SQLskills Immersion Event

IEPTO2: Performance Tuning and Optimization

Appendix: Data Collection and Baselining

Erin Stellato

Erin@SQLskills.com

SQLskills
immerse yourself in sql server

Overview

- **Introduction to Baselining**
- **Data collection methods and tools**
    - Performance Monitor and Collector Sets
        - PAL tool (Performance Analysis of Logs)
    - DMOs and catalog views
    - SQL Trace
        - Analyzing trace data
    - SQLDiag
    - SQL Nexus

Purpose of a Baseline

- **Provide a starting point for comparison of additional data over time**
  - Often represents the “normal” or typical state of the environment
  - Helps you understand where the system is today
- **Baseline data is invaluable during a performance “crisis”**
- **Can also be used to identify usage patterns and trending, and can be extremely helpful for capacity planning**
- **Used to measure the impact of changes**
  - Increased workload
  - Code and design
  - Hardware
  - Upgrades to the OS or SQL Server
  - Test in another environment before migrating to production

Benchmark vs. Baseline

- **A benchmark measures performance using a *specific set of indicators* to determine the performance level in a way that can be compared to other systems, business requirements, or previous benchmarks**
- **It is a *standard* for comparison**
- **May be established to determine the capacity limits of a system**
  - Maximum number of concurrent connections
  - Maximum number of transactions/batches per second
  - Use to forecast replacement/upgrade requirements before exceeding limits
- **Use benchmarks and baselines to reach a target or specific goal**
  - "This stored procedure used to run in 200 ms." (*this is your benchmark*)
  - "The SP now takes 3 seconds." (*this is your baseline*)
  - Compare the baseline to the benchmark
  - Improve the current value in steps (*this is tuning*)

Data Collection Examples (1)

- **Performance Monitor**
  - Single collection of performance counters for hardware, the Windows OS and SQL Server
  - Easy to use for trending over time
- **DMV output**
  - Wait statistics
  - File statistics
  - Buffer and plan cache usage
  - Index statistics (query optimization and Storage Engine)
  - Query plans
- **Trace data**
  - CPU, reads, writes, duration
    - Extended Events now more viable for SQL Server 2012 and higher

Data Collection Examples (2)

- **Catalog views/system tables**
  - SQL Server configuration
  - Database and file size
  - Maintenance job history
  - Schema
- **Application-specific**
  - User activity
  - System work
  - Batch jobs

Baselines: Deciding Where to Start

- There is a significant amount of data you can collect from a SQL Server environment
- Start by defining a goal
- Determine what data has the most value, as it relates to your goal
- After you decide what to collect, determine:
  - When you will capture data (time of day/week/month/quarter)
  - How often to capture it (every 5 minutes/every hour/once a day)
  - Where it should be stored
  - How the data will be accessed
  - Retention duration

Overview

- **Introduction to Baselining**
- **Data collection methods and tools**
  - Performance Monitor and Collector Sets
    - PAL tool (Performance Analysis of Logs)
  - DMOs and catalog views
  - SQL Trace
    - Analyzing trace data
  - SQLDiag
  - SQL Nexus
  - Distributed Replay Utility (DRU)

Performance Monitor Basics

- Performance Monitor is built in to Windows
- Hardware, OS, and SQL Server counters can be captured
- It can be used to monitor performance real-time, or capture metrics over a period of time
- Data collection can be automated
- Data can be processed manually or automatically

Reference: OS Counters to Collect

- **Processor**
  - % Processor Time
  - % Privileged Time
- **System**
  - Processor Queue Length
- **Memory**
  - Available Mbytes
  - Pages/sec
- **Paging File**
  - %Usage
- **PhysicalDisk**
  - Avg. Disk sec/Read
  - Avg. Disk sec/Write
  - Disk Reads/sec
  - Disk Writes/sec
- **Process (sqlservr.exe)**
  - % Processor Time
  - % Privileged Time

