

Reference Summary and Stories



Paul's Client Problems

- Long-term consulting client
- Complaint #1
 - Tempdb grows to 60GB+ every week and causes out-of-space issues for other databases
- Complaint #2
 - Consistent ~80% CPU load, regular query timeouts, high I/O load
- Hardware
 - SQL Server 2008 R2, 24-core server, 64GB memory
- Major design problem: database schema very over-normalized, necessitating lots of joins

Investigation + Solution

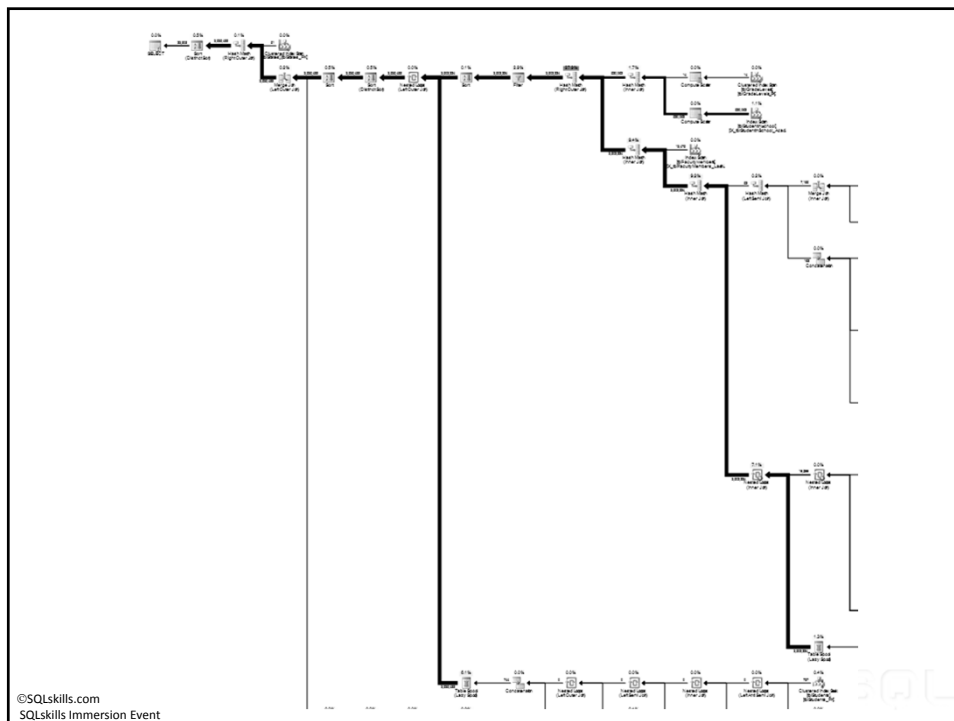
Complaint #1 – tempdb usage

- Implemented tempdb space usage monitoring to identify the proc that was causing most tempdb growth
- Visually identified issues with the proc
 - Using temp tables to pre-aggregate data from main database before further processing
 - Multiple single-column indexes created on each temp table that aren't used by the queries, plus clustered index is created before temp table population so no statistics
 - Not filtering out data so working over 10 years of data to produce results for one month
 - Reduced tempdb usage by 10-20GB by fixing these
- Annotated proc code with tempdb usage information for each block
 - Found code doing a join on two temp tables that already had those join results in, resulting in massive hash join and worktable
 - Reduced tempdb usage down to < 1GB by fixing this

©SQLskills.com
SQLskills Immersion Event

3

SQL



Investigation + Solution

Complaint #2 – high CPU, high I/O, query timeouts

- Indexes:
 - Analyzed index usage stats to remove indexes that are not being used to reduce index maintenance on insert/update/delete
 - Analyzed missing indexes and created the highest impact ones (27 indexes with improvement measure > 200000 after 2 weeks)
 - Examined index keys and INCLUDED columns to consolidate indexes
 - This all vastly reduced buffer pool pressure and I/O
- Analyzed query stats to determine most executed queries, and most expensive queries (two sets with some intersection)
 - Tuned to be as efficient as possible by removing RID/key lookups, large table/index scans, cartesian products
- After all of this over several months at 10-15 hours per week (they have *lots* of procs, views, functions), CPU constantly < 30% and no more query timeouts

Joe's Support Scenarios

- Is the issue happening right now?
 - How NOT to do it? Go on a fishing expedition
 - How TO do it? Waiting tasks / Running / Runnable
- Examples:
 - 1,000 stores – one server – no middle tier, ASYNC NETWORK
 - Is the query necessary?
 - BizTalk I/O issue – *no really, its I/O*
 - *Are all the right people involved?*
 - Not SQL's fault
 - Show, don't just tell
 - Observer overhead
 - Sometimes your own monitoring can be the cause of your root cause of your performance issues

