deferred drop' and was added in 2000 SP3
  - If less that 128 extents, done immediately
  - Otherwise allocations are unhooked and dropped in small chunks later by background task, so it appears as if the operation is not logged
- Try truncating a table in a transaction and then rolling it back – works!

Myth #6

BUSTED!!

- Using temp tables for intermediate query results is always a good idea

- Creating a temp 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
- And if using temp tables, use minimum amount of data and correct indexes

Myth #7

BUSTED!!

- "Fully logged" means you'll always see one log record for each part of an operation

- Consider a rebuild of a 100,000 row index
  - You would expect to see 100 thousand LOP_INSERT_ROW log records, right?
  - Wrong – it will log LOP_FORMAT_PAGE log records instead with full page images with the net effect of all the inserts on
- "Fully logged" simply means the transaction log contains enough information to reconstitute the transaction after a crash or restore

Myth #8

BUSTED!!

- NOLOCK / READ UNCOMMITTED means no locks

- First off, they're the same thing
- And they do have to acquire some locks:
  - Schema-stability locks (Sch-S) to prevent the structure of the table/index changing
  - BULK_OPERATION locks on heaps to prevent reading of unformatted pages
- 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

Myth #9

BUSTED!!

- You should always plan a backup strategy
  - Huh? Isn't this is a performance session?

- Always plan a *restore* strategy
- Then plan what backups you need to take

- The other way can result in a veeeeeery slow restore process

- Let me tell you a story…

Myth #10

BUSTED!!

- The best thing to put on SSDs are always tempdb and transaction logs

- Don't fall into the trap of listening to other people
- Investigate where your biggest I/O subsystem bottleneck is
  - Try to solve it within SQL Server
  - If not, put that on your SSD
- Or design a new I/O subsystem layout to take advantage of the SSD
- What about the RAID level to use?

Myth #11

BUSTED!!

- Index fragmentation only affects range scans

- Many people think this myth is actually true
- Index fragmentation has two forms:
  - Logical fragmentation that stops efficient readahead
  - Low page density that wastes space
- The second form is very important to consider:
  - Wasted buffer pool space = lower data density, more physical reads
  - Wasted disk and backup space
- And you *get* index fragmentation from page splits

Myth #12

BUSTED!!

- Adding more memory will always help performance

- Not if your workload isn't reading from or spilling to disk
  - If workload is memory resident, more memory doesn't reduce reads
  - If workload isn't spilling to disk, more memory won't help
- Consider some of the potential problems
  - Shutting down the instance will take longer
  - P.O.S.T. of the server will take longer
  - Allocating buffer pool memory may take longer (see KB article 2819662)
  - Warming up the buffer pool will take longer
  - Could lead to complacency

Myth #13

BUSTED!!

- Lock escalation goes from row to page to table

- Lock escalation is:
  - Row to table/partition
  - Page to table/partition
- Partition locks introduced in 2008 to reduce problems from lock escalation to table when other queries want other partitions
  - Off by default, configured per table

Myth #14

BUSTED!!

- Query Store does not have performance overhead

- Two ways Query Store can introduce problems:
- On a busy system with lots of ad hoc queries, query hash map is bottleneck and can lead to QDS_STMT waits
  - Set QUERY_CAPTURE_MODE to AUTO, parameterize, and maybe disable QS
- By default, QS loads all persisted data at startup, preventing database coming online, possibly for seconds or minutes (you'll see QDS_LOADDB wait increasing)
  - TF 7752 to make it asynchronous, and read-only mode until loaded

Myth #15

BUSTED!!

- Rebuilding indexes solves performance problems even when there's no index fragmentation

- It's the query plan recompilation that 'fixes' the performance problem
- Rebuilding an index causes plan recompilation for plans on that table
- If a poor query plan had resulted (e.g. from parameter sniffing), the next plan to be compiled might be better

Myth #16

MAYBE !!

- Adding an extra file to tempdb will help solve contention issues

- Adding an extra file means SQL Server can alternate between the files
- But allocation also takes into account proportional fill
  - It will allocate proportionally more from files with more free space
- If the existing file is quite full, the new file becomes allocation hot spot
  - No alleviation of contention issues!
- Make sure to take that into account when working with tempdb

Myth #17

BUSTED !!