Reference: SQL Counters to Collect

- **SQL Server:Access Methods**
  - Forwarded Records/sec
  - Full Scans/sec
  - Index Searches/sec
- **SQL Server:Buffer Manager**
  - Buffer cache hit ratio?
  - Free List Stalls/sec
  - Lazy Writes/sec
  - Page Life Expectancy?
  - Page Reads/sec
  - Page Writes/sec
- **SQL Server:General Statistics**
  - User Connections
- **SQL Server:Memory Manager**
  - Total Server Memory (KB)
  - Target Server Memory (KB)
- **SQL Server:SQL Statistics**
  - Batch Requests/sec
  - SQL Compilations/sec
  - SQL Re-Compilations/sec
- **SQL Server:Locks**
  - Lock Waits/sec
  - Number of Deadlocks/sec
- **SQL Server:Latches**
  - Latch Waits/sec

What does all of this actually tell us?

Data Collector Sets

- **Collector sets allow for repeated use**
  - User-defined vs. system
  - Can be exported/imported between servers
- **Collector sets can be started manually, via the built-in scheduler, or via command line with logman**
  - logman is available in Windows Server 2003+
- **Can be used to automate data collection as a result of a specific event or alert**

Performance Analysis of Logs Tool

- Free utility available for download from Codeplex
- Analyzes Performance Monitor counter logs using industry standard thresholds
- Includes a built in template for SQL Server created by David Pless, a Premier Field Engineer at Microsoft
  - This template can be imported into PerfMon to create a Data Collector Set
- The template, within PAL, can be customized to add additional counters or change thresholds if necessary
- Details of the individual performance counters, what they mean, how they relate to each other, and the thresholds being tested are available in the user interface

Using DMV/DMF Data for Baselines

- **There are 200 dynamic management objects in SQL Server 2014**
  - Available since SQL Server 2005
  - Provide information about the server and its databases that can be used to monitor health and performance as well as diagnose problems
- **Information does not persist between restarts**
  - One exception: sys.dm_db_index_physical_stats
  - In some cases, you can clear data without a restart
- **Snapshot data to a table at regular intervals**
  - Note that schema changes can occur between versions
- **Report on captured data as needed**

DMVs to Consider for Data Capture

- **sys.dm_os_wait_stats**
  - Aggregated waits for the instance
- **sys.dm_io_virtual_file_stats**
  - Reads, writes, latency, and current size for every database file
- **sys.dm_exec_query_stats**
  - Aggregate statistics for cached query plans including execution count, reads, writes, duration, and number of rows returned
- **sys.dm_db_index_usage_stats**
  - Cumulative seeks, scans, lookups and updates for an index
- **sys.dm_os_performance_counters**
  - Current value for SQL Server performance counters
    - For per-second counters, the value is cumulative
- **Reminder, this is not a comprehensive list**

Additional Data to Capture

- **System configuration**
  - sys.configurations, SERVERPROPERTY, DBCC TRACESTATUS, sys.databases
- **Database and file sizes**
  - sys.master_files, sys.database_files, DBCC SQLPERF
- **Database maintenance history**
  - msdb.dbo.backupset, msdb.dbo.sysjobhistory

