

SQLskills Immersion Event IE2: Performance Tuning

Module 12: Data Collection and Baselining

Jonathan Kehayias
Jonathan@SQLskills.com



Overview

- **Benchmark vs. baseline**
- **Data collection methods and tools**
- **Native tools:**
 - Collector Sets and Counter Logs
 - PAL Tool (Performance Analysis of Logs)
 - SQL Trace
 - Introduction Distributed Replay Utility (DRU)



2

© SQLskills, All rights reserved.
<http://www.SQLskills.com>



Purpose of a Baseline

Do You Know What “Normal” Looks Like?

- Provide a starting point for comparison of additional benchmarks over time
- Past baseline data is invaluable during a performance “crisis” and is also helpful in capacity planning
- Measure the impact of changes
 - Code and design changes
 - Increased workload
 - Hardware configuration changes
 - Upgrades to the OS or SQL Server
 - Test the changes on another server before going to production and measure the impact

Purpose of Benchmarking

What Are the Capabilities?

- Measure performance using a specific set of indicators to determine the performance level in a manner that can be compared to other systems, business requirements or previous benchmarks
- Determine the capacity limits of a system
 - Maximum number of concurrent connections
 - Maximum number of transactions/batches per second
 - Be able to forecast replacement/upgrade requirements before exceeding limits

Data Collection Examples

- **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)
- **Trace Data**
 - CPU, reads, writes, duration
 - Extended Events now more viable for SQL Server 2012

Reference: OS Counters to Collect

- | | |
|---|--|
| <ul style="list-style-type: none">▪ Processor<ul style="list-style-type: none">□ % Processor Time□ % Privileged Time▪ System<ul style="list-style-type: none">□ Processor Queue Length▪ Memory<ul style="list-style-type: none">□ Available Mbytes□ Pages/sec▪ Paging File<ul style="list-style-type: none">□ %Usage | <ul style="list-style-type: none">▪ PhysicalDisk<ul style="list-style-type: none">□ Avg. Disk sec/Read□ Avg. Disk sec/Write□ Disk Reads/sec□ Disk Writes/sec▪ Process (sqlservr.exe)<ul style="list-style-type: none">□ % 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:Locks**
 - Lock Waits/sec
 - Number of Deadlocks/sec
- **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:SQL Statistics**
 - Batch Requests/sec
 - SQL Compilations/sec
 - SQL Re-Compilations/sec
- **SQL Server:Latches**
 - Latch Waits/sec

What does all this actually tell us?

Performance Analysis of Logs Tool

- Free tool on Codeplex designed to analyze Performance Monitor counter logs using industry standard thresholds
- Provides a built in template for SQL Server created by David Pless, a Premier Field Engineer at Microsoft
- Template can be used to create a Data Collector Set template that can be imported into PerfMon
- 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
- The template can be customized to add additional counters or change thresholds if necessary

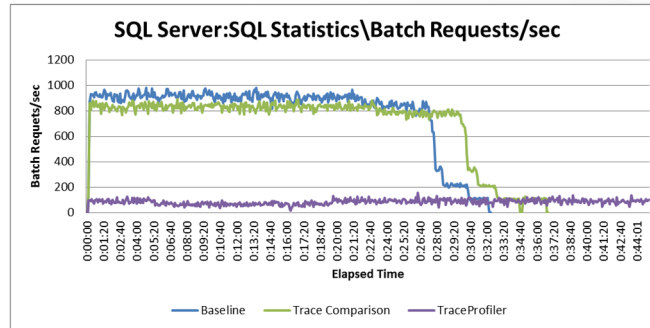
Demo

Data Collector Sets and Performance Analysis of Logs

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
- But watch out for “observer overhead”

