

SQLskills Immersion Event

IEPTO2: Performance Tuning and Optimization

Module 1: Creating a Foundation for Tuning and Optimizing

Erin Stellato

Erin@SQLskills.com



Overview

- How we approach tuning and optimization
- Server, instance, and database settings
- Checking system health
- Methods for analyzing performance

Start with what you know

1. What is the problem?

- ❑ We are experiencing turbulence with our SQL Server instance.
- ❑ There is an out of control query for one client on one database that impacts all other clients on the instance – it uses up all the threads available and causes problems with application blocking.
- ❑ We discovered a DBCC error.
- ❑ We just upgraded to SQL Server 2016 from SQL Server 2008R2 and we're having high CPU issues.
- ❑ We want to implement partitioning to help with performance.
- ❑ What is the performance impact of having OLTP and reporting on the same server?
- ❑ We have a server that is not functioning well, after the last reboot it started acting differently. We need to get it back to normal.
- ❑ Queries are running slow/queries are blocking.
- ❑ The system is down.

Start with what you know

1. What is the problem?

- ❑ Facts vs. opinion
- ❑ Ask the question: “What problem are you trying to solve?”
- ❑ Get as specific as possible here – whatever is defined as the problem is what you have to try and fix (scope is very important)

2. What does the system look like when there is not a problem?

- ❑ What is CPU typically?
- ❑ What does memory use/available memory look like?
- ❑ What are typical disk latency numbers?
- ❑ What are the usual waits?
- ❑ How long does it take for the top 20 queries to execute?
- ❑ Are there metrics for other regularly-scheduled processes?
- ❑ What are durations for maintenance tasks?
- ❑ Typical health of HA/DR components?

**Do you
have
baseline
data?**

Start with what you know

1. What is the problem?

- ❑ Facts vs. opinion
- ❑ Ask the question: “What problem are you trying to solve?”
- ❑ Be as specific as possible here – whatever is defined as the problem is what you have to try and fix (scope is very important)

2. What does the system look like when there is not a problem?

- ❑ A.k.a. “Does a baseline exist?”

3. What do you know about the system’s configuration?

4. What will define success/project completion?

- ❑ Again, be as specific as possible, and put it in writing.

Overview

- How we approach tuning and optimization
- Server, instance, and database settings
- Checking system health
- Methods for analyzing performance

Server Settings

- **Physical server or VM?**
- **Number of sockets and logical processors**
 - If VM, are these as expected?
 - NUMA configuration (physical and VM)
 - automatic soft-NUMA disabled
- **Memory**
- **Volume partition alignment**
- **Lock Pages in Memory**
- **Instant File Initialization**
- **Power Management**
- **BIOS settings**
 - Hardware virtualization support, Intel Turbo Boost, Intel HT

Instance Settings (1)

- **SQL Server version and Edition**
- **max server memory (MB) and min server memory (MB)**
- **CPU affinity mask**
 - Do the NUMA nodes and logical processors match what you see in msinfo32?
- **priority boost**
- **optimize for ad hoc workloads**
- **cost threshold for parallelism**
- **max degree of parallelism**
- **backup checksum**
- **backup compression**
- **remote admin connections**
- **Trace flags**

Instance Settings (2)

- **Is anything set that is not expected?**
 - Ad Hoc Distributed Queries
 - blocked process threshold (s)
 - clr enabled
 - fill factor (%)
 - Ole Automation Procedures
 - scan for startup procs
 - xp_cmdshell

Database Settings (1)

- Collation
- Recovery model
- Compatibility level
- Auto close
- Auto create incremental statistics
- Auto create statistics
- Auto shrink
- Auto update statistics
- Auto update statistics asynchronously

Database Settings (2)

- Legacy Cardinality Estimation
- Max DOP
- Parameter sniffing
- Query Optimizer fixes
- Delayed Durability
- RCSI
- Parameterization
- Indirect checkpoints
- Page verify
- Query Store
- tempdb configuration

Database Settings (3)

- Identity cache
- Interleaved Execution for multi-statement table-valued functions
- Batch mode memory grant feedback
- Batch mode adaptive joins

Database Settings (4)

- T-SQL Scalar UDF inlining
- Automatically elevate supported operations to resumable
- Row mode memory grant feedback
- Batch mode on rowstore
- Deferred compilation for table variables
- Accelerated plan forcing
- Lightweight query profiling
- Display error message: String or binary data would be truncated
- Collect last query plan statistics
- Determine whether row-level security affects execution plan cardinality

Additional Database Settings in Azure SQL DB

- Automatically elevate supported operations to online
- Optimize for ad hoc workloads
- Collect execution statistics for natively compiled T-SQL modules
- Automatically drop temporary tables
- Configure pause duration for resumable index activity
- Wait for Sch-M lock on a low priority queue for asynchronous statistics updates

Overview

- How we approach tuning and optimization
- Server, instance, and database settings
- Checking system health
- Methods for analyzing performance

What Does a Healthy SQL Server Look Like?