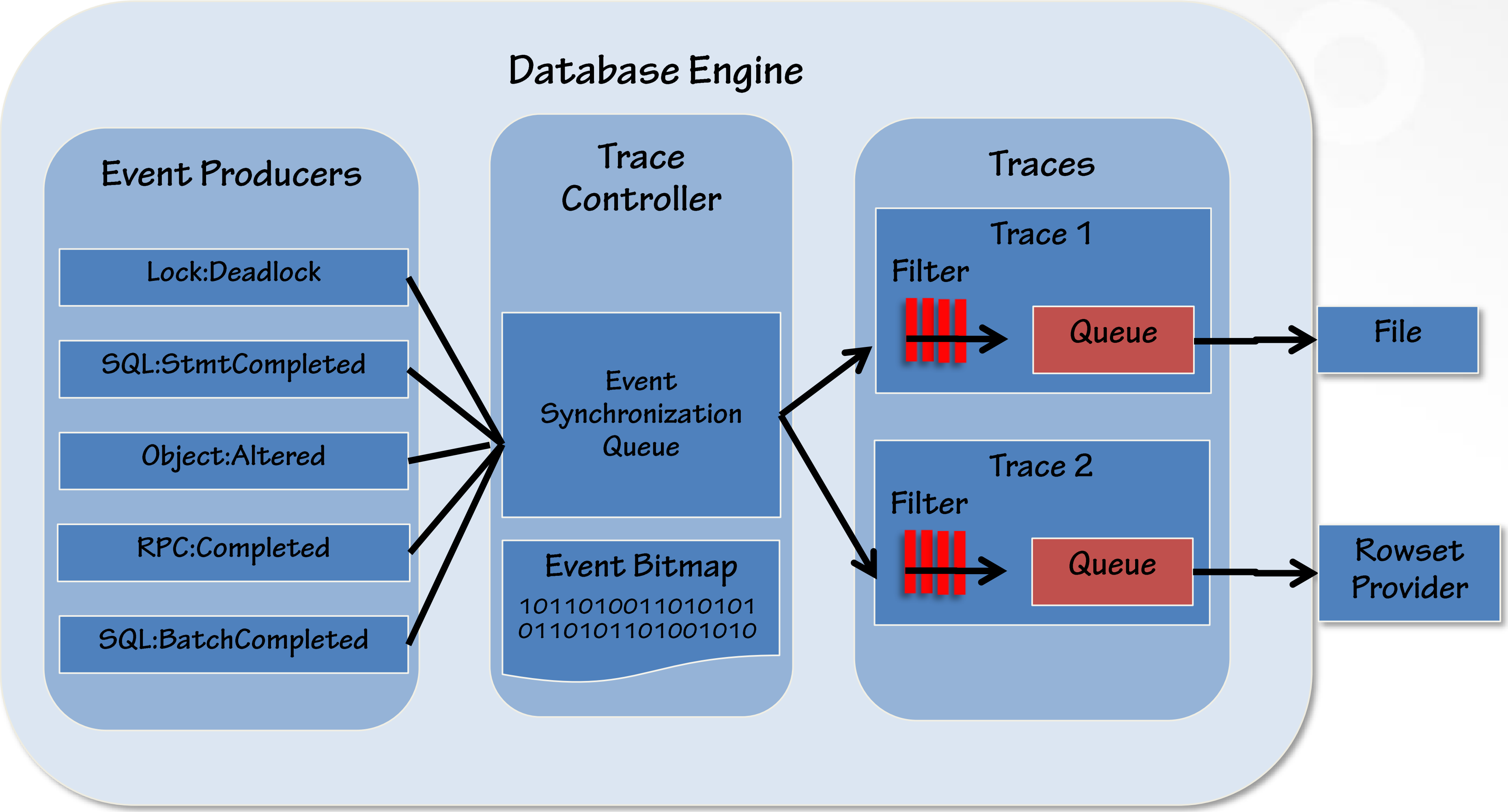
SQL Trace

- Real-time insight into SQL Server activity
- Understand duration, frequency, and resource utilization of queries
- Gather a baseline or benchmark of system activity for consolidation or load projections
- Troubleshoot application errors or performance problems
- Auditing user activity
- Watch out for "observer overhead"

How SQL Trace Works

- The trace controller inside the database engine maintains a bitmap of events that are being collected by an active trace
- Event providers check if their event is active in the bitmap and if it is, provides one copy of the event data to the trace controller
- The trace controller queues the event data and provides the event data to all active traces collecting the event
- The individual traces filter the event data removing any columns that are not needed, and discarding events not matching the trace filters
- The remaining event data is written to a file locally on the server, or buffered to the row-set provider for consumption by external applications like SMO and SQL Profiler

