

Performance, Baselineing, Benchmarking and Monitoring



Presented by Kevin Kline
SQL Server MVP since 2004

Agenda

- Intro Question
- Presentation
 - What's a baseline & a benchmark?
 - Building the baseline
 - Running the baseline
 - Interpreting results
 - Questions as needed
- Q & A

Kevin Kline



- *Technical Strategy Manager for SQL Server Solutions, Quest Software Inc.*
- Microsoft SQL Server MVP
- President of the International SQL Server User's Group – PASS
- SQL Server expert and author of the O'Reilly titles "SQL in a Nutshell" and "Transact-SQL Programming"



Introduction

- What will we cover today?
 - What is a baseline and a benchmark?
 - How to construct and perform a benchmarking test?
 - How to collect and assess results from benchmark test?
 - How to perform long-term, 24x7 monitoring?

Background

- Baseline = A known value against which later measurements can be compared
- Benchmarks are baselines at known and defined levels of load.
- Benchmarking facilitates better IT service, especially better *service monitoring*.



The importance of service monitoring

- Service monitoring = observing and assessing the health of a service in real-time.
- Enables you to observe service behavior proactively and quantitatively. Then, manage to that standard.
 - Allows you to avoid depending on “user experience” as the key indicator for performance
 - Helps find problems even when users aren’t on the system
 - Provides “consistency of service”

Goal of the Baseline

- Baselines:
 - Make you familiar with the operational behavior of each app/server.
 - Clearly document what is “normal” for a server and/or application.
 - Identifies types of problems that arise even when the server is behaving normally:
 - Some problems require a response.
 - Some problems have no response.
 - Tells you all about the performance of a server under normal conditions.
 - Document and understand as many as possible (if not all) background



Process for the Baseline/Benchmark Test

1. Step 1 – clearly define the business requirements, including functional, performance, and scalability requirements and translate them into system specs
2. Step 2 – generate a workload script, accounting for user load and transaction mix
3. Step 3 – stabilize the environment of and enable monitoring on monitored server
4. Step 4 – execute baseline/benchmark test (benchmark tests scale up in phases with each iteration)
5. Step 5 – assess the results

Step 1 – Requirements & Specs

1. Business requirements:
 - a. “system cannot be down for more than 45 minutes”
 - b. “report must return results within 30 seconds”
 - c. “data must be kept live for 3 months”
 - d. “backups must be stored offsite for 7 years”
2. ...Translate into system specifications:
 - a. “must sustain 500 transactions per second”
 - b. “under a concurrent user load of 120 users”
 - c. “database size is expected to be...”
 - d. “maximum read burst speed will be...”

Step 2 – Generate workload

1. Good: Reuse an industry standard test like TPC-C, TPC-E, or TPC-H
2. Better: Create a transaction generation utility or use “SQL Hammer” from the SQL Server ResKit.
3. Best: Trace real user activity using SQL Profiler for replay
 - a. Trace *textdata* and *starttime*
 - b. Substitute placeholders for random variables
 - c. Add think times (a.k.a. latency)

Step 3a – Stabilize the Environment

1. Prepare & document the OS - Set any Windows Registry settings in question like:
 - a. IO Page Lock Limit, Diskperf, or Large File System Cache
 - b. Pagefile.sys
2. Prepare & document the database platform - Set SQL Server and database settings like:
 - a. sp_configure settings like parallelism, affinity mask, lightweight pooling, max worker threads – *when in doubt, don't adjust!*
 - b. Pre-defined atabase size, indexes, fill factor – *has an enormous impact on system IO*
 - c. Fully document the database schema
3. Prepare a separate monitoring server/workstation

Step 3b – Monitor the Environment

1. Focus on two tools from a monitoring server – PerfMon and SQL Profiler
 - a. Windows Performance Monitor
 - 1) Capture SQL Server objects (e.g. SQLServer:Locks), plus Memory, Physical Disk, Process, and Processor
 - 2) Be sure to use a SQLServer:User Settable Query (a tracer query)
 - b. SQL Profiler or server-side traces
 - 1) Events – stored procedures: rpc completed, TSQL BatchCompleted
 - 2) Data Columns – TextData, Duration, CPU, Reads, Writes, ServerName, ApplicationName, EndTime
 - 3) Filters – System IDs, Duration >= Xms
 - c. Save results to a DB on the monitoring server. Make sure to add indexes to default tables. (See scripts.)