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%

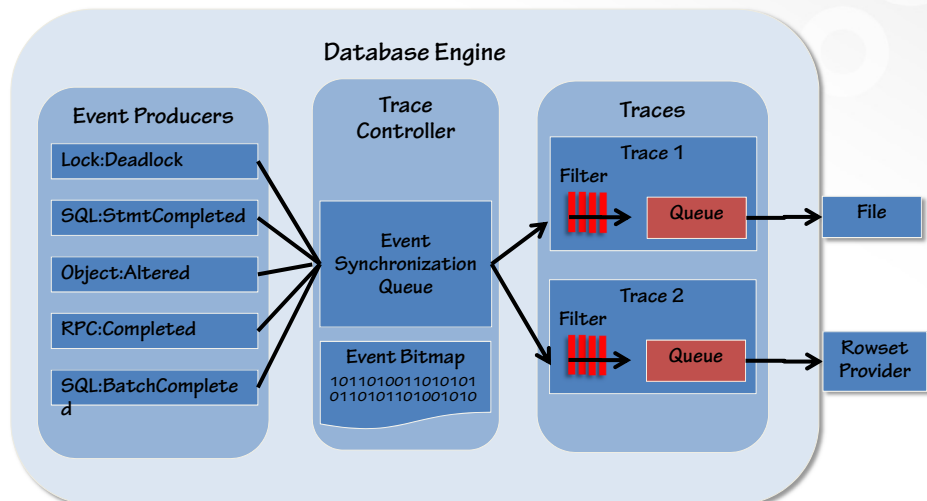
When to Use SQL Trace

- 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?
 - Even with DRU: SQL Trace replay data is still needed
- In response to a 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?

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



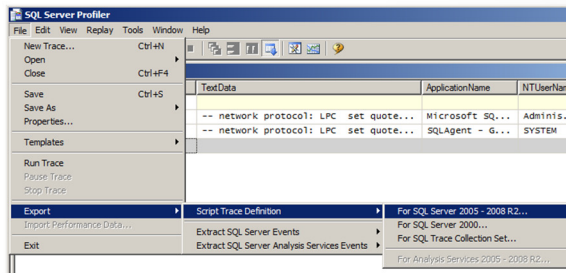
SQL Trace Wait Types

- **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)

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 trace's definition requires that the trace be in a stopped status (status=0)**

Building Trace Scripts with Profiler



```
7  -- Create a Queue
8  declare @rc int
9  declare @TraceID int
10 declare @maxfilesize bigint
11 set @maxfilesize = 5
12
13 -- Please replace the text InsertFileNameHere, with an
14 -- appropriate
15 -- filename prefixed by a path, e.g., c:\MyFolder\MyTrace.
16 -- The .trc extension
17 -- will be appended to the filename automatically. If you are
18 -- writing from
19 -- remote server to local drive, please use UNC path and make
20 -- sure server has
21 -- write access to your network share
22 exec @rc = sp_trace_create @TraceID output, 0, N
23 'InsertFileNameHere', @maxfilesize, NULL
24 if (@rc != 0) goto error
25 -- Client side File and Table cannot be scripted
26
27 -- 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`
 - Another option is a SQL Server Agent Job with a schedule type of "start automatically when SQL Server Agent starts"
 - Or you can 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 of the trace will fail to start**
- **Similar technique can be used with Extended Events session**

Demo

Automated profiling on increased IO waits

Analyzing Trace Data

- ReadTrace
- ClearTrace
- Qure Analyzer

Trace Analysis

Using SQLDiag to Collect Data

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

SQLDiag Configuration

- **Utilizes XML configuration file**
 - Default file created on first execution
- **Can be edited using any text edit (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 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

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

Creating a Custom Collector (1)

- **Custom collectors can be created and added to SQLDiag by adding to 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**

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

Demo

Using SQLDiag

SQLDiag: Perfstats Script

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

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

- Requires SQL Server 2005 or higher database to import data into
- Requires RML Utilities (ReadTrace)
- Microsoft Report Viewer Redistributable 2008 SP1

Demo

Using SQL Nexus

SQL Server 2012

Introduction to Distributed Replay Utility

- **Capture a trace and replay from multiple computers**
 - Profiler limited to replay from one computer
- **Topology:**
 - Distributed Replay controller
 - Orchestrates actions of clients (can be only one)
 - Distributed Replay clients
 - Multiple clients (≤ 16) work together to simulate workload
 - Distributed Replay administration tool
 - DReplay.exe used to talk to controller
 - Target server
- **Settings**
 - Synchronization mode (replay in order of original event – synchronized across clients)
 - Stress mode (drive load, and no synchronization across clients)
 - Can decrease “think time” and “connect time”

Review

- **Benchmark vs. baseline**
- **Data collection methods and tools**
- **Native tools:**
 - Collector Sets and Counter Logs
 - PAL Tool (Performance Analysis of Logs)
 - SQL Trace
 - Introduction Distributed Replay Utility (DRU)

Questions?