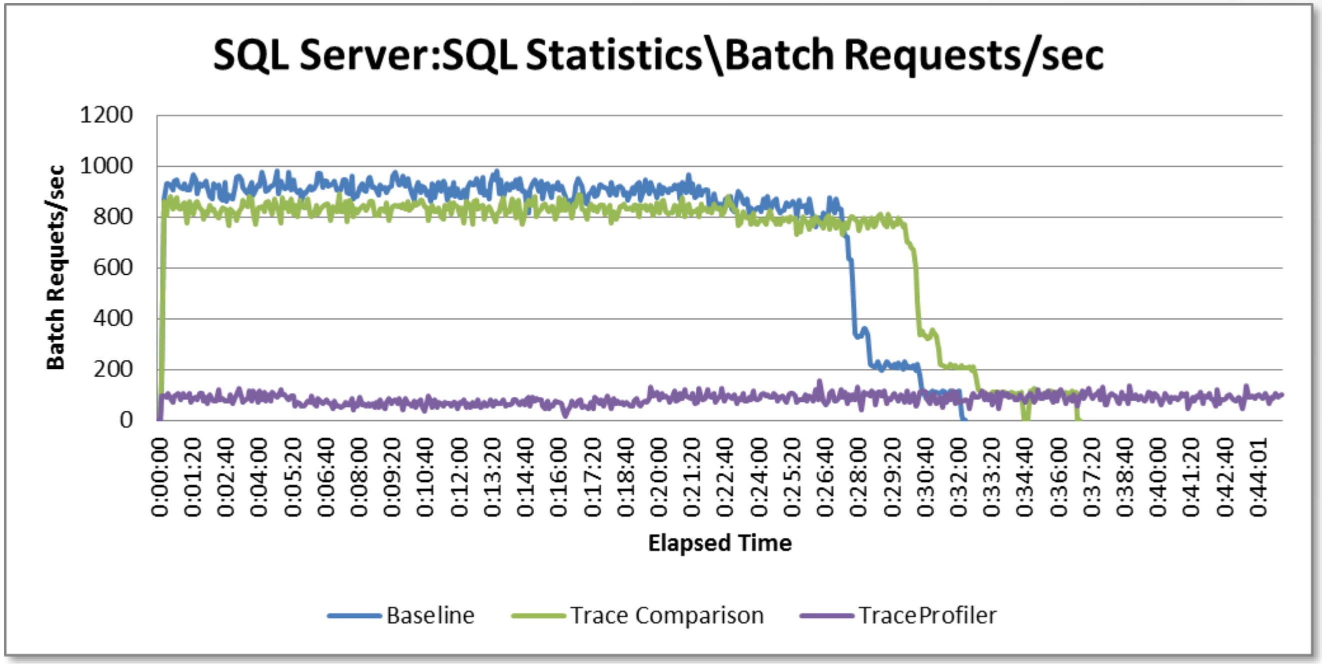
SQL Trace Architecture



SQL Trace Wait Types

- **The file provider is designed with a guarantee that no event data will be lost**
  - During I/O pressure or stalls, internal buffers begin to fill if disk writes are not keeping up
  - Once the buffers fill up, threads sending event data to the trace wait for buffer space
- **The rowset provider is not designed to make data loss guarantees**
  - If data is not being consumed fast enough, internal buffers will fill and events will be jettisoned after 20 seconds
  - SQL Server Profiler client tool sends an error message for dropped events
  - Monitor the TRACEWRITE wait type (threads waiting for free buffers)

Observer Overhead



- **Replay workload processed by Distributed Replay with 4 clients against a 4vCPU SQL Server 2012 VM with 8GB RAM**

	Duration (hh:mm:ss)	Avg. Batch Req/sec	% of Baseline
Baseline	0:32:10	896.25	100.0%
Server Side Trace	0:36:50	822.1	91.7%
Profiler Trace	5:18:50	81.18	9.1%