PerfMon Counters, OS

- Memory – Pages/sec
- Network Interface – Bytes total/sec
- PhysicalDisk - Disk Transfers/sec
- PhysicalDisk – Ave Queue Length/sec
- Processor - % Processor Time

PerfMon Counters, Database IO

- SQLServer:Access Methods - Full Scans/sec
- SQLServer:Buffer Manager – Buffer Cache Hit Ratio
- SQLServer:Databases - Log Growths
- SQLServer:Databases Application Database - Percent Log Used
- SQLServer:Databases Application Database - Transactions/sec

PerfMon Counters, Locking

- SQLServer:Latches – Average Latch Wait Time
- SQLServer:Locks – Average Wait Time
- SQLServer:Locks – Lock Waits/sec
- SQLServer:Locks - Number of Deadlocks/sec

PerfMon Counters, General Health

- SQLServer:General Statistics - User Connections
- SQLServer:Memory Manager - Memory Grants Pending
- SQLServer:Memory Manager – Page Life Expectancy
- SQLServer:User Settable – Query (a tracer query)

Step 4 – Test!

1. Run the baseline/benchmark test while watching your monitoring tools
 - Scale up gradually using tools like SQLCMD.EXE
 - Be ready to abort when things go sideways
2. Record everything you can
3. Consider artificial throttles
 - It's *hard* to stress current generations of hardware
4. Between iterations, only change one element of the test at a time



Demonstration #1

Using PerfMon & Profiler

PerfMon Best Practices

The best practices reduce the performance overhead and impact:

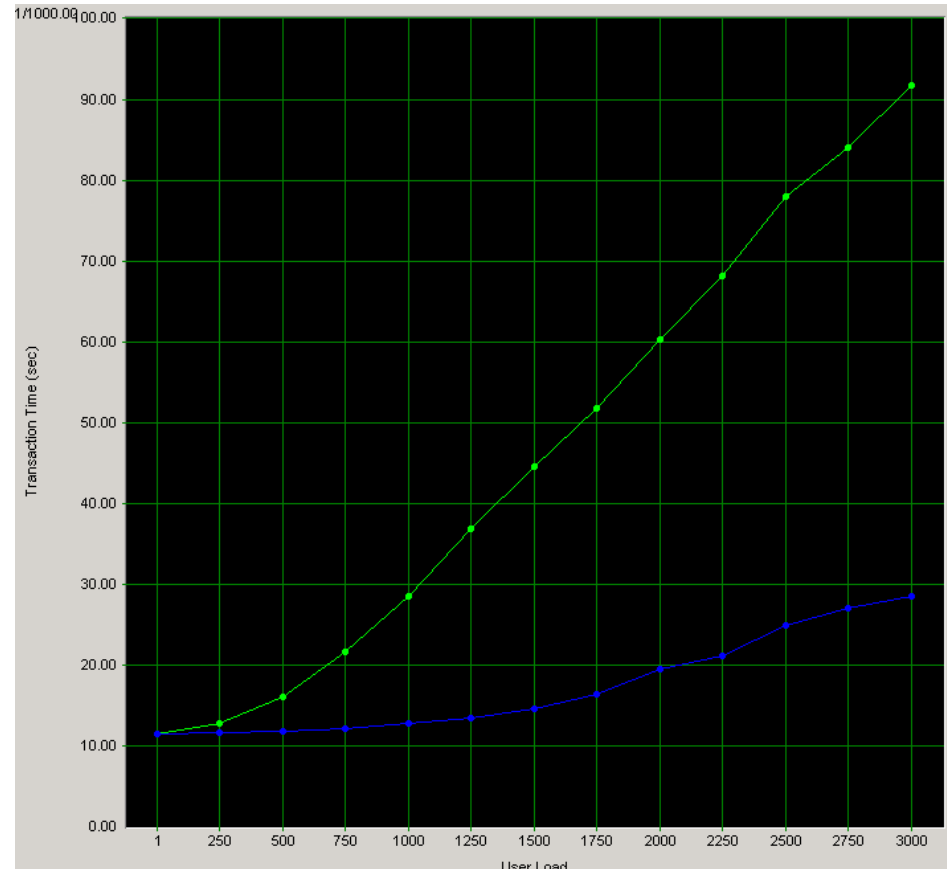
1. Minimize the number of counters you are monitoring. Less monitoring = less overhead.
2. Run counter logs from a different machine than monitored server
3. Write counter logs to a different physical disk than the one you are monitoring.
4. Pay attention to polling frequency. Don't monitor more counters or more often than necessary.
5. When experiencing extreme overhead, log directly to disk using/without watching the System Monitor Graph, since writing the log file to disk requires fewer resources.