Jon's Client Problem

- Customer complaint
 - Frequent failures of SQL Server 2008 Failover Cluster with prolonged recovery times for the primary database
- Hardware:
 - HP DL785 Server 4 x 12 core AMD Magny-Cours processors, 512GB RAM
 - EMC Clarion SAN
- Software:
 - Two node SQL Server 2008 failover cluster running on Windows Server 2008

Investigation

- SQL Server Error Log terminates abruptly
- Windows Event Logs show loss of cluster connectivity to SAN disks
- SQL Server Error Log contains I/O stall warnings
- SQL Server Error Log startup entries show a single NUMA node being created
- Server wide MAXDOP configure to 1
- Virtual file stats contain high latency values for the disk arrays with large numbers of read and write I/O stall counts

Solution Part 1

Enable Hardware NUMA

- DL785 Servers using AMD Magny-Cours processors should not use Node Interleaving when running SQL Server
 - This limits the system to a single Lazy Writer thread to manage the configured 480GB of Buffer Pool memory
 - Removing Node Interleaving created 8 hardware NUMA nodes of 6 processors each and 8 separate Lazy Writer threads that each manage 60GB of Buffer Pool memory

Solution Part 2

Reduce HBA Queue Depth Configuration

- The SAN FA ports were reaching saturation resulting in lost connectivity by the cluster for the shared disks
 - Explains the termination of the SQL Server ErrorLog and the missing memory dump files on the server
 - Reducing queue depth from 64 to 32 on the servers reduced the FA saturation under peak load until a SAN upgrade could be performed by the hosting facility

Solution Part 3

Change Backup Configuration

- The long recovery times for the primary database were being caused by the size of the active portion of the transaction log at the time of failure
 - At the point of failure a FULL backup of the database had been running for many hours delaying log truncation from occurring
 - The transaction logs had excessive VLFs from a small autogrowth setting as well
- Changing backup configuration from being over the network to using SAN disks, using multiple backup files, and enabling backup compression reduced the time to complete a FULL backup

Kimberly's Client Problems

- New consulting client – wanted to talk (phone) and “discuss” problem
 - They had been burned by [multiple] consultants that charged them a lot and didn't deliver (which adds another level to the interaction & communications... sometimes this job is more than just technical skill)
 - See Paul's blog post: [The most important consulting skill is... effective communication](#)
- Complaint #1
 - CPU periodically goes to 100% and then seems to stay there – cannot do anything (nothing seems to run, no new connections, system essentially hung)
 - No code changes, no schema changes, nothing changed
(*I finally found that they did change the client application to allow parameters to procs that weren't allowed previously.*)
 - **Customer's Solution:**
 - Failover their cluster to get out of this problem... (not good)
- Hardware
 - SQL Server 2005 Failover Cluster, 24-way, 128GB memory

Investigation + Solution

Complaint #1 – CPU spikes/stays

- Customer couldn't connect – wasn't aware of the DAC
 - Made them aware that they could connect at the time when users couldn't
 - Allowed them to see what was running at the time through DMV queries
 - Found **one** stored procedure that had execution times WAY out of normal time
- Reviewed code – found horrible “multi-purpose procs” as **general** coding strategy
 - Numerous dialogs had multiple optional parameters
 - Each dialog used **one** procedure no matter what parameters supplied
 - Stored procedures were filled with:


```
WHERE (columnx = @variablex OR @variablex IS NULL)
AND (columny = @variabley OR @variabley IS NULL)
AND (columnz = @variablez OR @variablez IS NULL)
```
- Customer **VERY** nervous about changing any code whatsoever
 - Were very nervous due to lack of success with prior consultants
 - Weren't ready to adopt different code – wanted “anything” else instead of making code changes
- Solution 1: Agent job – every minute, check execution times and sp_recompile tableX **IF** an “abnormal” execution time (> Y seconds) in cache... but, only caught somewhat poor plans (not really horrible ones)

©SQLskills.com
SQLskills Immersion Event

13

Solution Part 2

- Customer started to believe me and started to trust us...
- Re-wrote the code of the worst offender – put it into QA, put it into production (used the “agent job” solution as the interim solution)
- Delivered training (webcasts) to their team in Russia
 - Lots of debates over best practices – had to convince them (different folks) all over again
 - Had to convince them that recompilations are NOT always bad (nor are they always good...)
 - Indoctrinated them into the “it depends” way of life ☺
 - Changed their coding style to use EXEC (@string) in some cases and sp_executesql in others (and when/why)
- Finally, got their problems solved
- Went on-site + helped them interview their new full-time DBA (**very** happy ending)

©SQLskills.com
SQLskills Immersion Event

14

Summary

- It depends.....
- But... now you know:
 - Where to look to analyze the problem
 - Numerous solutions with a plethora of possible options to fix it
 - How to use the tools effectively
 - And most importantly:
What it depends on!