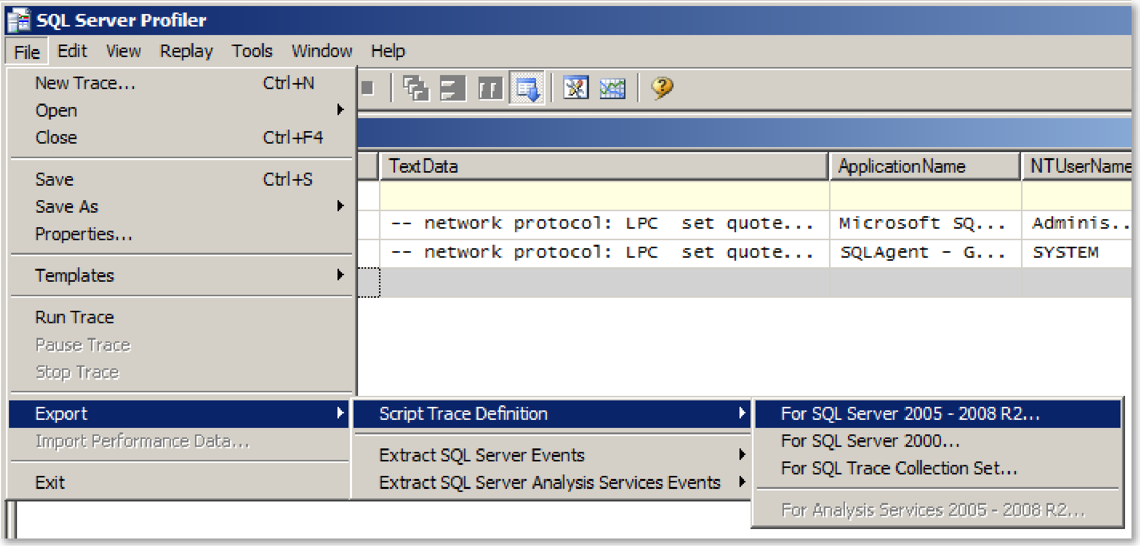
When to Use SQL Trace

- **Benchmarking or baselining where a specific workload must be captured and replayed**
  - If not for B&B: step back and ask if you can achieve the same objectives through Dynamic Management Views and Functions?
  - For DRU: SQL Trace replay data can be used, but latest RML Utilities will convert an Extended Events output file
- **In response to an error or alert**
- **Proactive tracing can be used to prevent having to wait for a problem to reoccur to see what caused it**
  - Problematic… space usage… overhead… is it worth it?