- Supports a given workload within resource limits and query performance meets business-defined SLAs
- Maintenance tasks execute regularly and successfully
- Architecture can meet defined RTO and RPO for all databases
- Has every option set according to best practices
- Running the latest build of SQL Server available
- All performance counters and other metrics within “standard” values

How to Check System Health

- **It starts with a review of system configuration**
 - May find potential problems and easy wins here
- **Examine resource use**
 - CPU, Memory, Disk, SQL Server counters
 - Just because values are outside “standards” does not mean there’s a problem
- **Start digging into what’s running and how it’s running**
- **Can you boil down performance problems to a few simple things?**
 - Specifically, not the CAUSE, but how it manifests

Options for Checking Settings

- All instance and database-level settings can be checked using system views or DBCC commands, or manually
- Server-level information can be verified with PowerShell (using WMI or CMI) or manually
- The only time you should be investigating whether a setting has the optimal value is the *first* time you see a server
- Re-checking a system, or checking settings as part of troubleshooting, should just be *validation*
- This is why you need a baseline

Checking Settings Quickly

- **A scripted solution is the easiest, fastest, and most reliable way to check settings of interest**
- **There are many free scripts and tools available to do this**
 - Use them as they are
 - Customize to fit your needs
 - Write your own as an exercise in scripting
 - What you create depends on what's important to you
- **Beyond checking, you need to validate that settings haven't changed recently**
 - Capture on a regular basis in an "admin" database and compare
 - Custom scripts
 - A combination of Audit and Extended Events

Demo

Checking system health

Overview

- How we approach tuning and optimization
- Server, instance, and database settings
- Checking system health
- Methods for analyzing performance

Troubleshooting vs. Tuning

- Troubleshooting = I need to *find* the problem
- Tuning = I need to *fix* the problem
- The same information is gathered whether you're troubleshooting or tuning

Troubleshooting First Steps

- **Description of problem from customer/users**
 - Be wary of their perception
 - Focus on facts
 - Try to avoid assumptions (this can be hard!)

Troubleshooting Basics

- **Don't assume Symptom = Root Cause**
 - Troubleshooting is not an exact science, and the same symptoms can result from many root causes
- **Don't do 'knee-jerk' troubleshooting**
 - DBCC FREEPROCCACHE
 - UPDATE statistics
 - Rebuild indexes
 - Failover or reboot
- **If you do any of those and they “work”, don't do it again**
 - You didn't solve the problem – you didn't even define it; you just made it go away
- **Work through the data to see what may be the root cause**
 - You may have to let performance suffer to find the problem
 - You control how long the system suffers

Diagnostic Data Collection (1)

- Data collection should include multiple sources to facilitate root cause identification and validation from multiple sources
- Good starting points:
 - Wait statistics
 - This is *always* where we start
 - Adam Machanic's who_is_active SP
 - Query DMVs directly if having memory issues
 - Performance Monitor collection of performance counters for hardware, the Windows OS, and SQL Server
- Can you confirm that the problem is with SQL Server and not another application?
 - Sometimes you are working to prove the issue is *not* due to the database
 - As much as you try to pinpoint the root issue, you also identify what is *not* the problem

Diagnostic Data Collection (2)

- Can you identify blocking?
- Is the problem contained to a specific database?
- More advanced collection
 - Glenn Berry's DMV Diagnostics script
 - Buffer and plan cache usage
 - Index statistics
 - File statistics
 - Extended Events
 - CPU, reads, writes, duration
 - Query Store

Sources of Data

- **PerfMon**
- **DMVs**
- **Query plans**
- **Extended Events/Trace**
- **DBCC commands**
- **SQL Server ERRORLOG**
- **Third-party tools**
- **Vendor tools for storage, VMware, etc.**
- **Windows System and Application logs**
- **CU/SP release notes**

Additional Considerations

- **What's sharing resources/storage?**
- **Has maintenance been running, and as expected?**
 - Backups, statistics, index fragmentation, DBCC CHECKDB
- **What features are in use?**
- **Impact of HA and/or DR architecture**
- **The data itself**
- **Database schema/design**
 - Filegroups
 - Files
 - Normalization
 - Data types
 - Indexes
 - Constraints

Methodology for Tuning and Optimizing

- **Define the problem**
- **Define the goal**
- **Capture appropriate metrics related to the problem**
 - Represent “typical” performance
 - If you don’t do this first, how can you quantify improvement?
- **Begin tuning**
 - This is an iterative process
 - Capture the same metrics again throughout
- **When finished, capture the EXACT SAME metrics again**
 - This seems really obvious, but it’s extremely easy to get distracted

Key Takeaways

- **When troubleshooting and tuning, start with specific, measurable facts about the problem**
 - “The database is slow” is not specific or measurable
- **Baselines are essential when investigating a performance issue**
 - At a minimum, establish methods for capturing server, instance, and database settings on a regular basis
- **Don’t knee-jerk when troubleshooting**
 - Having a library of scripts available creates a methodical process to follow, even when it seems like everything is on fire
- **There is a lot of data available in SQL Server**
 - Understand where data is located and how to find it
 - Use multiple data sources to corroborate a finding

Additional Resources

■ Pluralsight courses

- SQL Server 2017: Diagnosing Configuration Issues with DMVs
 - <https://www.pluralsight.com/courses/sqlserver-diagnosing-configuration-issues-dmv>
- SQL Server 2017: Diagnosing Performance Issues with DMVs
 - <https://app.pluralsight.com/library/courses/sqlserver-2017-diagnosing-performance-issues-dmvs/>
- SQL Server: Installing and Configuring SQL Server 2016
 - <https://bit.ly/2yoRNyX>
- SQL Server 2014 DMV Diagnostic Queries - Part 1
 - <https://bit.ly/2qMfKdA>
- SQL Server 2014 DMV Diagnostic Queries - Part 2
 - <https://bit.ly/2qGJbNO>
- SQL Server 2014 DMV Diagnostic Queries - Part 3
 - <https://bit.ly/2J7nZIG>

Review

- How we approach tuning and optimization
- Server, instance, and database settings
- Checking system health
- Methods for analyzing performance

Questions?