Step 5 – Assess and Interpret

1. Correlate PerfMon counters to SQL Profiler and various DMVs
 - `sys.dm_os_memory_caches`, `xxx_clerks`, `xxx_objects` for all things memory
 - `sys.dm_exec_query_stats` to skip SQL Profiler in some situations.
 - `sys.dm_os_threads` to figure out thread memory consumption.
 - Many, many more (e.g. Greg Low's session here at TechEd)
2. Get immediate feedback with the new Performance Dashboard
3. Use PerfMon data to build generalized graphs of performance and SQL Profiler to drill down on possible culprits of bad performance

Building Meaningful Graphics

1. Some *low* numbers are obviously good (e.g. transaction time) or bad (e.g. cache hits)
2. Some *high* numbers are obviously good (e.g. trans/second) or obviously bad (e.g. deadlocks/sec)
3. And some numbers don't mean anything without context!





After Testing?

- After testing is complete, you now have a basis for providing “context” for the system’s performance in the future.
- Alert other teams of important thresholds you’ve discovered.
- Document all findings.
 - Note impact of important business cycles
 - Note impact of scheduled preventative maintenance, such as backups, DBCC processes, and index defragmentation
- Implement ongoing 24x7 monitoring.



Documenting the Final Analysis

Remember the axioms of the scientific community when testing. You should test...

- ... without bias or prejudice
- ... with clearly defined assumptions
- ... with a clearly-stated objective
- ... with an easily reproducible test case or benchmark
- ... while measuring and recording metrics accurately
- ... on relevant key differentiators
- ... using correct logical/statistical inference
- ... using proper attribution to sources or external references



Ongoing 24x7 Monitoring

- Is an important, if not the most important, component of proactive monitoring.
- If nothing else, use PerfMon set to 15 minute polling frequency checking on these counters:
 - Memory – Pages/sec
 - Network Interface – Bytes total/sec
 - Physical Disk – Disk Transfers/sec
 - Processor - % Processor Time
 - SQLServer:Access Methods - Full Scans/sec
 - SQLServer:Buffer Manager – Buffer Cache Hit Ratio
 - SQLServer:Databases Application Database - Transactions/sec
 - SQLServer:General Statistics - User Connections
 - SQLServer:Latches – Average Latch Wait Time
 - SQLServer:Locks - Average Wait Time
 - SQLServer:Locks - Lock Timeouts/sec
 - SQLServer:Locks - Number of Deadlocks/sec
 - SQLServer:Memory Manager - Memory Grants Pending

Alerting

- Alerts are defined events that raise a notification of some kind in the SQL Server Alerts.
- But which alerts to raise? At a minimum, use this reference list of alerts:
 - Errors affecting service – specifically errors with a severity of 19 to 25!
 - Deadlocks
 - CPU utilization
 - Disk utilization
 - Full table and index scans
- For extra simplicity, use “Performance Condition Alerts” instead of PerfMon counters



Demonstration #2

Alerting and Event Forwarding



Demonstration #3

A Quick Look at the TPC Disclaimers

Summary

- Build a performance baseline of server & application
- Build benchmarks for better understanding of server & application performance. Review and analyze!
- Use PerfMon to get a general understanding of performance and SQL Profiler for root-cause analysis
- Use PerfMon counters judiciously. Less monitoring = Less overhead.
- Perform on-going monitoring
- Use SQL Server Alerts and Event Forwarding to stay on top of a large SQL Server environment

Additional Resources

- Microsoft SQL IO stress test utility - <http://support.microsoft.com/?id=231619>
- SQL Server Performance Dashboard - <http://www.microsoft.com/downloads/thankyou.aspx?familyId=1d3a4a0d-7e0c-4730-8204-e419218c1efc&displayLang=en>
- SQLDiag and SQLNexus – <http://msdn2.microsoft.com/en-us/library/ms162833.aspx> and <http://www.sqlnexus.net>
- Tool Time Forum - <http://sqlforums.windowsitpro.com/web/forum/categories.aspx?catid=169&entercat=y>
- The Transaction Processing Counsel – <http://www.tpc.org>
- SQLBlog.com, especially Linchi Shea



Questions & Answers

Thank you for attending this session and the

More questions? Email me at kevin.kline@quest.com