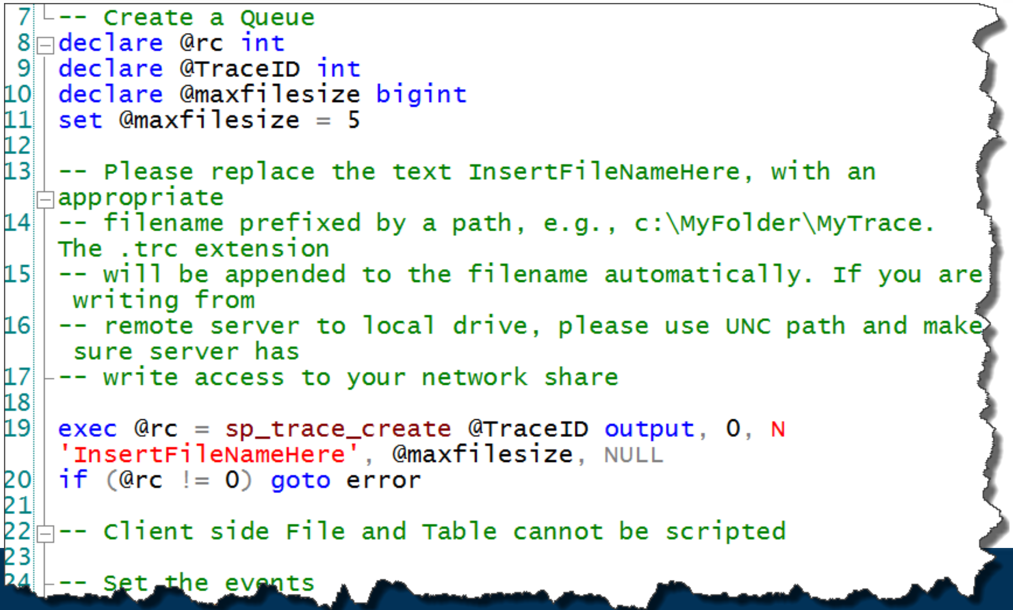
Creating SQL Trace Sessions

- **sp_trace_create**
  - Creates a trace with the provided configuration and returns the trace_id for the new trace
- **sp_trace_setevent**
  - Adds/removes an event or column to an existing trace
- **sp_trace_setfilter**
  - Applies a filter to an existing trace
- **sp_trace_setstatus**
  - Modifies the status of a trace (0-stop, 1-start, 2-delete)
- **Changing a traces definition requires that the trace be in a stopped status (status=0)**

Building Trace Scripts with Profiler



TextData	ApplicationName	NTUserName
-- network protocol: LPC set quote...	Microsoft SQ...	Adminis..
-- network protocol: LPC set quote...	SQLAgent - G...	SYSTEM



```
-- Create a Queue
declare @rc int
declare @TraceID int
declare @maxfilesize bigint
set @maxfilesize = 5

-- Please replace the text InsertFileNameHere, with an appropriate
-- filename prefixed by a path, e.g., c:\MyFolder\MyTrace. The .trc extension
-- will be appended to the filename automatically. If you are writing from
-- remote server to local drive, please use UNC path and make sure server has
-- write access to your network share

exec @rc = sp_trace_create @TraceID output, 0, N'InsertFileNameHere', @maxfilesize, NULL
if (@rc != 0) goto error

-- Client side File and Table cannot be scripted

-- Set the events
```

Automating Trace Capture

- Create a trace script as a stored procedure on the server
  - If created in the master database the procedure can be marked for automatic execution at startup with sp_procoption
- Create a SQL Server Agent Job with a schedule type of “start automatically when SQL Server Agent starts”
- Automate based on an event and then shut down after a specific period of time
- Ensure that the filename is generated dynamically and maintains uniqueness or the trace will fail to start
- Automating Extended Events sessions requires less effort

Analyzing Trace Data

- **There are multiple free tools that exist for analyzing trace data**
  - ClearTrace
  - ReadTrace
  - Qure Analyzer
- **All tools normalize ad hoc workloads and can group queries to help identify patterns (e.g. LoginName, TextData)**
- **ReadTrace and Qure allow you to compare two trace files**
  - Larger installation footprint compared to ClearTrace

Using SQLDiag to Collect Data

- **SQLDiag is a data capture utility for collecting diagnostic data from SQL Server, including:**
  - PerfMon logs
  - SQL Trace files
  - Windows event logs
  - SQL Server error logs
  - Msinfo32 information
- **Installed by default from SQL Server 2005 onwards**
  - C:\Program Files\Microsoft SQL Server\[90|100|110|120]\Tools\Binn\sqldiag.exe
- **This is what we use to drive our remote health checks for clients**

SQLDiag Configuration

- **Uses an XML configuration file**
  - Default file created on first execution
- **Can be edited using any text edit application (e.g. notepad) or the Business Intelligence Development Studio (BIDS) environment from Visual Studio**
  - Editing with BIDS simplifies editing by making subsections collapsible minimizing the viewable XML
- **Contains machine and instance level collectors**
- **Customizations must be saved as a new file name and utilized with the /I (capital-i) command line switch**
  - The default SQLDiag.xml file is overwritten at SQLDiag startup

Machine Collectors

- **Collect information from Windows Server**
- **EventLogCollector**
  - Collects Windows Event Logs for analysis
- **PerfmonCollector**
  - Collects Perfmon counters for analysis

Instance Collectors

- **Collects information for *all* SQL Server Instances installed on a server by default**
  - Can be targeted to a specific instance or multiple instances by modifying the XML configuration
- **SQLDiagCollector**
- **BlockingCollector**
- **ProfilerCollector**
- **CustomDiagnostics**