- Database snapshots do not cause performance problems

- Firstly, a database snapshot is not a full copy of the database
- Pages are synchronously pushed into the snapshot the first time they're changed after the snapshot is created
- One synchronous copy per snapshot that exists, so more snapshots equals more overhead
- What about reverting to a snapshot, and the transaction log?

Myth #18

BUSTED !!

- Online index operations do not acquire locks

- They do, but ‘almost online index operations’ isn’t a good name ☺
- At the start there’s a short-term table S lock
- At the end there’s a short-term table SCH-M lock
- Both have potential for blocking
- Can kind of work around that from 2014 using WAIT_AT_LOW_PRIORITY
- And 2017 introduced pause/resume of online index operations

Myth #19

BUSTED !!

- FILLFACTOR is used by regular DML operations

- That wouldn't make any sense…
- FILLFACTOR proactively leaves space for new or expanded rows to avoid page splits
- Only applies during index build, rebuild, reorganize

Myth #20

TRUE !!

- Logging is more efficient in tempdb

- No crash recovery in tempdb, so…
  - No redo information in tempdb log records
  - Log blocks don't flush in tempdb when a transaction commits
  - Checkpoints don't always flush out data pages
  - Checkpoints only happen when the log becomes 70% full

Myth #21

BUSTED !!

- Always use data compression

- Data compression is not suitable for all workloads
- We designed it in 2008 for data warehouses, but it can be used for OLTP
- For busy OLTP, overhead on every read or change
  - No cached uncompressed data stored anywhere
- Trade-off of reduced storage space and costs vs. slow down in workload
- Consider using backup compression and/or removing wasted empty space

Myth #22

BUSTED !!

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

- SQL Server 2000: rule was #files = #logical processor cores, and TF 1118
  - E.g. my laptop CPU has 4 physical cores plus hyperthreading = 8 logical cores
- SQL Server 2005 onwards: Microsoft guidance was same until 2011
  - Everyone else said to start with ¼ to ½ the number of logical processor cores
- Universal guidance now in KB article 2154845
  - < 8 cores, start with #files = #cores
  - > 8 cores, start with #files = 8
  - Increase in blocks of 4 if still seeing contention

Myth #23

BUSTED !!

- Multiple log files will help performance

- SQL Server will always use log files sequentially
- You may see them all having I/Os, but that's just updating the file header pages
- The only time another log file is needed is if the first one fills up and cannot grow, you cannot take a log backup, and you do not want to break the log backup chain
- Remove additional log files once you don't need them

Myth #24

BUSTED !!

- The log should always be as small as possible

- The log needs to be as big as it needs to be
- Do not regularly shrink the transaction log
  - It'll just have to grow again, and can't use instant initialization
- How big should the log be?
  - Single largest transaction (ETL, large index rebuild, large update)
  - Asynchronous database mirroring/AG SEND queue
  - How long is the longest data backup?
  - Transactional replication (beware of CDC too)

Finally…

- It depends

YES, IT REALLY DOES!!

- The answer to all questions about SQL Server that do not have obvious yes/no answers always starts with 'it depends'
- The trick is then to explain *why* it depends, *what* dependson, and *when* it depends
- Don't answer 'it depends' and then answer with no follow-on explanation

NOOOOOOO!!!

- One exception: should auto-shrink be enabled?

Plenty More Performance Myths Around…

Nested transactions help performance

The log is zeroed when cleared

BULK_LOGGED helps log backup performance

Transactions are faster in SIMPLE

Restoring a backup fixes fragmentation and statistics

Partitioning always helps query performance

Backups cause blocking

Instant file initialization works for transaction log files

ALL BUSTED !!

Summary and Resources

- Make sure you corroborate what you read online
- If something sounds fishy, try it yourself!

- Blog:
  - https://www.sqlskills.com/blogs/paul/category/misconceptions/
- Pluralsight
  - SQL Server: Myths and Misconceptions

Thank you!

Questions? Paul@SQLskills.com