SQLDiag: Perfstats Script

- **Part of the SQLNexus project**
- **Provides multiple SQLDiag configurations for extended collection, for example:**
  - □ Adds collectors for DMV data
  - □ Captures additional blocking information

Creating a Custom Collector

- **Custom collectors can be created and added to SQLDiag by editing the CustomDiagnostics section of the XML configuration**
- **Custom collector types:**
  - □ TSQL Command
  - □ TSQL Script
  - □ Utility (.cmd files or command line strings)
  - □ VB Script
  - □ Copy File
  - □ Registry Query
- **Custom collectors can be grouped using a CustomGroup specification**

SQLDiag Command Line Options

- **/I cfgfile**
  - Sets the configuration file to use
- **/O outputpath**
  - Sets the output location
- **/N #**
  - Folder management: 1 = overwrite, 2 = rename
- **/X**
  - Snapshot mode (collect diagnostics then exit immediately)
- **/C #**
  - Sets file compression type: 0 = none, 1 = NTFS, 2 = CAB
- **/B YYYYMMDD_HH:MM:SS**
  - Sets a start time
- **/E YYYYMMDD_HH:MM:SS**
  - Sets an end time

Installing as a Service

- **/R**
  - Registers SQLDiag as a service
- **/A**
  - Sets an application name
- **/U**
  - Removes specified SQLDiag service
- **All options specified when the service registers are maintained when the service starts**
  - E.g. (sqldiag /R /A SQLDiagTuning /I C:\SQLDiagTuning\SQLDiagTuning.XML /O C:\SQLDiagTuning\Output /N 2 /C 2)
- **To control service:**
  - sqldiag START /A SQLDiagTuning
  - sqldiag STOP /A SQLDiagTuning

SQL Nexus

- Analysis tool originally developed by Ken Henderson for use by Product Support Services to simplify analysis of the information collected by PSSDiag
- Released to the community as a open source project on Codeplex
- Offline analysis of data previously collected with SQLDiag and Perfstats script only (NOT a real-time monitoring tool)

SQL Nexus Features

- **Simplified data loading**
  - SQL Trace files, T-SQL script outputs, and Performance Monitor logs
- **Simplified reporting**
  - Includes five SSRS reports for analyzing data
- **Aggregates trace data**
  - Uses ReadTrace to aggregate data to find the top most expensive queries
- **Analyzes wait stats**
  - Provides visual representation of resource contention
- **Extensible**
  - Custom reports and importers can be built and added to the application

SQL Nexus Requirements

- **Current release (4.0.0.64) requires SQL Server 2008 or higher database to import data into**
  - Supports importing SQL Server 2005 data
- **Also requires:**
  - .NET 4.0
  - RML Utilities
  - Microsoft Report Viewer Redistributable 2010
    - If you're running SQL Server 2012 or higher this is not required

Key Takeaways

- Baselines are essential to have for your system – you need to know what "normal" looks like so you have a frame of reference when problems arise
- Decide what's most important to capture based on problems you're trying to solve, or potential problems
- Start simple and work your way up
- In addition to deciding what data to collect, you also need to decide what method to use, how often to capture the information, how long to keep it, *and* you need to have a plan to look at it regularly
- All suggestions provided in this module are ones you can implement on your own
- There are third-party applications that can automate all of this and make your life much easier

Additional Resources

- **Pluralsight courses**
  - SQL Server: Benchmarking and Baselining http://bit.ly/1uUMlrw
- **Articles**
  - Collection of Baseline Scripts http://bit.ly/1MxpAHZ
  - SQL Server Central baseline articles http://bit.ly/1qL4wfk
  - Performance resources http://bit.ly/Yxxj0A
- **Glenn's DMV queries**
  - https://www.SQLskills.com/blogs/glenn/category/dmv-queries/

Review

- **Introduction to Baselining**
- **Data collection methods and tools**
  - Performance Monitor and Collector Sets
    - PAL tool (Performance Analysis of Logs)
  - DMOs and catalog views
  - SQL Trace
    - Analyzing trace data
  - SQLDiag
  - SQL Nexus